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

(Poster Abstract)

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 provide an open-source software emulator setup for MANETs and study its fidelity in providing reliable performance estimates for real-time traffic. 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. As an instance, we study the performance of real-time streaming media over a mesh network supported by OLSR. We also use the setup to analyze the impact of the wireless network on the self-similarity of the traffic.

Categories and Subject Descriptors

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

General Terms

Experimentation, Performance, Measurement

Keywords

Software Emulation, Performance Evaluation

1. INTRODUCTION

There has been growing interest in both military and commercial spaces to deploy Mobile Ad-hoc Networks (MANETs) for real-time traffic applications. However, their development has been hampered by the lack of reliable performance estimates. Most of the work in estimating the performance has been done using network simulators such as ns-3 and OPNET Modeler, 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, cannot capture the high variability in real-time traffic. Detailed and precise statistics of real-time network statistics can be measured using testbeds, which are prohibitively expensive. Network emulation provides a suitable alternative to both simplistic simulators and expensive testbeds. Among network emulators, there are two different approaches: hardware emulation [3] and software solutions [1, 2], with the later being significantly less expensive than the former. We believe that a high-fidelity software emulation is an attractive solution to obtain reliable performance metrics of MANET protocols.

In this paper we describe the use of an open-source software emulator for real-time traffic analysis. We setup a software emulator on a cluster using EMANE [2] for emulating the IEEE 802.11 MAC/PHY. 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 running Optimized Link State Routing (OLSR), a popular MANET routing protocol. Second, we study burstiness of traffic in wireless networks 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 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.

2. EMULATION SETUP

We use the open-source Extensible Mobile Ad-hoc Network Emulator (EMANE) [2] as the framework for wireless network emulation. We use EMANE because: 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 MAC and PHY layer architecture; and it can be extended with custom-built physical and link layer models.

EMANE provides a set of APIs to allow independent development of network emulation modules (NEMs), emula-

*Research supported by awards MURI W911-NF-08-1-0238 and CTA DAAD19-01-2-0011 from the US Army Research Office, and by the award CNS1018346 from the NSF.

SIMUTOOLS 2011, March 21-25, Barcelona, Spain

DOI 10.4108/icst.simutools.2011.245554

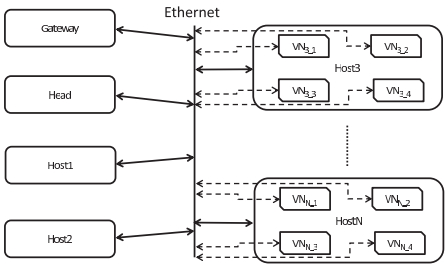

Figure 1: Virtual Nodes in the Cluster

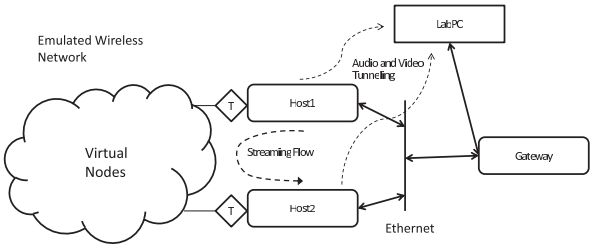

Figure 2: Audio and Video Tunneling over Ethernet and Wireless Network over Virtual Interface

tion/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. 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, which model the wireless network. EMANE has a well-developed IEEE 802.11 a/b/g model. 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 and fading. However, the current release of EMANE (release 0.6.4) is a beta release. While it has most of the important features, there are certain limitations. 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. In addition, 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).

For setting up the emulator, 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 1. The physical nodes form a LAN over ethernet, and are accessible via a *Gateway* node, as shown in the figure. The rest of the nodes are labeled *Head, Host1, Host2,..,HostN*. Since we use want to use the physical cluster for emulation of a large wireless network, we use the Xen hypervisor 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. Each node runs the OLSR protocol on its virtual interface *emane0* using the open-source olsrd-0.5.5 [4] implementation. However, for the OLSR codes released so far (upto release 0.6.0), the topology control mechanism does not work efficiently; instead of optimizing the link state information flooded in the network, all the information is flooded. 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. 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 use VLC player 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 sound server 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. 2.

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, *Traffic App*. We use *Traffic App* to generate Constant Bit Rate (CBR) UDP traffic. For the MAC and PHY layers, we use 802.11b at 11 Mbps, and use RTS-CTS mechanism for sending data packets. For each scenario, we capture packets using Wireshark and analyze the carried load, delay and jitter using Matlab scripts.

We study a 26-node network with a mobile VLC client. 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. A mesh network of backbone static wireless nodes consisting of 25 nodes is deployed as a grid network, i.e., in a 5×5 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.

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 3 and 4 respectively show the bitrate and jitter between the streaming server-client. The plot for delay is similar to 4 and is omitted in the interest of space. We observe long periods of disconnectedness due to link changes and route-detection delays of OLSR. 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 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. We believe that the differences in delay and jitter are because of the differences in modeling of collisions in EMANE and OPNET.

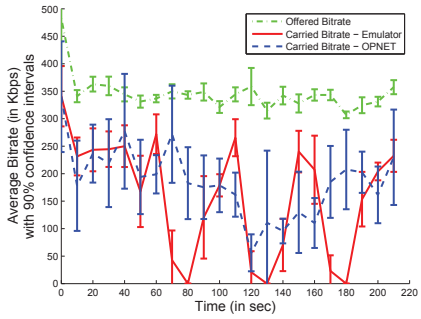

Figure 3: Streaming rate at server and client for 26-node scenarios.

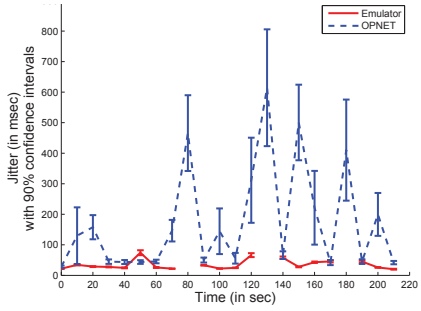

Figure 4: Jitter of streaming media at the client for 26-node scenario.

4. TRAFFIC SELF-SIMILARITY

As another usage of the setup, we investigate the aggregate traffic behavior at larger time scales. In particular, we try to identify traffic self-similarity in the emulated wireless networks. We verify the reliability of the results by comparing with those obtained from OPNET. The importance of traffic self-similarity is that it impacts network performance [5] by preserving traffic burstiness even at large timescales. Also, because of the related phenomenon of *long-range dependence (LRD)*, the Markovian queuing models do give inaccurate performance estimates. 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. Here we study the existence of self-similarity in wireless networks. Self-similarity of network traffic is characterized by the traffic's *Hurst Exponent (H)*. We use the wavelet analysis method [5] to get a robust estimate of H.

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

Table 1: Hurst Parameter Estimates

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. We study the following scenarios and compare them 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, and (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 1. 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. 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. REFERENCES

[1] 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 *Proceedings of SIMUTools*, 2010.

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

[3] 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.

[4] OLSRD. Olsr routing daemon. http://www.olsr.org.

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