THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2010-13

Study of OLSR for Real-time Media Streaming over 802.11 Wireless Network in Software Emulation Environment

Kaustubh Jain, Kiran Somasundaram, Brian Wang,
John Baras, Ayan Roy-Chowdhury

Study of OLSR for Real-time Media Streaming over 802.11 Wireless Network in Software Emulation Environment

Kaustubh Jain, Kiran Somasundaram, Brian K. Wang, John S. Baras
Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: {ksjain, kirans, briankw, baras}@umd.edu

Ayan Roy-Chowdhury
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: ayan@umd.edu

***Abstract*—In this paper we present a setup to study the real-time traffic carrying performance of Optimized Link State Routing (OLSR) protocol using software emulation. We emulate the IEEE 802.11 MAC/PHY using the EMANE software emulator, on a cluster of machines, for different multi-hop wireless scenarios. As an instance of real-world usage scenario, we study the performance of real-time streaming media over a mesh network supported by OLSR. In particular, we study the effect of mobility and background traffic on carried load and jitter. We propose to extend this emulation setup to test the real-time performance of prototype routing protocols such as Stable Path Topology Control (SPTC) and other real-time applications.**

I. Introduction

Mobile Ad Hoc Networks (MANETs) are a collection of mobile stations that can dynamically connect, re-configure and self-organize in a completely ad-hoc manner, without any pre-established infrastructure. With the introduction of low-cost wireless technologies, such as IEEE 802.11 family of link layer protocols, there has been renewed interest in both military and commercial spaces to deploy MANETs. For example, the authors in [1] report the implementation details of a community mesh network deployment in Vienna, Austria. With increased proliferation of such community networks and other application-specific MANET deployments (such as in intelligent transportation systems, public safety and public internet access [2]), it becomes essential to obtain real-time performance estimates of these network before deployment.

Most of the work in estimating the performance has been done using network simulators such as NS2 [3] and OPNET [4], which are, in general, inexpensive and quick to collect statistics [5]. However, these simulators are not capable of providing statistics of real-time performance, which is of primary importance to the applications suggested in [2]. At the other end, there are expensive test-bed setups that provide more detailed and precise statistics [5], [6], which support analysis of real-time network parameters. In the middle ground, there is a less expensive solution: network emulation. Network emulation can provide real-time performance measurements of production-ready prototype technologies in a laboratory abstraction of real-world networks. The primary advantage of emulation is the time savings in prototype testing. Further, the ability to port emulators into general purpose clusters, also, offers a scalable solution. Among network emulators, there are two different approaches: hardware [7] emulation and software [8], [9] solutions. Apart from the time savings, software emulators can also be substantially cheaper than real test beds, or hardware emulators for that matter. We believe that software emulation, being less expensive and portable, is a good solution to obtain the performance metrics of MANET technologies before deployment.

In this paper, we study the real-time traffic carrying capabilities and performance of Optimized Link State Routing (OLSR) [10] protocol over multi-hop IEEE 802.11 networks. We choose OLSR because it is one of the most popular routing protocols for MANETs, which has been used in practical implementations [1]. Also, we propose to study the shortcomings of the protocol in terms of its control overhead and oscillations that occur due to its link-state dissemination mechanism. We use the Extensible Mobile Ad-hoc Network Emulator (EMANE) [9] to emulate the link layer and physical layer of IEEE 802.11. As a test scenario, we use streaming media to study the real-time performance of OLSR running on a MANET.

The rest of the paper is organized as follows. In section II, we provide the implementation details of the EMANE emulator setup on a general purpose cluster along with the software stack required to support the real-time streaming media. In section III, we present the results of two different scenarios. Finally, in section IV we discuss the results and proposed future work.

II. Emulation Setup

A. Cluster

We are using a cluster of 28 Sun Fire v60 machines running the Debian Lenny Linux distribution [11]. Each machine is

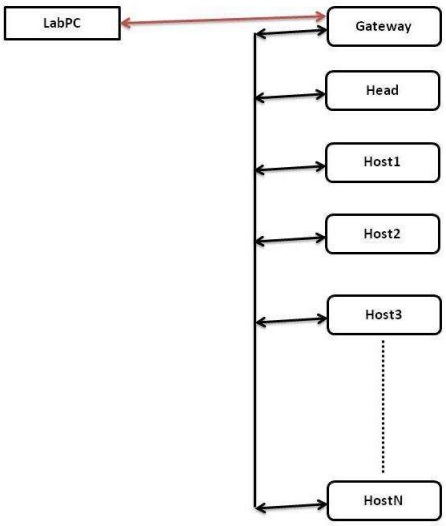

Fig. 1: The Cluster Stack

equipped with a Intel Xeon dual-core 2.80GHz processor with 1 GB RAM. Typical cluster applications such as scientific computing do not require any media output and hence the cluster machines are not equipped with sound cards. The machines form an Ethernet LAN and they are accessible via a *gateway* node that is connected to the external world (as shown in figure 1). Apart from the gateway, other machines in the cluster are labeled *head*, *host1*, *host2*,..,*hostN*. The machines are remotely accessed from lab computers using the SSH protocol to connect through the gateway node. Our objective is to use the physical cluster for wireless network emulation with a large number of nodes. In order to scale the emulation with a large number of nodes, we use the Xen hypervisor [12] to create virtual nodes. On each physical node that is used for virtualization, Xen allocates a user-specified share of system resources to create the virtual machines. Xen also creates virtual ethernet interfaces on the base machine and bridges them with the ethernet interfaces of the virtual nodes and thus they are integrated in the LAN. The virtual nodes behave as normal machines except that they have less dedicated RAM, and other hardware resources, including the processor, are shared. In our setup, in order to avoid overburdening the shared resources, we create only 4 virtual nodes on each physical node. We allocate 128MB RAM to each virtual node - this is sufficient for most intended applications. In case of a few graphics and processor intensive applications, we use cluster nodes without virtualization. Nodes *head*, *host1* and *host2* do not have any virtual nodes. *host3..host10* each have 4 virtual nodes labeled *VN_1..VN_32*. The cluster configuration with emulation nodes (a mix of virtual and real machines) is shown in Figure 2.

B. Description of the Wireless Network Emulator

We use the Extensible Mobile Ad hoc Network Emulator (EMANE) [9] as the framework for the wireless network

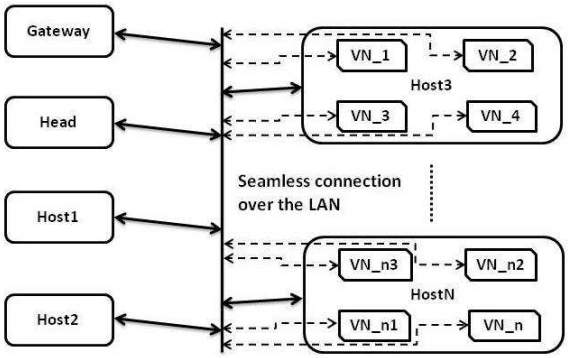

Fig. 2: Virtual Nodes in the Cluster

emulation environment. EMANE allows for heterogeneous network emulation using a pluggable media access control (MAC) and physical (PHY) layer architecture. EMANE is developed by CenGen Inc. and released under the BSD license. EMANE has been gaining recognition in the scientific and academic communities of recent due to its modular nature and ability to inter-operate with other modeling tools and even real hardware systems. The fact that it is freely available makes it even more attractive. In addition, it is open source and hence we can extend the emulator with custom-built physical and link layer models.

From the description on [9], EMANE supports emulation of simple as well as complex network architectures; provides mechanisms to bridge emulation environment control information with non-emulation aware components; supports large scale testbeds with the same ease as small test networks; and supports cross platform deployment (Unix, Linux, OSX, MS Windows). EMANE provides a set of application programming interfaces (APIs) to allow independent development of network emulation modules (NEMs), emulation/application boundary interfaces (transports), and emulation environmental data distribution mechanisms (events). Each node in the emulation is represented by an instance of an emulation stack. This stack encapsulates the functionality necessary to transmit, receive and operate on data routed through the emulation space. As shown in figure 3, each emulation stack has three components:

- Transport - Mechanism responsible for transporting packets to and from the emulation space (emulation stack entry/exit point);
- NEM - Emulation implementation logic for a given radio model; and,
- OTA - Over-The-Air Manager provides the mechanism emulation nodes use to communicate.

The PHY and MAC layer components are part of the NEM module. Radio PHY models in EMANE use a common header to allow cooperation between heterogeneous models in the same deployment. At present, two models are implemented by CenGen for the NEM module (EMANE version 0.6.2) - the RF Pipe model and the IEEE 802.11abg model. The IEEE

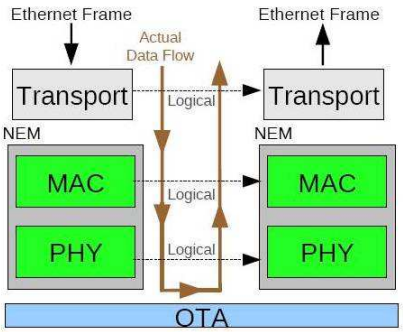

Fig. 3: EMANE Architecture. Source [9].

802.11abg model emulates IEEE 802.11 MAC layer's Distributed Coordination Function (DCF) channel access scheme on top of the IEEE 802.11 Direct Spread Spectrum Sequence (DSS) and Orthogonal Frequency Division Multiplexing (OFDM). The MAC layer features include:

- 802.11b (DSS rates: 1, 2, 5.5 and 11 Mbps)
- 802.11a/g (OFDM rates: 6, 9, 12, 18, 24, 36, 48 and 54 Mbps)
- 802.11b/g (DSS and OFDM rates)
- DCF channel access
- Unicast and broadcast (RTS/CTS/ACK capable)

The 802.11abg models PHY layer features include:

- Out of band packet filtering
- Externally computed pathloss model with real-time dissemination
- Half duplex operation
- Probability of Error (PoR) utilizing Bit Error Rate (BER) curves based on RSSI, Noise/Interference, and data rate

EMANE creates a virtual interface (labeled *emane0*) on each machine emulating a user node. Any traffic sent over the virtual interface goes through the NEM (IEEE 802.11abg in our system) and the EMANE platform server. The wireless network is modeled by the NEM and the server.

We use a centralized deployment of EMANE. All user node NEMs are connected to a single platform server, and the communication between different nodes is channeled through the central server. The EMANE server is run on a *head* node. The remaining nodes, *host1, host2, VN_1, ... VN_32*, form the nodes of the ad-hoc network. Figure 4 illustrates the deployment.

EMANE provides the ability to emulate mobile scenarios, where the node locations and the path loss between each pair of nodes can be update at one second granularity. The information are stored in flat files that are input to the emulator using the *emaneeventservice* generator. We have created different network scenarios by varying the location and pathloss values of the emulated nodes.

C. Routing Protocols and Real-time Streaming Applications

A primary advantage of our EMANE emulation setup is that we can run real-time protocols and applications easily

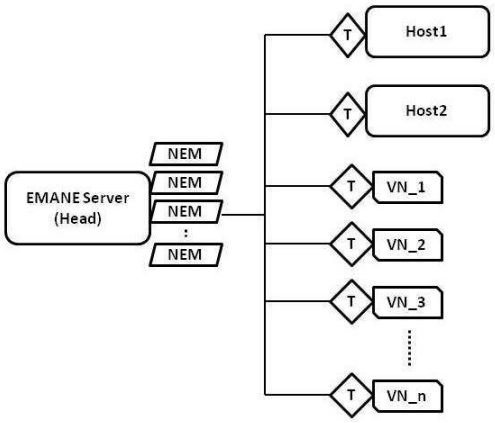

Fig. 4: Centralized Deployment of EMANE and the Virtual Interfaces. Here, NEM is the Network Emulation Modules and T is the transport deamon.

and measure their performance under the network constraints without the need for expensive emulator hardware. The primary objective of our work is to study the performance of ad hoc routing protocols in real-time scenarios. The routing protocol we are currently investigating is OLSR [10]. We run OLSR on each emulated user node, with the protocol listening on interface *emane0*. We use the open-source olsrd-0.5.5 [13] implementation. The current version of olsrd have some known bugs with the use of topology control mechanisms of OLSR. Hence, we have to run OLSR with some redundancy in broadcasted links in order to discover the routes. But the redundancy provides robust routing at a small expense of additional control overhead.

We use video streaming as an exemplary real-time application. We use VLC player [14] to stream videos between two user nodes in the cluster in a client-server setup. The video streaming is using Real-time Transport Protocol (RTP) [15]. RTP runs over UDP and provides ordering and timing information suitable for multimedia sessions. Since the machines are not equipped with sound cards, we use Pulseaudio [16] for audio tunneling. Pulseaudio is a cross-platform networked sound server acting in between the audio applications and the hardware devices. By setting up a Pulseaudio server and having a client talk to server, audio applications of the client can be made to access the server's sound devices.

We use X Server for tunneling the display. The audio and video are tunneled from the client and server cluster nodes to a remote lab machine. The protocol and application performance are analyzed within the cluster, and we use audio and video tunneling for the perceptual experience of the streaming. The overall network setup is illustrated in Figure. 5

D. Tools for Network Performance Analysis

We use Iperf [17] to send UDP traffic at specified rates between a set of source-destination pairs. This allows us to

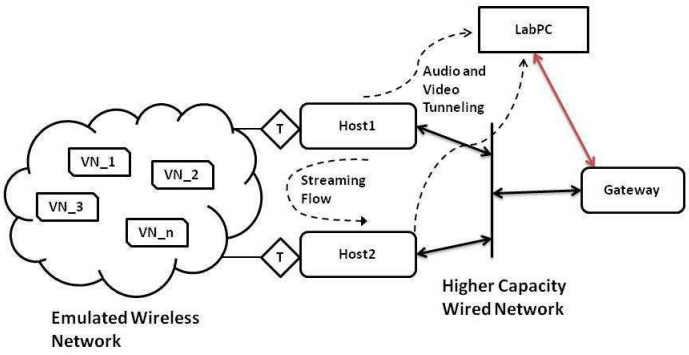

Fig. 5: Audio and Video Tunneling over Ethernet and Wireless Network over Virtual Interface

generate background traffic in the wireless network, while we analyze the performance of the routing protocol and the real-time application with varying network load. In addition, we have developed a custom application *Traffic App* that sends CBR traffic to the specified UDP connection, and analyzes the packet-loss, delay and throughput at the destination. In order to accurately measure the end-to-end delay over UDP, all the network nodes are synchronized using NTP [18], [19]. We setup the gateway node as the NTP server, and the other cluster nodes poll the NTP server. In order to keep the clock drifts minimal (and increase synchronization accuracy), we use a script to poll the NTP server frequently. We use Wireshark [20] to capture the protocol and application packets and use Wireshark's analysis capabilities for obtaining statistics on jitter, packet loss and bit rate.

III. Emulation Scenarios and Results

Our primary objective is to study the effectiveness of OLSR in establishing routes between source-destination pairs for video streaming under varying network conditions - path-loss changes due to node mobility and multipath propagation, and increasing network load. The video streaming VLC server is hosted on *host1*, while the VLC client is on *host2*. A subset of the remaining nodes in the wireless multihop network carry some background traffic. The entire 802.11 network is emulated on the EMANE platform running on the cluster. The default OLSR protocol parameters are used. For the MAC and PHY, we use 802.11b at 11 Mbps, and use RTS-CTS mechanism for sending data packets. For each scenario, we study the carried load/ achieved bitrate and jitter.

The results in this paper are using two scenarios - a 6-node clique, and a 26-node network with a mobile VLC client. The clique scenario does not need OLSR routing, but is studied mainly to verify the emulation setup without the complexities of a multihop network.

A. Single Cell Network

We set up the 6-node clique network (every node can listen and transmit to every other node) with low path-loss values. Video is streamed between a client-server pair, while the other four nodes send constant bit rate (CBR) background traffic in two source-destination pairs. We scale the background traffic uniformly and study how the streaming video quality is affected. When a low quality video ($\sim$350 Kbps MPEG video) is streamed, we see very little performance degradation in video quality. This is because, in a single cell network with near-constant traffic each transmitter gets equal share of the channel bandwidth, which in this case is more than sufficient for streaming. However, when a high quality video ($\sim$1.5Mbps) is used, we observer performance variation. Figures 6 and 7 illustrate the degradation of the streaming (reception) rate and jitter, respectively, at the client with increasing background traffic. The streaming (offered) rate at the server is on an average 1.465 Mbps (with a 90% confidence interval of $[1.402,\ 1.528]$ Mbps). The percentage of packets lost are shown in Table I. Since the path losses are low, the physical layer losses play a minimal role here. The performance degradation is primarily due to the contention and collision at the 802.11 MAC layer because of increased traffic. We also obtain a couple of screenshots of the streaming media at the client side. Figure 8 shows a good quality streaming obtained for no background traffic scenario. Whereas the poor quality screenshot in Figure 9 is taken when the loses are high due to high background traffic.

Background Traffic (in Mbps)	**0**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**
Percentage of Packets Lost	**0.1**	**2.2**	**4.1**	**9.6**	**19.4**	**30.6**	**41.3**

TABLE I: Packet Loss for the Clique Scenario

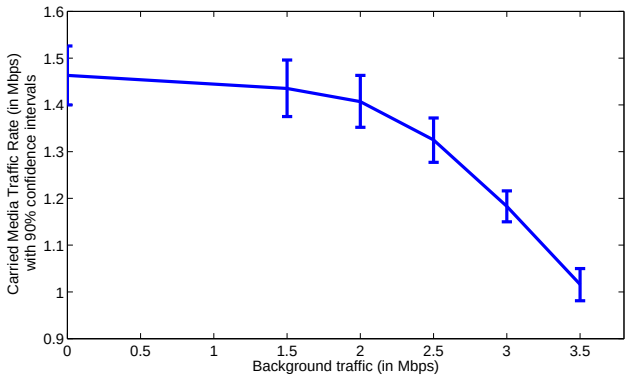

Fig. 6: Carried rate of media stream.

B. Grid Network with Mobile Client

The 26-node network is set up in a 5×5 grid, with the server at one corner of the grid, while the client is the 26^{th} node that is mobile along the diagonally opposite end of the grid (figure 10). Path-loss values are set such that each node can talk to its adjacent nodes on the grid. We study two scenarios: static, when the client is stationary at

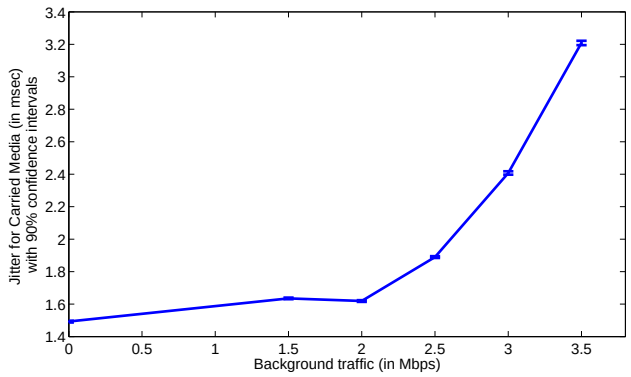

Fig. 7: Jitter of media stream.

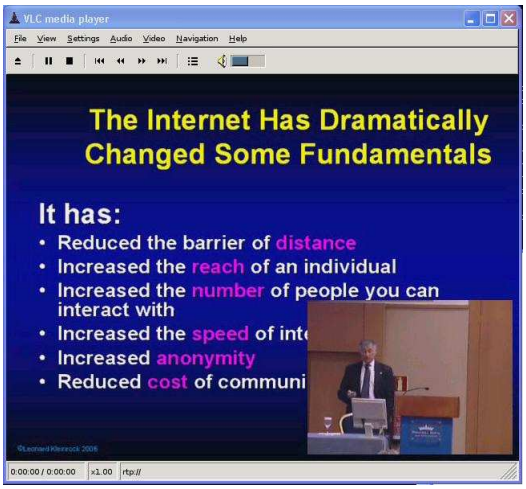

Fig. 8: Good Quality Streaming

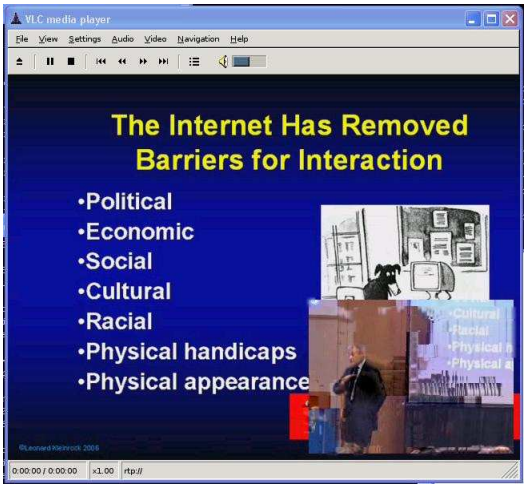

Fig. 9: Poor Quality Streaming

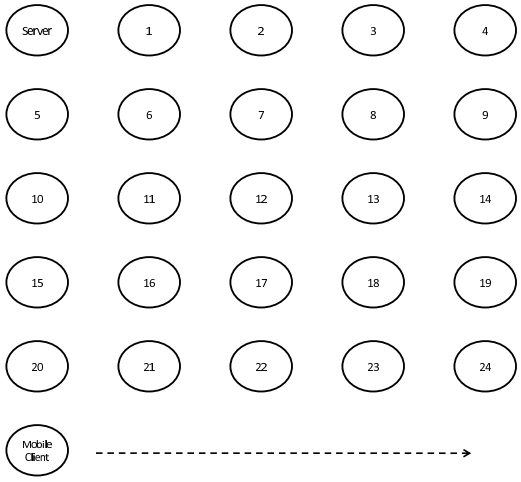

Fig. 10: Grid Network with Mobile Client

its initial position, and mobile, when the client moves at a constant speed. At any point of time, the client is within the communication range of two grid nodes. We use a low quality video (∼350 Kbps) because we observe that capacity is low for the multi-hop connection.

In this scenario, we study the impact of mobility on the streaming capabilities of OLSR. We set 5 different connections in the grid, each sending traffic at 500 Kbps. Figure 11 shows the time series statistic of the offered rate of the streaming media at the server. Figures 12 and 13 show the streaming rate and jitter at the client for the static and mobile scenarios. For the mobile scenario, we observe long periods of disconnectivity. This is due to link changes and route-detection delays of OLSR. During these periods, the jitter value cannot be obtained since no packets are received. In the static scenario, we observe 28% packet drops due to background traffic. The packet drop increases to 48% when the client is mobile.

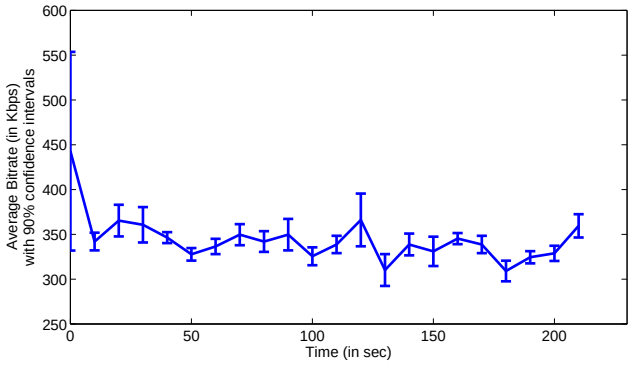

Fig. 11: Streaming (offered) rate at the server

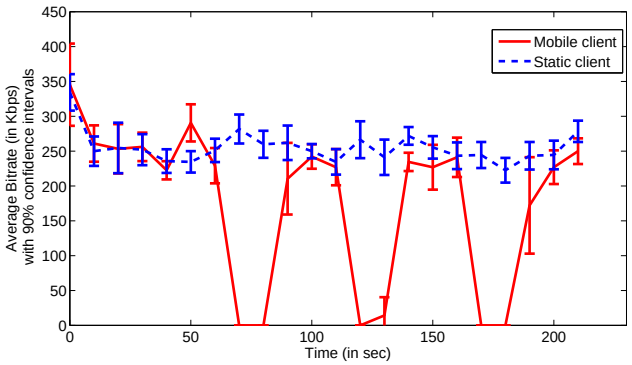

Fig. 12: Streaming (carried) rate at the client for both static and mobile scenarios

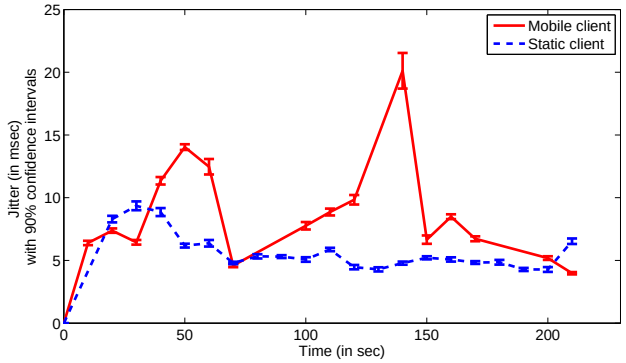

Fig. 13: Jitter of streaming media at the client for both static and mobile scenarios

IV. Discussion and Future Work

The scenarios studied show the impact of PHY, MAC and routing on the performance of streaming video: the clique scenario shows the impact of MAC, and the grid network shows the impact of all three. In particular, we see the adverse impact of OLSR routing in the grid network with mobile client scenario. Even though the mobile client maintains connectivity with at least two nodes in the grid at all times, the link-state changes take some time to be propagated.

Due to the current limitations of the emulated setup, we do not have realistic pathlosses. In the future, we plan to study the impact of a realistic wireless channel on routing and live-streaming applications. We will implement detailed physical layer models in EMANE that incorporate path losses and model various wireless fading phenomena such as flat and frequency selective fading. For the routing, we plan to use the different link quality metrics in OLSR and study their performance. We plan to implement the *Stable Path Topology Control* (SPTC) [21] routing protocol, which is a modified version of OLSR that offers better throughput guarantees, on our emulator setup. We also plan to study the performance of other real-time applications such as VoIP and video-conferencing with both OLSR and SPTC.

Acknowledgment

The authors would like to thank Dr. Vahid Tabatabaee for his advise with the experimental setup. CenGen Inc. was very helpful with answering our questions about EMANE and troubleshooting our emulation setup.

References

[1] H. Rogge, E. Baccelli, and A. Kaplan, “Packet sequence number based etx metric for mobile ad hoc networks,” IETF Draft, March 2010.
[2] R. Bruno, M. Conti, and E. Gregori, “Mesh networks: Commodity multihop ad hoc networks,” *IEEE Communications Magazine*, 2005.
[3] N. 2, “Network simulator 2,” URL. [Online]. Available: MeshNetworks:CommodityMultihophttp://www.isi.edu/nsnam/ns/
[4] OPNET, “Opnet modeler 14.5,” URL. [Online]. Available: http://www.opnet.com/solutions/network_rd/modeler.html
[5] L. Barolli, M. Ikeda, G. D. Marco, A. Durresi, and F. Xhafa, “Performance analysis of olsr and batman protocols considering link quality parameter,” *Advanced Information Networking and Applications, International Conference on*, vol. 0, pp. 307–314, 2009.
[6] E. Macias, A. Suarez, J. Martin, and V. Sunderam, “Using olsr for streaming video in 802.11 ad hoc networks to save bandwidth,” *International Journal of Computer Science*, 2007.
[7] G. Judd, X. Wang, M.-H. Lu, and P. Steenkiste, “Using physical layer emulation to optimize and evaluate mobile and wireless systems,” in *International Conference on Mobile and Ubiquituous Systems*, 2008.
[8] S. Doshi, U. Lee, R. Bagrodia, and D. McKeon, “Network design and implementation using emulation-based analysis,” in *MILCOM 2007: Proceedings of the 26th IEEE Military Communications Conference*, 2007, pp. 1–8.
[9] CENGEN, “Emane - extendable mobile ad-hoc network emulator,” URL. [Online]. Available: http://labs.cengen.com/emane/
[10] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot, “Optimized link state routing protocol (OLSR),” RFC 3626, October 2003, network Working Group. [Online]. Available: http://ietf.org/rfc/rfc3626.txt
[11] Debian, “Debian “lenny” release information,” URL, January 2010. [Online]. Available: http://www.debian.org/releases/lenny/
[12] XEN, “Xen - virtualization tool,” URL. [Online]. Available: http://www.xen.org
[13] OLSR, “Olsr routing protocol implementation version 0.5.3,” URL. [Online]. Available: http://www.olsr.org
[14] VideoLAN, “Vlc - open source cross-patform media player,” URL. [Online]. Available: http://www.videolan.org
[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “Rtp: A transport protocol for real-time applications,” RFC 1305, July 2003. [Online]. Available: http://tools.ietf.org/html/rfc3550
[16] Pulseaudio, “Pulseaudio- cross-platform networked sound server,” URL. [Online]. Available: http://www.pulseaudio.org
[17] Iperf, “Iperf,” URL. [Online]. Available: http://iperf.sourceforge.net
[18] D. L. Mills, “Network time protocol (NTP),” RFC 1305, March 1992, network Working Group. [Online]. Available: http://tools.ietf.org/html/rfc1305
[19] Mills, “Network time protocol,” URL. [Online]. Available: http://www.ntp.org
[20] Wireshark, “Wireshark: A network protocol analyzer,” URL. [Online]. Available: http://www.wireshark.org
[21] K. Somasundaram, J. Baras, K. Jain, and V. Tabatabaee, “Distributed topology control for stable path routing in mobile ad hoc networks,” Institute of Systems Research, Tech. Rep., 2009.