WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.2246

RESEARCH ARTICLE

Optimized link state routing for quality-of-service provisioning: implementation, measurement, and performance evaluation

Hassan Sinky and Bechir Hamdaoui*

School of EECS, Oregon State University, Corvallis, OR 97339, U.S.A.

ABSTRACT

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 and deployed within our department building. 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 deemed useful for quality of service. Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, jitter, route changes, bandwidth, and overall stability, yielding much more robust, reliable, and efficient routing.

KEYWORDS

testbed implementation and measurement; cross-layer aware routing; OLSR protocol; performance evaluation and analysis; wireless mesh networks

***Correspondence**

Bechir Hamdaoui, School of EECS, Oregon State University, Corvallis, OR 97339, U.S.A.
E-mail: hamdaoub@onid.orst.edu

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 [1]. 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 as a result of 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 because 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 [2] on a mesh network that we recently built from off-the-shelf commercial components. OLSR is a proactive, table-driven, link state routing protocol for mobile and wireless multi-hop networks, and as defined in RFC 3626 [2], 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 (LQ). Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, jitter, route changes, bandwidth, and overall stability, yielding much more robust, reliable, and efficient routing.

Our contributions can be summarized as follows:

- Wireless nodes were built from off-the-shelf components, and a WMN testbed was deployed in the College of Engineering at Oregon State University.
- A software-based implementation of the OLSR protocol, OLSRd, was installed on our wireless nodes and modified for our research.
- The packet-pair technique is implemented in order to measure actual link bandwidth
- An exponential weighted moving average (EWMA) is used to keep track of link bandwidth calculations to better represent link condition over time.
- The effect of bandwidth detection on network performance is analyzed through the use of a real deployed WMN.

The rest of the paper is organized as follows. We begin by discussing related work in Section 2. Our WMN testbed is introduced in Section 3. We then, in Section 4, present OLSR and its routing metrics. In Section 5, we introduce the ETT routing metric and cover its implementation process with OLSR. Using our deployed testbed, in Section 6, we perform a series of tests to evaluate and compare the performance and stability of our implementation of the ETT metric (OLSR-ETT) with that of the ETX metric (OLSR-ETX). In Section 7, we present and analyze our results. Finally, we conclude the paper in Section 8.

2. RELATED WORK

Research in wireless networks and routing in general have mostly been simulation based rather than focused on implementation or real-world testing. Oftentimes, simulations rely on ideal scenarios that may not be preferred in accurately describing how WMNs truly react to their environment and surroundings. Many routing metrics and protocols have been introduced; however, although infrequent, few have been implemented and evaluated in real networks.

In [3], a WMN testbed is built using commercial Linksys WRT54G routers (Cisco Systems, Inc., San Jose, CA, USA) as their nodes. The main focus of this paper is to compare two routing metrics, OLSR-ETX and OLSR-ML. OSLR-ETX will be discussed later and is one of the default metrics used by OLSR. In this work, the OLSR-ML metric was implemented, which incorporates minimum loss probability into its calculations. The difference between OLSR-ML and OLSR-ETX is that the former uses a multiplicative metric for a path and the latter uses an additive metric. In other words, the path with the minimum loss probability is chosen instead of the path with the least number expected transmissions. The paper argues that using a multiplicative metric with OLSR yields better results. However, one of the disadvantages to this work is the lack of bandwidth detection in its implementation.

In [4], implementation in a real network is also used for testing purposes. A WMN is built using a combination of desktop personal computers and commercial Linksys WRT54G routers as the nodes where the ETT metric is implemented within OLSR. However, one of the drawbacks is the lack of stability and throughput analysis of the network as a result of the metrics used as well as implications of bandwidth detection in terms of quality of service. It mainly focuses on the creation of an ETT plug-in with OLSR. In addition, the ETT implementation does not use an EWMA to maintain bandwidth measurements but rather uses a normal average. With EWMA, precedence can be placed on recent data as well as considering old data. This is particularly useful in dynamic systems where providing an accurate and current representation of a link's changing bandwidth over time is necessary.

In [5], Microsoft researchers implemented the weighted cumulative expected transmission time (WCETT) metric for use in a multi-radio/channel testbed. WCETT aims to optimize throughput and packet loss rates in multi-channel paths within a network while avoiding interference caused by multiple channels. MR-LQSR, a DSR-based multi-radio routing protocol, is implemented using WCETT as weights for the links. This work uses commercial HP desktop personal computers as their nodes, which are more powerful and less mobile than our stripped-down wireless devices. In addition, unlike [5], our research focuses mainly on a single-channel WMN where multi-channel research is left for future study.

In our research, OLSR was used as the routing protocol. As we will discuss later, the inclusion of bandwidth detection in the routing protocol not only plays a crucial role in network stability and performance but also in quality of service where guaranteeing a certain level of performance is necessary. The following section discusses our WMN testbed.

3. AN EXPERIMENTAL NETWORK: THE TESTBED

We designed and built a WMN on the third floor of the EECS building at Oregon State University. Each node (i.e., wireless router) consists of an Alix.3C2 board (www.pcengines.ch) with a 500-MHz AMD Geode processor and 256-MB DDR RAM. The board uses a 512-MB 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 Wi-Fi Protected Access and Wired Equivalent Privacy 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 128 MB 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 (www.madwifi.com) 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 OLSR protocol (OLSRd, v. 0.5.5; www.olsr.org). 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 Sections 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.

4. OPTIMIZED LINK STATE ROUTING

Optimized Link State Routing protocol is a proactive, Djikstra-based routing protocol for mobile, ad hoc mesh networks that is particularly suitable for large and dense networks [2]. OLSRd also provides an optional extension to include and account for LQ information in determining shortest paths. It uses ETX 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 LQ.

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 (MPRs). MPRs are solely responsible for forwarding information regarding their MPR selectors throughout the network. This is what is called selective flooding because 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 that 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; that is, 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 LQs.

4.2. Route calculation

Standard OLSR (OLSR-HOPS) calculates routes purely based on least number of hops; that is, 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 Figure 1. Because 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 ($A \rightarrow B \rightarrow D \rightarrow E$).

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 ETX 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, LQs, and neighbor link qualities (NLQs). As each node sends out hello messages to find and detect their direct neighbors, LQs and NLQs

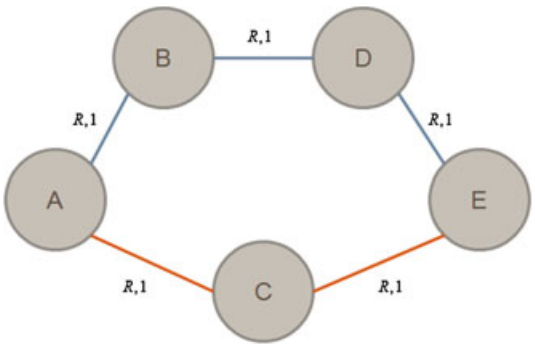


Figure 1. Shortest path routing metric: OLSR-HOPS.

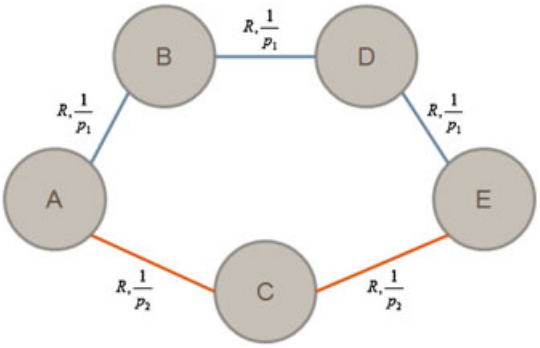


Figure 2. ETX routing metric: OLSR-ETX.

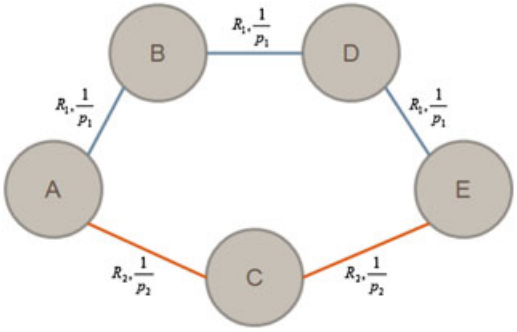


Figure 3. ETT routing metric: OLSR-ETT.

can be calculated on the basis of 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 LQ and the NLQ; by accounting for LQ and NLQ, the ETX value is the same for both directions and is

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

Topology control 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. In this work, a link's ETX value is calculated every 2 s through the use and exchange of the aforementioned hello packets.

Let us consider the same network example as shown in Figure 2. Assume that 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 shortest path while accounting for packet retransmission or finds the path with the least end-to-end delay. (Hereafter, we consider transmission delays only; we ignore all other types of delays.) 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/s is the data rate supported on each link. Thus, OLSR-ETX chooses Route 2 over Route 1 if and only if $p_2 > \frac{2}{3} * p_1$.

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 LQ 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 LQ metric, called the 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 [6–8].

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 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.

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.

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 ETT routing metric:

$$ETT = ETX * \frac{L}{R} \tag{2}$$

We can see from the aforementioned equation that ETT considers three factors in its calculation: ETX, throughput, and, because the weight of a path is equated to the summation of the ETT values across the path, 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, WMNs 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 [9]:

$$b = \frac{s}{t_1 - t_0} \tag{3}$$

In this work, we implemented the aforementioned packet-pair technique 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 (2). The packet-pair technique is performed periodically every 10 s. A per-packet approach is avoided as to minimize unnecessary network overhead.

5.2. Implementation

One of the advantages of OLSRd is that it provides a convenient plug-in interface for users. A plug-in can be created to access OLSRd data structures without modifying OLSRd'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 EWMA is then maintained for each neighbor node. These values are shared among 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 by simply broadcasting the packet-pair probes is inaccurate because the broadcast uses the IEEE 802.11 basic physical rates [4]. To resolve this issue, the use of inter-process communications 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 inter-process communications, 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. Next, we describe the metrics and experiments performed in this work.

6. EVALUATION METRICS AND TESTING

In this section, we discuss performance and stability metrics measured on our testbed. We then cover our test topology/configuration and how the experiments were conducted.

6.1. Performance and stability metrics

Overall network performance is impacted by many environmental and physical aspects and limitations as well as the choice in approaches used for routing. It is crucial to understand what factors play a role in network performance and stability and what can be done to improve them. We discuss and measure five main factors, experienced by wireless networks, that influence network stability and performance. The following subsections briefly discuss these factors.

6.1.1. Packet loss.

Wireless networks are bound to experience an amount of packet loss because of many factors including physical interference, path loss, collisions, and congestion [10,11].

Packet loss varies depending on the transmission protocol used. Generally, Transmission Control Protocol (TCP) is used for high-reliability transmissions, whereas User Datagram Protocol (UDP) is used for low-overhead transmissions. With UDP, there are no packet checks with the advantage of being faster than TCP. Applications such as voice over IP (VoIP), streaming media, online gaming and video conferencing avoid using TCP due to its long setup times and throughput debilitating properties. Naturally, UDP experiences higher packet loss than TCP since there is no mechanism to ensure reliable packet delivery. Thus packet loss plays a crucial role in accurately measuring and representing network stability/performance. In this work we evaluate both TCP and UDP packet loss.

6.1.2. Round-trip time.

Similar to packet loss, packet round-trip times (RTT) can be measured as a means to evaluate the level of network performance and stability. A network undergoing long RTTs usually is the result of congestion, packet loss and queueing delays. A path frequently taken naturally undergoes increases in data traffic resulting in rising queue levels. Queued packets can cause variations in RTTs and can not only increase delays but also packet loss. Large RTT values indicate the average queue length at intermediate nodes is large. Thus, our definition for network stability and performance includes RTTs where a lower RTT value implies less congestion and low average queue lengths. In other words, a stable and high performance network should have low average queue lengths and in turn low RTTs. In this work we evaluate the average TCP RTTs.

6.1.3. Route changes.

Frequent topology and route changes within a network tend to complicate and challenge routing. Route switches can incur an amount of routing overhead that can be detrimental to the stability of a network. Frequent route switching or flapping represents a lack of consistency in the basic topology of the network resulting in unpredictability. The time it takes for a route to be reestablished significantly and adversely impacts throughput of the TCP [12–15]. Thus, our definition for stability includes the number of route fluctuations a particular network/route undergoes. With that said, topology and route changes express a vital aspect of network stability and can militate against quality-of-service guarantees [16]. In this work we evaluate the total number of route changes.

6.1.4. Jitter.

Variation in end-to-end delay between packets causes jitter. However, a network experiencing constant latency between packets has no jitter [10]. Variable delay can be attributed to frequent topology/route changes and congestion. As a result of frequent route changes the interval at which packets are received also varies. In addition, as a result of congestion a packet may be queued or delayed on a path where there was no queueing or delay for other packets [17]. This in turn causes a variation in latency. In VoIP scenarios, high variation in packet delay of an audio stream can result in poor call quality where audio delays and dropouts are experienced. In order to minimize delay variation and provide a problem-free audio stream a jitter buffer is used to temporarily store arriving packets [17]. If packets arrive too late, they are discarded and problems arise. Thus, jitter is a major quality-of-service impairment in real-time applications and is another useful metric to measure network and broadband stability [18]. In this work, we evaluate the average UDP jitter.

6.1.5. Bandwidth.

Bandwidth constantly fluctuates and can vary depending on the transmission protocol used (i.e., TCP and UDP). If congestion is detected TCP backs off (slows down) and thus influences network bandwidth causing transmission rate fluctuations [19,20]. TCP adds latency whenever necessary to maintain equal bandwidth sharing and network stability. Using TCP may have an adverse effect on the network and provoke an unnecessary amount of overhead as well as an inaccurate representation of the bandwidth capable by our testbed. This may result in a skewed representation of the performance and stability of the network in terms of real-time applications. Thus, we measure UDP bandwidth to accurately represent real-time performance/stability of the testbed.

6.2. Test topology configuration

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

Tests were conducted over a 2 week span. Each test, on average, took anywhere between 10 min and 8 h to complete, depending on the test variables. Wireless interferences in the EECS building are naturally fluctuating; in 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 because of the interference and multiple networks present in the building. This is understandable because experimental speeds tend to be much less than theoretical link speeds [21].

Network channels are auto-set, switching from channel to channel, meaning that there was no way to set the testbed on an un-interfered channel. Because of 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

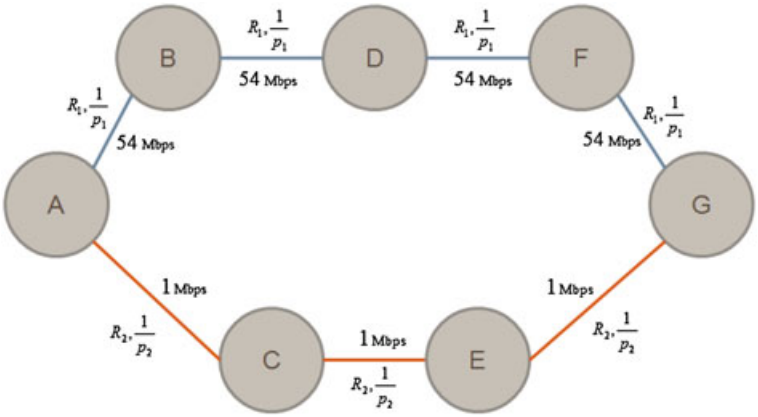


Figure 4. Test topology configuration.

shorter path were compromised and set to 1 Mbps, whereas data rates for the longer path were set to 54 Mbps as illustrated in Figure 4. Our test configuration shown in Figure 4 represents a worst-case scenario where data rates are drastically different and the need to detect them is required to achieve reasonable data transmissions. All of the tests were performed from node *A* through the terminal interface 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, RTTs, and briefly streaming video. Both laptops have OLSRd running and are part of the mesh network.

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 RTTs. 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 RTT of a ping is timed and recorded when an ECHO-REPLY is received; otherwise, a lost ping is not calculated into the average RTT. When flooding the network, ping packets are output as fast as they return or 100 times per second, whichever is more. Furthermore, for the remaining tests, we used *Iperf*. *Iperf* is a network tool used for analyzing the performance and behavior of wired and wireless networks alike. It creates data streams to perform network performance measurements and provide statistics concerning network links. It is completely written in C++. Moreover, *Iperf* can be installed very easily on any UNIX/Linux or Microsoft Windows system. One host must be set as the client and another as the server.

Our first set of tests mainly involved TCP where packet loss and RTT measurements were made using the widely used ping network tool. The performance test involved *varying the number* of pings flooded/injected into the network and measuring the percentage of packet loss as well as average RTT while fixing the size of the pings to 300 bytes. In addition, another test was performed by fixing the number of pings flooded into the network to 100 000 and *varying the size* of the pings from 100 to 1000 bytes.

Our second set of tests involved using *Iperf* to measure UDP *bandwidth*, *jitter*, and *packet loss*, mainly, measuring how well the metrics work with real-time applications. Using *Iperf*, we are given three measurement values: packet loss, jitter, and bandwidth. In Figure 4, nodes *A* and *G* are laptops, where node *A* is running an instance of the *Iperf* client and node *G* is running an instance of the *Iperf* server, respectively. UDP data is sent at a rate of 6.75 MBps, which is equivalent to 54 Mbps, from node *A* to node *G* and measurements of these three values are made. Tests are ran each time, changing the duration (minutes) of the UDP data injection to get an accurate representation of the behavior, performance, and stability of the testbed network over time. The ping network tool was also used to count the *number of route changes* experienced by the network. Pings are flooded into the network, from node *A* to node *G*, with the *-R* option to record the route taken. Once the operation is complete, the number of route changes are counted.

In Section 7, we analyze the results obtained from measuring the aforementioned stability and performance factors conducted on the testbed in Figure 4.

7. PERFORMANCE MEASUREMENTS AND ANALYSIS

Prior to performing the tests our assumptions were that OLSR-ETT would provide much shorter RTTs and less packet loss compared with OLSR-ETX because of 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 RTTs. 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.

Upon the completion of the OLSR-ETT plug-in, 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 performed 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 because the implemented OLSR-ETT was correctly detecting faster links as opposed to OSLR-ETX and thus providing a higher quality stream.

Although our testing parameters are extreme and, as mentioned earlier, express a worst-case scenario in terms of the set throughput at each node, it emphasizes the influence of throughput on a network's stability and performance. Our results are encouraging and shine a light on the importance of throughput detection within a network.

Our first set of tests yielded impressive results. Figure 5 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%. However, OLSR-ETT packet loss only ranged from 0% to 2%. Similarly, we can see from Figure 6 that, as the number of pings flooded into the network increases, the average RTT of successful pings increases rapidly with OLSR-ETX, whereas OLSR-ETT increases at a much slower rate. OLSR-ETX RTTs varied from 78.1 to 7300.4 ms. OLSR-ETT RTTs only ranged from 3.67 to 18.76 ms.

In addition, varying the *size* of the pings flooded using OLSR-ETT, we observed a much more stable and efficient network. Packet loss rarely exceeded 1% as shown in Figure 7. 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 with the previous tests ranging from 23% to 43%. As a result of OLSR-ETT, we were left with a network that saw little to no route changes and much less RTTs. RTTs remained somewhat consistent throughout the testing

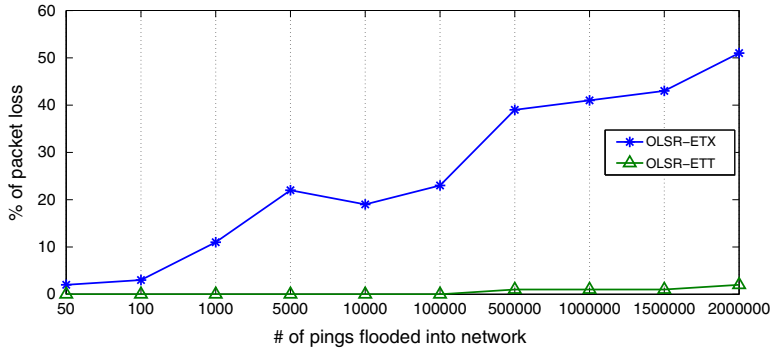


Figure 5. Packet loss as a function of the number of 300-byte pings.

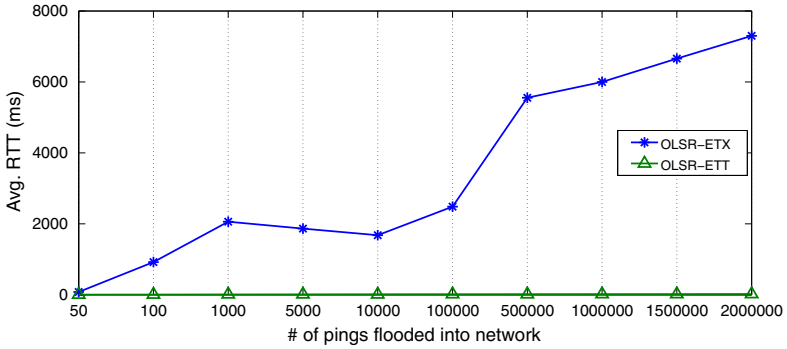


Figure 6. Average RTT as a function of the number of 300-byte pings.

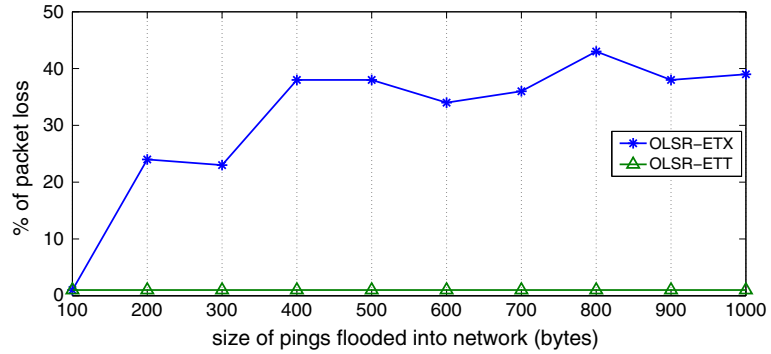


Figure 7. Packet loss measured as a function of ping sizes.

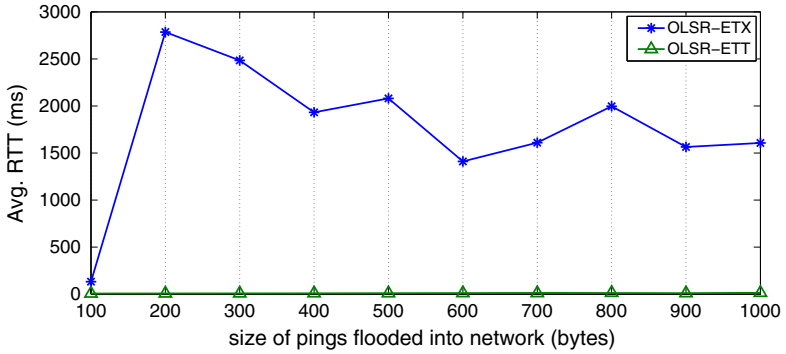


Figure 8. Average RTT measured as a function of ping sizes.

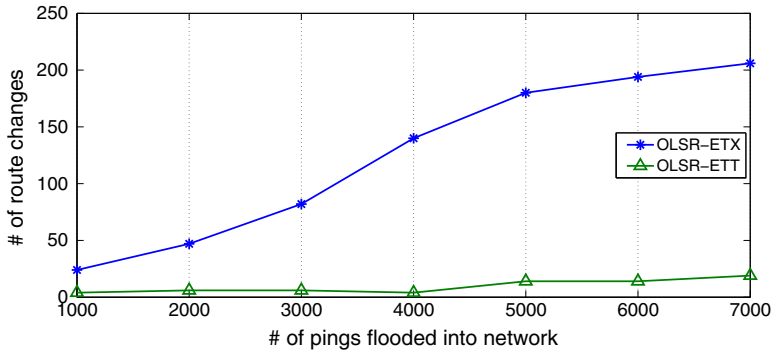


Figure 9. Route changes as a function of the number of pings.

process, ranging from 6 to 14 ms, as shown in Figure 8. This was not the case with OSLR-ETX where RTTs ranged from 132.08 to 2785.3 ms.

Our second set of tests yielded the following results where additional performance and stability evaluations were investigated. In Figure 9, we are showing the number of route changes experienced by our testbed while 300-byte pings are flooded into the network using the two metrics, respectively. As illustrated by the figure, OLSR-ETX undergoes much more topology and route changes than OLSR-ETT. The reason for this is due to OLSR-ETX's inability to detect faster links and, as a result, prefers the incapacitated route $A \rightarrow C \rightarrow E \rightarrow G$. As data is loaded onto this route at a rate of 54 Mbps, it quickly becomes congested because of the path's low data rate capabilities. Once the route is deemed incapable of handling this amount and rate of data due to congestion, packet loss, delay, and so on, the preferable path becomes $A \rightarrow B \rightarrow D \rightarrow F \rightarrow G$. Once the former route is restored and congestion dwindles, OLSR-ETX switches back to it, and the process is frequently repeated, more so than with OLSR-ETT, because of OLSR-ETX's aforementioned shortcomings. We can see that as the number of pings flooded into the network increases, so does the number of route changes

experienced by OSLR-ETX, which reaches a maximum value of 206. OLSR-ETT is not entirely immune to route changes and does experience a small amount and reaches a maximum value of 19.

In Figure 10, the results of measuring the UDP jitter as a function of time within our testbed are shown. The amount of frequent topology and route changes experienced by the network while using OLSR-ETX influences arrival times between packets resulting in a wide range of packet delay variation or jitter. Thus, the amount of route changes depicted by Figure 9 has a direct impact on the amount of jitter within a network. With OLSR-ETX, we can see that jitter is much larger than with OLSR-ETT with values ranging from 45.1 up to 103.7 ms. Whereas with OLSR-ETT jitter ranges from 0.8 ms and reaches a maximum of only 7.3 ms. With low jitter, the inter-arrival times of packets is low indicating less delay, less route changes and overall better consistency. Even though OLSR-ETX experiences high packet loss, those packets are not factored into the jitter calculation.

Figure 11 illustrates the results of measuring the UDP packet loss as a function of time within our testbed. One thing to note is that we can see that UDP packet loss is much higher than the ping tests discussed in Section 7. This is expected because UDP does not have a hand-shaking mechanism to improve reliability of packet transmissions as do TCP pings and tends to be more susceptible to packet loss. As depicted, we can see that UDP packet loss is much higher with OLSR-ETX than OLSR-ETT. As the duration of our test increases, it is evident that OLSR-ETX's packet loss increases to a debilitating state handicapping our testbed network. Without the ability to detect bandwidth, it is alarming that OLSR-ETX experiences packet loss ranging from 20% to 77% over a 50-min time span. OLSR-ETT on the other hand proves to be consistent in our scenario providing an average packet loss of less than 1%.

Finally, Figure 12 depicts the results of measuring the UDP bandwidth as a function of time within our testbed. We can see from the figure that OLSR-ETX achieves an average bandwidth slightly greater than 1 Mbps (ranging from 1.02 to 1.92 Mbps) even though the path preferred by OLSR-ETX is $A \rightarrow C \rightarrow E \rightarrow G$, which has links with bandwidths of 1 Mbps. The reason for this is that once OLSR-ETX's preferred path is overly congested and thus consists of higher ETX values, the preferred path flips to the path with links having bandwidths of 54 Mbps. Obviously, OLSR-ETX is not selecting this path because it has a higher bandwidth but because the three-hop route is too congested and dropping packets. Once the three-hop route is restored and congestion has alleviated, the preferred route reverts to the original path. During this time, OLSR-ETX is experiencing a higher data rate. This explains the average throughput being higher than the allocated bandwidth of 1 Mbps. OLSR-ETT rarely falls victim to the high data rate induced by *Iperf* providing an average throughput of around 6 Mbps (ranging from 5.78 to

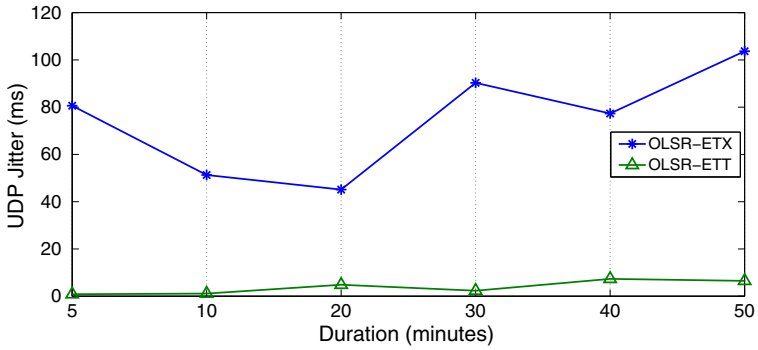


Figure 10. UDP jitter as a function of time.

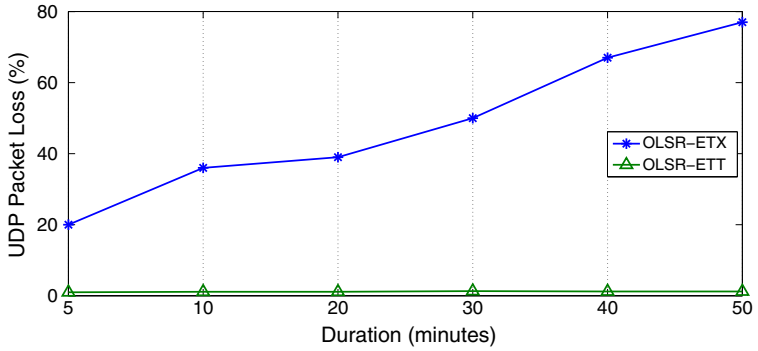


Figure 11. UDP packet loss as a function of time.

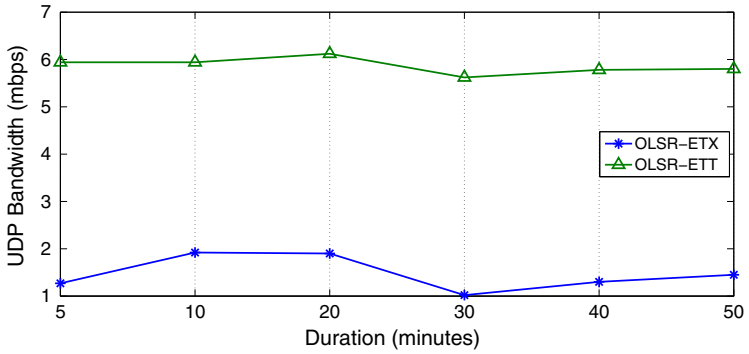


Figure 12. UDP bandwidth as a function of time.

6.1 Mbps). If bandwidth is identical or similar across the network, then bandwidth within a path should not play a factor in route calculations.

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 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 and stability 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. A majority of routing protocols today overlook the fact that an initial route may deteriorate over time and become sub-optimal resulting in instability. Our analysis may help to emphasize OLSR-ETT as a viable metric worth incorporating in future protocols.

8. CONCLUSION AND FUTURE EXTENSION

This work implements the OLSR protocol on a WMN, 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). Tests were performed to analyze and evaluate the stability achieved by the respective metrics and to emphasize the importance of bandwidth detection and its effect on network stability. 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. Because different channels are likely to support different data rates, OLSR-ETT may be well suited for WMNs that are capable of multiple channel access, enabling then more robust and efficient routing.

REFERENCES

1. Bruno R, Conti M, Gregori E. Mesh networks: commodity multihop ad hoc networks. *IEEE Communications Magazine* March 2005; **43**(3): 123–131.
2. Clausen T, Jacquet P, 2003. www.ietf.org/rfc/rfc3626.txt.
3. Passos D, Teixeira D, Muchaluat-saade D, Magalhães L, Albuquerque C. Mesh network performance measurements, In *5th International Information and Telecommunication Technologies Symposium*, Montreux, Switzerland, 2005.
4. Esposito PM, Campista MEM, Moraes IM, Costa LHMK, Duarte OCMB, Rubinstein MG. Implementing the expected transmission time metric for olsr wireless mesh networks, In *1st IFIP Wireless Days Conference 2008*, Dubai, United Arab Emirates, July 2008; 1–5.
5. Draves BZR, Padhye J. Routing in multi-radio, multihop wireless mesh networks, In *MobiCom '04 Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, Philadelphia, Pennsylvania, USA, 2004; 114–128.
6. Hamdaoui B, Shin KG. OS-MAC: an efficient MAC protocol for spectrum-agile wireless networks. *IEEE Transactions on Mobile Computing,* July 2008; **7**(8): 915–930.
7. Ma L, Han X, Shen CC. Dynamic open spectrum sharing MAC protocol for wireless ad hoc networks,

In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN 2005*, Baltimore, Maryland, USA, 2005; 203–213.
8. Wu S, Lin C, Tseng Y, Sheu J. A new multi-channel MAC protocol with on demmand channel assignment for multi-hop mobile ad hoc networks, In *ISPAN '00 Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks*, Dallas, Texas, USA, 2000; 232–237.
9. Lai K. Packet pair technique, April 2001. http://www.hpl.hp.com/personal/Kevin_Lai/projects/nettimer/publications/usits2001/node2.html.
10. Comer DE. *Computer Networks and Internets*. Prentice Hall: New Jersey, USA, 2008.
11. Manoj B, Murthy C. *Ad Hoc Wireless Networks*. Prentice Hall: New Jersey, USA, 2008.
12. Dyer TD, Boppana RV. A comparison of TCP performance over three routing protocols for mobile ad hoc networks, In *MobiHoc '01 Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Long Beach, California, USA, 2001; 56–66.
13. Holland G, Vaidya NH. Analysis of TCP performance over ad hoc networks, In *MobiCom '99 Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, Washington, USA, 1999; 219–230.
14. Ahuja A *et al.* Performance of TCP over different routing protocols in mobile ad hoc networks, In *IEEE 51st In Vehicular Technology Conference Proceedings*, Tokyo, Japan, 2000; 2315–2319.
15. Gerla M, Tang K, Bagrodia R. TCP performance in wireless multihop networks, In *WMCSA '99 Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, New Orleans, Louisiana, USA, 1999; 41–50.
16. Ilyas M. *The Handbook of Ad Hoc Wireless Networks*. CRC Press LLC: Boca Raton, Florida, 2003.
17. Systems C. Understanding jitter in packet voice networks, February 2006. http://www.cisco.com/application/pdf/paws/18902/jitter_packet_voice.pdf.
18. Munoz D, Villarreal S, Houmayoun F, Basu K. Heavy tail jitter in mobile packet networks, In *IEEE 53rd Vehicular Technology Conference*, Tel Aviv, Israel, 2001; 2224–2228.
19. Jacobson V, Karels M. Congestion avoidance and control, In *SIGCOMM '88 Symposium Proceedings on Communications Architectures and Protocols*, Stanford, California, USA, 1988; 314–329.
20. Allman M, Paxson V, Blanton E, 2009. http://www.ietf.org/rfc/rfc5681.txt.
21. Kang M, Kang D, Suh J. Real-time support on 802.11 wireless ad-hoc networks: reality vs. theory. *IEICE Transactions on Communications* 2009; **E92-B**(3): 737–744.

AUTHORS' BIOGRAPHIES

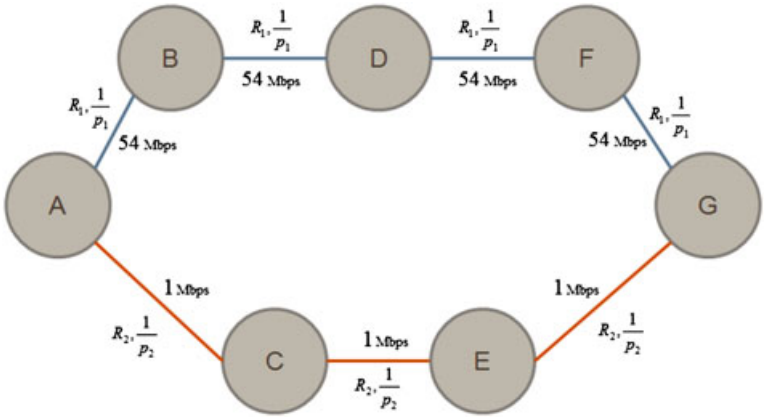
Hassan Sinky is a PhD candidate in the School of Electrical Engineering and Computer Science at Oregon State University. He received his Bachelors and Masters degrees in Computer Science at Oregon State University in 2007 and 2010, respectively. His current research interests include wireless mesh networks and vertical network handoff in heterogeneous wireless networks.

Bechir Hamdaoui received the Diploma of Graduate Engineer (1997) from the National School of Engineers at Tunis, Tunisia. He also received MS degrees in both Electrical and Computer Engineering (2002) and Computer Sciences (2004), and the PhD degree in Computer Engineering (2005) all from the University of Wisconsin at Madison. In September of 2005, he joined the RTCL Lab at the University of Michigan at Ann Arbor as a postdoctoral researcher. Since September of 2007, he has been with the School of EECS at Oregon State University as an assistant professor. His research focus is on cross-layer protocol design, system performance modeling and analysis, adaptive and learning technique development, and resource and service management for next-generation wireless networks and communications systems. He has won the NSF CAREER Award (2009), and is presently an associate editor for IEEE Transactions on Vehicular Technology (2009–present) and Wireless Communications and Computing Journal (2009–present). He also served as an associate editor for Journal of Computer Systems, Networks, and Communications (2007–2009). He served as the chair for the 2011 ACM MobiCom's Student Research Competition and as the program chair/co-chair of the Pervasive Wireless Networking Workshop (PERCOM 2009), the WiMAX/WiBro Services and QoS Management Symposium (IWCMC 2009), the Broadband Wireless Access Symposium (IWCMC 2010), the Cooperative and Cognitive Networks Workshop (IWCMC 2011 and 2012), and the Internet of Things, Machine to Machine, and Smart Services Applications Workshop (CTS 2012). He also served on program committees of several IEEE conferences. He is a member of IEEE, IEEE Computer Society, IEEE Vehicular Society, and IEEE Communications Society.

Research Article

Optimized link state routing for quality-of-service provisioning: implementation, measurement, and performance evaluation

Hassan Sinky and Bechir Hamdaoui



This work implements and evaluates the performance of the OLSR protocol on a wireless mesh network, built from off-the-shelf commercial components at Oregon State University. Two versions of OLSR were implemented, tested, and evaluated: OLSR-ETX and OLSR-ETT. Tests were performed to analyze and evaluate the stability achieved by the respective metrics and emphasize the importance of bandwidth detection and its effect on network stability. Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, and stability.