ISR TECHNICAL REPORT 2011-01

Studying Real-time Traffic in Multi-hop Networks Using the EMANE Emulator: Capabilities and Limitations

Kaustubh Jain
Ayan Roy-Choudhary
Kiran K.Somasundaram
Baobing Wang
John S. Baras

Studying Real-time Traffic in Multi-hop Networks Using the EMANE Emulator: Capabilities and Limitations

Kaustubh Jain, Ayan Roy-Chowdhury, Kiran K. Somasundaram,
Baobing Wang, John S. Baras
Institute for Systems Research
University of Maryland
College Park, MD, USA
{ksjain,ayan,kirans,brainkw,baras}@umd.edu

ABSTRACT

In this paper, we study the fidelity of an open-source software emulator to provide reliable estimation of performance for real-time traffic in mobile ad-hoc networks. We emulate the IEEE 802.11 MAC/PHY (DCF) using the EMANE software emulator deployed on a cluster and run experiments for different multi-hop wireless scenarios with the Optimized Link State Routing (OLSR) protocol. 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, delay and jitter. As another application, we analyze the impact of the wireless network on the self-similarity of aggregate traffic. Using traffic source models with high variability, we show that the aggregate traffic in the wireless network is self-similar and hence preserves its burstiness at larger time scales. The results are consistent with those obtained from high-fidelity simulation within some limitations of the emulator.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communications*

General Terms

Experimentation, Performance, Measurement

Keywords

Software Emulation, MANETs, Performance Evaluation

1. INTRODUCTION

In recent years, there has been growing interest in both military and commercial spaces to deploy Mobile Ad-hoc Networks (MANETs) for real-time traffic applications. For example, the authors in [16] 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, it becomes essential to obtain estimates of real-time performance metrics of these networks before deployment.

Most of the work in estimating the performance has been done using network simulators such as ns-2/ns-3 [9] and OPNET Modeler [11], which are relatively inexpensive to collect statistics in comparison to field tests. However, these simulators are not capable of providing statistics of real-time performance. A distinguishing characteristic of real-time applications is user interaction, and simulators are handicapped to capture this; simplistic traffic sources used in simulators to model such user interaction, such as http traffic models [4], cannot capture the high variability in real-time traffic. Detailed and precise statistics of real-time network statistics can be measured using testbeds [3, 8], which are prohibitively expensive. Network emulation provides a suitable alternative to both simplistic simulators and expensive testbeds. It 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 offers a scalable solution. Among network emulators, there are two different approaches: hardware emulation [7] and software solutions [2, 5, 14]. with the later being significantly less expensive than the former. We believe that a high-fidelity software emulation is an attractive solution to obtain the performance metrics of MANET protocols before deployment.

In this paper we describe the use of an open-source software emulator for real-time traffic analysis. We setup a software emulator called EMANE [5], which emulates the IEEE 802.11 MAC/PHY, on a cluster. Our setup provides a scalable platform that can emulate large wireless networks. To illustrate the limitations and capabilities of this setup, we study two scenarios. First, we study the video-streaming performance of multi-hop 802.11 network using Optimized Link State Routing (OLSR) [6], a popular MANET routing protocol [16]. Apart from the standard metrics of bitrate, delay and jitter, we look at the perceptual performance of the streaming media. Second, we study burstiness of traffic in wireless network with different traffic source models, which model real-time user interactions. Our objective here is to verify whether the emulator can provide reliable estimates of self-similarity in wireless network traffic. We use

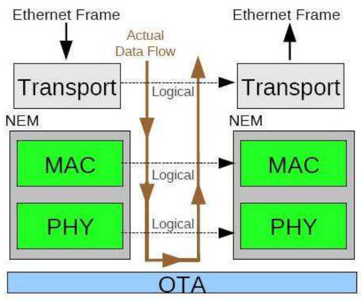


Figure 1: EMANE Architecture. Source [5].

OPNET Modeler, which is generally accepted as a mature network simulator, to compare the results of our emulator setup. These studies illustrate the limitations and capabilities of our emulation setup.

The rest of the paper is organized as follows. In section 2, we describe EMANE and provide implementation details of the emulator setup on a general purpose cluster. In section 3, we use EMANE to study the impact of OLSR and 802.11 MAC on a video streaming connection. In section 4 we study the aggregate traffic characteristics with different source traffic models. Finally, in section 5 we discuss the results obtained so far and propose future work.

2. EMULATION SETUP

2.1 Wireless Network Emulator

We use the Extensible Mobile Ad-hoc Network Emulator (EMANE) [5] as the framework for wireless network emulation. EMANE is an open source project developed by CenGen Inc. and freely available under the BSD license. It is modular and highly scalable. It can inter-operate with other modeling tools and real hardware systems. It allows for heterogeneous network emulation using a pluggable Media Access Control (MAC) and physical (PHY) layer architecture. The emulator can be extended with custom-built physical and link layer models.

A detailed description of EMANE is provided in [5], but we summarize it here. 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 emulator 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 1, each emulation stack has three components: (i) Transport - Mechanism responsible for transporting packets to and from the emulation space (emulation stack entry/exit point); (ii) NEM - Emulation implementation logic for a given radio model; and, (iii) OTA - Over-The-Air Manager provides the mechanism emulation nodes use to communicate.

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 and the EMANE platform server. The wireless network is modeled by the

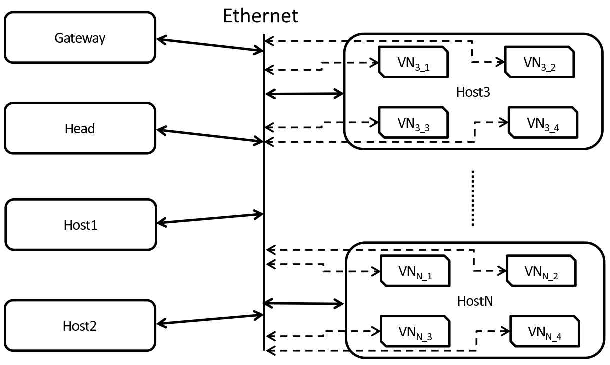


Figure 2: Virtual Nodes in the Cluster

NEM and the server. EMANE has a well developed IEEE 802.11 a/b/g model which emulates IEEE 802.11 MAC layer's DCF channel access scheme on top of IEEE 802.11 DSS and OFDM PHY. It features unicast and broadcast packets with support for RTS/CTS mode. The PHY layer computes the packet probability of error based on the specified pathlosses and BER curves. Time-varying pathlosses can be assigned to account for node mobility.

2.2 Cluster Setup

We use a cluster of 28 Sun Fire v60 machines/nodes running the Debian Lenny Linux distribution. Each node is equipped with an Intel Xeon dual-core 2.80GHz processor with 1 GB RAM. Our cluster configuration is shown in Figure 2. The physical nodes form a LAN over ethernet, and these physical nodes are accessible via a *Gateway* node, as shown in the figure. Rest of the nodes are labeled *Head, Host1, Host2,..,HostN*. Our objective is to use the physical cluster for emulation of a large wireless network. Consequently, we use the Xen hypervisor [21] to create virtual nodes. Among the physical nodes, *Gateway, Head, Host1* and *Host2* are not virtualized because these nodes are dedicated to handle processor-intensive tasks. The remaining nodes, *Host3,..HostN*, have 4 virtual nodes each. The virtual nodes for *Hostn* are labeled VN_{n_1}, VN_{n_2}, VN_{n_3} and VN_{n_4}.

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 the *head* node. The remaining nodes, *host1, host2* and the virtual nodes, form the nodes of the ad-hoc network.

2.3 Limitations of the Emulator

The current release of EMANE (release 0.6.4) is a beta release. While it has most of the important features of our interest included, there are certain limitations relevant to this study, which are listed here. In 802.11 MAC models, the simulation of packet collisions are not modeled accurately: in the interest of making the calculations in real-time, approximations are made on the message durations as well as the retry attempts. Backoff and RTS/CTS timings are incorporated in the message duration, but collisions are approximated at the level of RTS/CTS exchange. In addition,

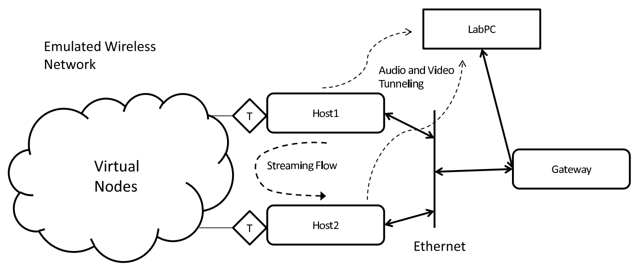


Figure 3: Audio and Video Tunneling over Ethernet and Wireless Network over Virtual Interface

the emulator does not model interference based on Signal-to-Interference-and-Noise-Ratio (SINR), but simply drops a packet if there is a collision. In the absence of collision, the packet error rate is calculated based on Signal-to-Noise-Ratio (SNR)and not SINR.

3. VIDEO STREAMING ANALYSIS

In this section, we study the effectiveness of OLSR running on the emulator in establishing routes and how it affects real-time performance. We run OLSR with default parameters on each emulated user node, with the protocol using the virtual interface *emane0*. We use the open-source olsrd-0.5.5 [10] implementation. We study the video streaming performance under varying network conditions - path-loss changes due to node mobility and multipath propagation, and increasing network load. We use VLC player [19] to stream videos between two user nodes in the cluster in a client-server setup. Since the physical nodes of the cluster are not equipped with sound cards, we use Pulseaudio [15] for audio tunneling. 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 node. The overall network setup is illustrated in Figure. 3.

The video streaming VLC server is hosted on *host1*, while the VLC client is on *host2*. A subset of the remaining nodes in the network carry some background traffic. We use an in-house traffic generator tool, which we call *Traffic App*. For this video-streaming analysis, *Traffic App* generates Constant Bit Rate (CBR) UDP traffic, i.e, generates packets of constant size and constant inter-request time. We use a packet size of 1500 bytes, and change the inter-request times for changing the offered traffic rate. 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 capture and decode all UDP packets using Wireshark [20] and analyze the carried load, delay and jitter using Matlab scripts.

We study 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. We also compare the results of the emulation setup with results obtained from OPNET. For simulating the video streaming traffic flow in OPNET, we use the actual bit-rate trace obtained at the VLC server, as a traffic flow imported in the OPNET scenario.

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 1: Packet Loss for the Clique Scenario

Before presenting the results, we would like to point out one important limitation of the OLSR code used in the emulator. For the OLSR codes released so far (upto release 0.6.0), the MPR selection algorithm / topology control mechanism does not work efficiently. The topology control mechanism is responsible for optimizing the link state information flooded in the network. So currently, all the link state information are flooded in the network. While the nodes potentially get a better view of the network, this leads to an increased overhead and also, in some cases, larger convergence time for finding fresh routes.

3.1 Single Cell Network

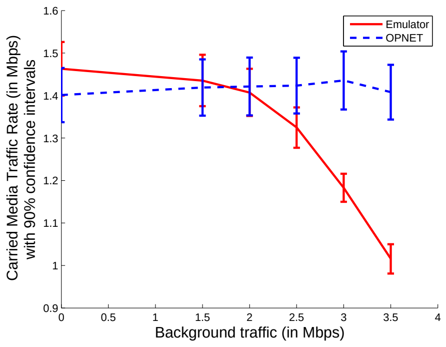


Figure 4: Carried rate of media stream for 6-node clique.

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 (∼350 Kbps MPEG video) is streamed, we see very little performance degradation in video quality for our setting. However, when a high quality video (∼1.5Mbps) is used, we observe performance variation. Figures 4, 5 and 6 illustrate the degradation of the bitrate, delay and jitter, respectively, at the client with increasing background traffic. We also notice a perceptual difference in the streamed video quality (screenshots omitted due to space constraints). The streaming 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 1. 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. When we compare the emulator results with that of OPNET, we see that the emulator under-estimates the capacity of the network because of the approximations in modeling collisions. However, the significantly lower delay in OPNET results is because it does not model the packet processing time.

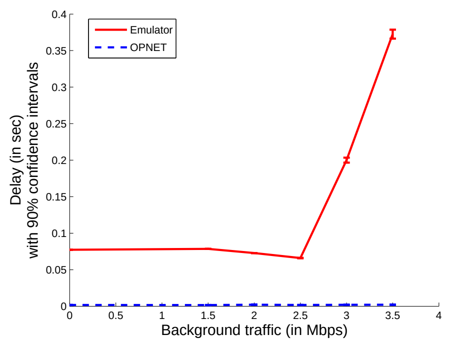


Figure 5: Delav of media stream for 6-node clique.

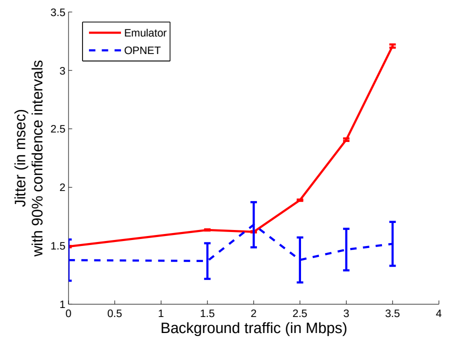
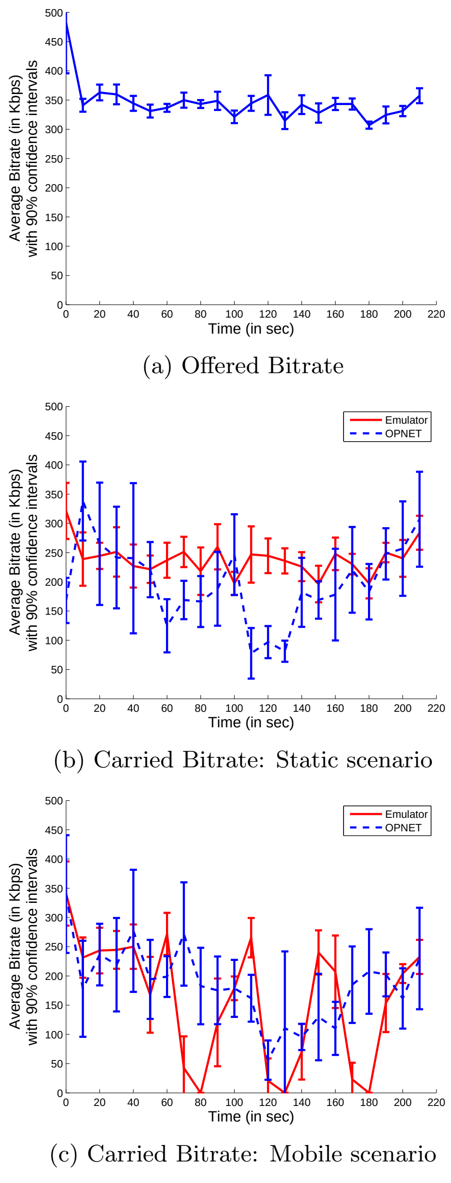


Figure 6: Jitter of media stream for 6-node clique.

3.2 Grid Network with Mobile Client

A mesh network of backbone static wireless nodes consisting of 25 nodes is deployed as a grid network, i.e., in a 5 $\times$ 5 grid topology. Let the 4 corners of the grid be annotated as A,B,C,D counterclockwise. A mobile client is also deployed alongside the grid network. For the video streaming, a serving node is chosen as one of the corner nodes of the grid, say A. The mobility pattern of the mobile client, which receives the video stream is defined so that it moves from corner B to corner C. Path-loss values are set such that each node can talk to its adjacent nodes on the grid. Hence at any point of time, the client is within the communication range of two grid nodes. We study two scenarios: static, when the client is stationary at its initial position, and mobile, when the client moves at a constant speed. We use a low quality video (~350 Kbps) because we observed 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. Figures 7, 8 and 9 respectively show the bitrate, delay and jitter between the streaming server-client. For the mobile scenario, we observe long periods of disconnectedness. This is due to link changes and route-detection delays of OLSR. During these periods, the delay and jitter values 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.

When we compare the emulator results with OPNET, we see that the delay and jitter is significantly higher in OPNET, but there are no long periods of zero bitrate at the client in mobile scenario. In the emulations, due to the absence of topology control and frequent changes in the topology, the OLSR algorithm takes more time to converge to fresh routes. This results in the zero bitrate for more than 10 sec duration in the mobile client scenario. We believe that the differences in delay and jitter are because of the differences in modeling of collisions in EMANE and OPNET.



(a) Offered Bitrate

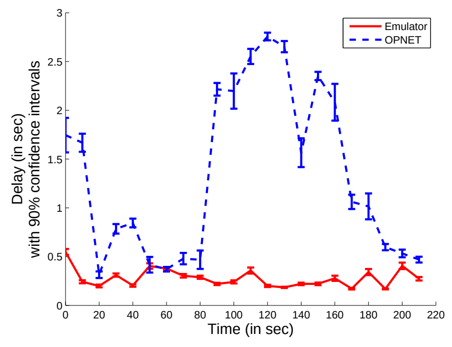
(b) Carried Bitrate: Static scenario

(c) Carried Bitrate: Mobile scenario

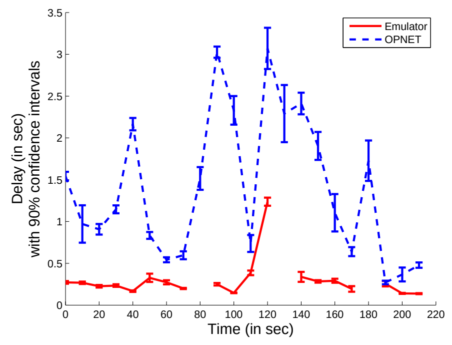
Figure 7: Streaming rate at server and client for 26-node scenarios.

4. AGGREGATE TRAFFIC ANALYSIS

Given the limitations of the emulator, how useful is it in its current form? We try to answer this question by investigating aggregate traffic behavior at larger time scales using the emulator setup. In particular, we check whether we can identify traffic self-similarity in the emulated wireless networks. We verify the reliability of the results by comparing with those obtained from OPNET for similar network and source traffic models.

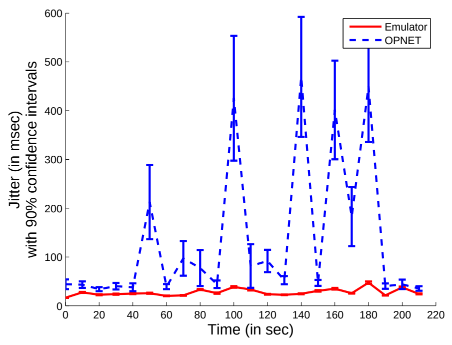


(a) Static scenario

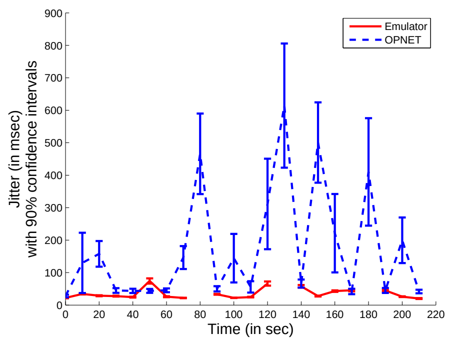


(b) Mobile scenario

Figure 8: Streaming media client delay for 26-node scenarios.



(a) Static scenario



(b) Mobile scenario

Figure 9: Streaming media client jitter for 26-node scenarios.

The importance of traffic self-similarity is that it significantly impacts network performance [13, 17] through the related phenomenon of *long-range dependence (LRD)*. Because of LRD, the Markovian queuing models do not hold and give inaccurate performance results. Furthermore, self-similarity preserves the burstiness of traffic even at large timescales and hence needs to be accounted for in network provisioning and control. The source of self-similarity in many networks has been shown to be the high variations in file sizes and inter-arrival times. However the underlying network protocols also play a significant role [12, 13]. Here we study the existence of self-similarity in wireless networks.

4.1 Self-Similarity

Depending on the choice of source traffic models, the aggregate traffic in the network can show very different characteristics. The popular Markovian models with exponential file sizes and Poisson arrivals have bursty traffic at small time scales, but the burstiness is averaged out at large time scales. However, high variability in the traffic demands can lead to self-similarity, hence preserving the burstiness even at large time scales. In particular, source models with finite mean, but infinite variance (either in the file sizes, or inter-request times) lead to self-similarity in aggregate traffic in many network scenarios. A commonly used model for generating files sizes or inter-request times with high variability is the Pareto distribution.

The self-similar nature of the traffic is characterized by it's *Hurst Exponent (H)*. There are many statistical tests for self-similarity [13, 17]. Two of the commonly used tests - *R/S method* and *Variance-time method* - give a rough estimate of the H. However, the wavelet analysis method of [1] gives a more robust estimate of H. For practical scenarios, a value of H between 0.5 and 1 is obtained and higher H implies burstier traffic.

4.2 Scenario for Studying Self-Similarity

We emulate a 10 node clique scenario with 9 connections. We use our traffic generator tool *Traffic App* to generate UDP traffic with different distributions for file-sizes and inter-request times. The tool can generate UDP packets with constant, exponential or Pareto distribution for sizes or inter-requests. We study the following scenarios and compare the results with OPNET -

1. Exponential file sizes and exponential inter-request times
2. Pareto file sizes and exponential inter-request times
3. Exponential file size and pareto inter-request times
4. Pareto file size and pareto inter-request times

In each of the 4 scenarios, all the 9 connections use identical distributions. The mean packet size is 1500 Bytes and the mean inter-request time is 60 msec. Using Wireshark, we capture all the packets received at the destination and find out the aggregate traffic rate (in bps). The Hurst estimates of the corresponding time-series using the Wavelet method are given in Table 2. We obtain similar Hurst estimates from R/S method and Variance-time method. We can see that heavy-tails in the distribution leads to higher values of H implying stronger self-similarity and long-range dependence. In particular, the value of H for scenario 1 is close to 0.5 implying absence of long-range dependence and absence of burstiness at larger time scales.

Scenario	1	2	3	4
Emulator	0.518	0.572	0.631	0.687
OPNET	0.506	0.625	0.702	0.625

Table 2: Hurst Parameter Estimates

Even though the values of the Hurst estimates vary between the emulator and OPNET, they predict similar behavior in terms of the presence or absence of self-similarity. But a more detailed analysis of the impact of the protocols and network load will need correct modeling of protocol functioning. Whereas correct modeling of many different higher layer protocols in simulators like OPNET will take a lot of time, improving the MAC models in EMANE seems to be easier. Once that is achieved, the emulator setup can be used with real applications for a more accurate study.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we described an emulation setup to study the impact of MAC and routing on the performance of real-time traffic in wireless networks. We presented our work with a new open-source software emulator called EMANE. Our objective was to analyze how reliable are the results of experiments on the emulator, and, therefore, how useful the software emulator will be for high-fidelity wireless performance tests. Comparing our results from the emulator with those from OPNET, we see that they are in good agreement within the differences of certain implemented features.

However, there are significant limitations with the current versions of EMANE, which we have described in the paper. The same holds true for the OLSR code we used. We are trying to fix the issues together with the respective authors. We will implement detailed path loss and channel fading models in EMANE to study the impact of the wireless channel. For the study of routing protocols, we are also working on implementing the *Stable Path Topology Control* (SPTC) [18] modifications to OLSR, which we believe will give better real-time performance in realistic wireless scenario. For traffic self-similarity analysis, we plan to use more realistic traffic models and actual applications like VoIP and VBR video-streaming. The detailed modeling of the wireless channel together with real applications will provide more realistic results than simulations and help study the impact of the wireless setting.

Acknowledgment

The authors would like to thank CenGen Inc. for answering our questions about EMANE and troubleshooting our emulation setup.

Research supported by the US Army Research Office through MURI awards with numbers W911-NF-08-1-0238 and W911-NF-07-1-0287, and by the Defense Advanced Research Projects Agency (DARPA) under award number 013641-001 for the Multi-Scale Systems Center (MuSyC), through the FRCP of SRC and DARPA.

6. REFERENCES

[1] P. Abry and D. Veitch. Wavelet analysis of long-range dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, Jan. 1998.

[2] A. Alvarez, R. Orea, S. Cabrero, X. G. Pa neda, R. García, and D. Melendi. Limitations of network emulation with single-machine and distributed ns-3. In *SIMUTools '10: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[3] 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*, 0:307–314, 2009.

[4] J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. Smith, and M. Weigle. Stochastic models for generating synthetic http source traffic. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004.

[5] CENGEN. Emane - extendable mobile ad-hoc network emulator. http://labs.cengen.com/emane/.

[6] 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.

[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] 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.

[9] NS-3. Network simulator 3. http://www.nsnam.org/.

[10] OLSR. Olsr routing protocol implementation version 0.5.3. http://www.olsr.org.

[11] OPNET-Modeler. http://www.opnet.com/solutions/network_rd/modeler.html.

[12] K. Park. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *In Proc. IEEE International Conference on Network Protocols*, pages 171–180, 1996.

[13] K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation.* John Wiley & Sons, Inc., New York, NY, USA, 2000.

[14] M. Pu̍dar and T. Plagemann. Neman: A network emulator for mobile ad-hoc networks. In *8th International Conference on Telecommunications ConTEL 2005*, 2005.

[15] Pulseaudio. http://www.pulseaudio.org.

[16] H. Rogge, E. Baccelli, and A. Kaplan. Packet sequence number based etx metric for mobile ad hoc networks. IETF Draft, March 2010.

[17] O. Sheluhin, S. Smolskiy, and A. Osin. *Self-Similar Processes in Telecommunications.* John Wiley & Sons, Inc., New York, NY, USA, 2007.

[18] K. Somasundaram, J. Baras, K. Jain, and V. Tabatabaee. Distributed topology control for stable path routing in mobile ad hoc networks. In *Conference on Decision and Control*, 2010.

[19] VideoLAN. http://www.videolan.org.

[20] Wireshark. http://www.wireshark.org.

[21] XEN. http://www.xen.org.