An Evaluation of IPv6 in Simulation using OPNET Modeler

Brittany Clore*†, Matthew Dunlop*†, Randolph Marchany†, Joseph Tront*
*Bradley Department of Electrical and Computer Engineering
†Virginia Tech Information Technology Security Office
Virginia Tech
Blacksburg, Virginia 24061, USA
e-mail: {clore, dunlop, marchany, jgtront}@vt.edu

Abstract— **Simulation is vital to be able to test various network topologies and new components in a cost effective manner. With the push to adopt Internet Protocol version 6 (IPv6), many network administrators need to be able to test their hardware and specialized applications before deploying them on a live network. OPNET Modeler provides the capability to simulate an IPv6 network and the OPNET System in the Loop, an add-on module, allows for real devices to be tested over the simulated network. This study evaluates the support of IPv6 in OPNET Modeler 16.1 with the System in the Loop module. The results show that this module does not fully support IPv6 at this time but with improvements can be an important part to planning and implementing IPv6 networks.**

Keywords - Simulation; IPv6; System in the Loop; OPNET

I. INTRODUCTION

With the assignment of the last block of IPv4 addresses in February 2011, IPv6 is being pushed to rapidly be deployed in new networks. These networks can have a wide range of devices connected into it including specialized software. Before committing to implement a full scale IPv6 production network, simulation of the environment allows network administrators to analyze how their configuration will function. There is a wide range of simulation tools available that can achieve this goal. OPNET Modeler is a commercial solution that provides a wide range of simulated network devices from workstations to switches and routers. While users primarily interact within the graphical user interface, the software is expandable with user written code. The code is C based, with OPNET providing its own classes and functions [1].

While simulating a basic network is vital to examine for IPv6 readiness, many software and hardware vendors are adapting their technologies to support IPv6. There is a need to be able to test these products on an IPv6 network. Simulation is a cost effective way to conduct testing due the capability to simulate various network topologies, sizes, and conditions [2]. The problem is that there are very few network modeling tools that are IPv6 capable. The main simulators that claim to be IPv6 capable are NS3 [3], OMNeT++ [4], and OPNET [5]. Of these, OPNET possesses the best capability to tie in live systems to a simulation environment. For that reason, OPNET was selected as the network simulator to test IPv6 research in the Virginia Tech Information Technology Security Office. OPNET's System in the Loop module is a way to test actual products on an IPv6 network without having to convert the code into simulation. This module was the focus of the study to test the extent of the IPv6 support.

This remainder of this paper is organized as follows: Section II describes other work related to simulating IPv6 in OPNET. Section III provides some background on the OPNET System in the Loop module as well as IPv6 in general. Section IV discusses the set up considerations that are needed for proper functionality. Section V describes the design of the study while Section VI demonstrates the results. The paper is concluded in Section VII along with a discussion of some future work.

II. RELATED WORK

OPNET Modeler is a widely used simulation program that advertises IPv6 support. There have been various studies that have assessed applications in IPv6 within a fully simulated network. One study by Aziz et al. looked at the performance of video and voice traffic in IPv6 [6]. The authors used OPNET Modeler to run simulations in IPv4 and IPv6 to compare throughput and were able to show that IPv6 slightly decreases throughput due to its packet overhead. Le et al. [7] assessed the Mobile IPv6 model for IPv6 header support and routing. They found that the model was able to correctly handle IPv6. These both show that Modeler is capable of simulating an IPv6 network successfully.

Green et al. [8] characterized a test bed for IPv6 applications. Their setup was a simulated network communicating between one real device using a "hardware in the loop" scheme which is very similar to System in the Loop. The difference is that System in the Loop can test a single software piece without specialized hardware. One of their observations was that OPNET does not provide the capability to do a one-to-one match with real packet data to simulated packet data but this could be added with additional code. Their scheme did not function for real time traffic but rather worked for a single non-real time stream of traffic.

ISBN: 978-1-61208-199-1

III. Background

OPNET Modeler is a tool that allows for a wide range of simulation. To extend the simulation, modules can be added on that add extra features. One module is the System in the Loop module. It is also important to understand IPv6 to identify what options need to be implemented to verify that this module has IPv6 support.

A. OPNET System in the Loop

OPNET's System in the Loop allows for communication between real, physical devices or software and a simulated network. It does this by using a specialized node that listens on a given network interface and filters incoming packets (real to simulated network) using the Berkeley Packet Filter syntax. Once it receives the packets, a translation function converts the packet headers and payload into the simulation packet format. The module currently supports the translation of the following protocols: IPv4, IPv6, ICMP, ICMPv6, OSPF, RIPv1, RIPv2, TCP, UDP, and FTP [1].

There are three configurations in which the simulation communication can be set up: physical device to simulated device, simulated device to simulated device through a real device and real device to real device through a simulated device. The assessment was done for the third type of communication setup, real device to real device through simulation. This configuration provided the capability to evaluate packet behavior as real packets enter the simulation environment and again as the same packets are translated back into real packets for delivery to a live destination.

B. IPv6 Background

IPv6 differs significantly from Internet Protocol version 4 (IPv4). The most noticeable difference is that IPv6 uses a 128 bit addressing space while IPv4 uses a 32 bit addressing space [9]. Within the OPNET simulation code this translates into using a pointer versus using a defined type such as a double. To provide an idea of the scope of the IPv6 address size, the entire IPv4 address space fits into a single IPv6 subnet over four billion times.

Another difference is the packet header. Where IPv4 headers were of variable length due to the possible inclusion of options, IPv6 headers are a fixed 40 bytes [9]. Options are includes as extension headers and become part of the payload. Extension headers do not have a specified order and contain a next header field that acts like a chain within the header to connect all the extension headers. Extension headers are used to specify what protocol is next being used in the packet as well as other functions like fragmentation and Internet Protocol Security (IPSec) options. An extension header also exists where users can define their own functionality. This type of extension header is referred to a Destination Options header. Destination options contain information that only pertains to the final intended recipient. This flexibility poses a problem in the packet translation to simulation.

Another significant difference is how IPv6 accomplishes address resolution. Due to the large address size, hosts in IPv6 generate their own addresses. This reduces the management burden placed on network managers. Hosts use a process called Stateless Address Auto configuration (SLAAC) to generate addresses. SLAAC addresses are advertised to other network hosts using the Neighbor Discovery Protocol (NDP), which replaces the address resolution protocol (ARP) used in IPv4. NDP uses a series of Internet Control Messaging Protocol version 6 (ICMPv6) messages to advertise addresses as well as solicit for router and other hosts. NDP removes the need to perform certain tasks like specifying a gateway, as this is accomplished by router solicitations and advertisements by the protocol. In addition to NDP messages, ICMPv6 includes other error message types also used by ICMP in IPv4.

IV. Set up

There are specific configuration details that are required for OPNET's System in the Loop module to operate properly. For example, it is important to signify the right interface by including the source Ethernet media access control (MAC) address in the packet filter. Other filters for protocol can be used to further limit the traffic that the module translates into simulation.

The simplest interface configuration is to have one physical network interface per real device. This is not always feasible and so it is possible to have one interface handling the traffic of all real devices; however, the packet filter has to be very specific otherwise traffic can be sent through the wrong section of the simulated network.

Within the simulation environment, a System in the Loop node can only be connected to another node through Ethernet. This connection is defined as a duplex 10Gbps link. Half-duplex links are not allowed. Also within the simulation environment, the System in the Loop node has to define the translation function it's using as well. For this study, the default translation function was being assessed.

V. Design

In its current form, OPNET's System in the Loop supports a small set of protocols. The purpose of this study was to assess the support of IPv6. The main goal was to achieve communication through the simulated network with the intent to measure various IPv6 applications performances.

The design for this study was an isolated network in which two physical nodes were connected by a simulated set of routers. The physical nodes were virtual machines that were hosted on the same machine that runs OPNET Modeler. The virtual machines' network interfaces were bridged with two separate network interface cards that were installed on the host machine. These cards solely handle traffic to and from the virtual machines. Fig. 1 depicts the layout of the virtual machines and simulation. The encompassing box represents the host machine.

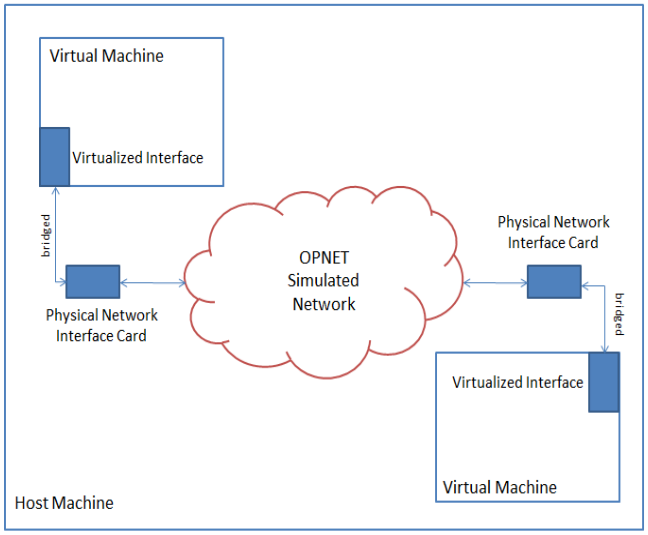


Figure 1. Layout of Virtual Machines and OPNET Simulated Network on the Host Machine

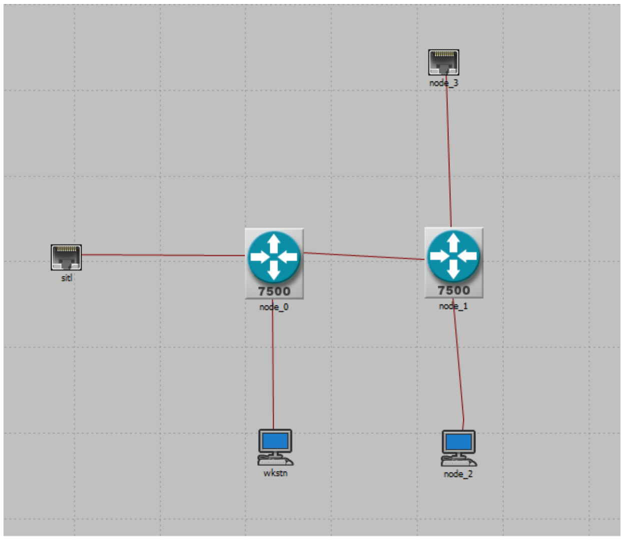


Figure 2. OPNET Simulated Network

The System in the Loop module listens on these network interfaces. The packet filter is defined to filter for IPv6 traffic coming from the MAC address of each virtual machine. The simulated network contains two routers and two workstations. Fig. 2 shows the topology of the simulated network. The workstation nodes are generic nodes defined by OPNET. The routers are simulated Cisco 7507 devices. The simulated workstations provided the option to test communication within simulation. The System in the Loop nodes are located at the right and to the top left. The icon for that node is an Ethernet port. To communicate, these nodes would have to make two hops. Both virtual machines run the Ubuntu 11.4 operating system which supports IPv6 networking.

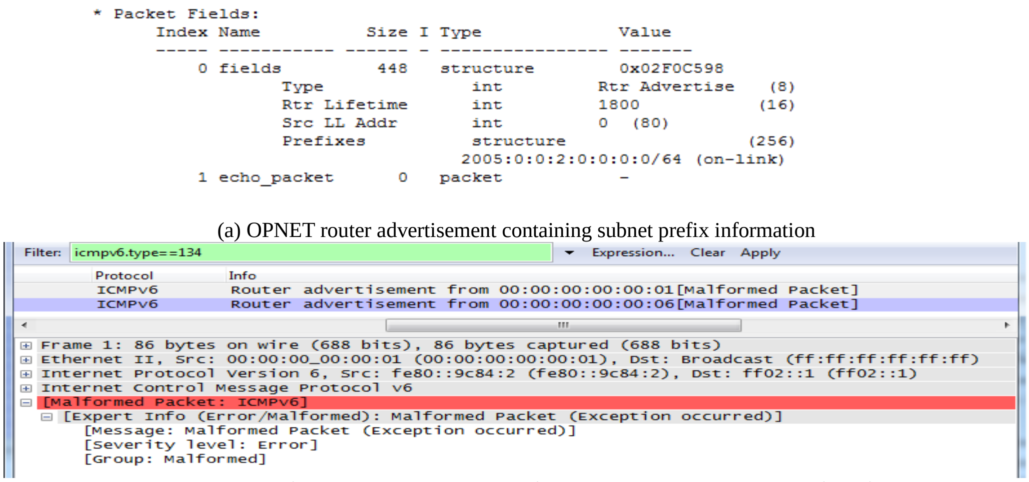
Two categories of tests were run on the simulation. The first set was to achieve communication between the physical nodes with a set of ping messages. This tested connectionless ICMPv6 ping message support and Neighbor Discovery Protocol support. A standard 1-second ping was used as well as executing a 10,000 packet ping flood. The goal of the second set of tests was to test connection-oriented transmission control protocol (TCP) and hypertext transfer protocol (HTTP) communication using IPv6 addressing. This was achieved through accessing a webpage being hosted on one of the virtual machines and doing a series of files transfers using wget, a free software package that allows files transfer through the HTTP protocol. The file sizes transferred ranged from 1-kilobyte files to 1-gigabyte files. These tests were executed on the live nodes through the command line.

VI. RESULTS

OPNET's System in the Loop proved to not fully support IPv6. There are two main issues that were found preventing this software module from being fully able to simulate real IPv6 communication. One issue related to proper support of NDP while the other issue was caused by lack of support for some ICMPv6 message types.

The first issue was caused by the inability to properly process NDP router advertisement messages. In IPv6, hosts rely on router advertisements sent by local routers to auto-configure addresses and to learn of possible gateways. Without these router advertisements, hosts cannot learn what subnet they are connected to nor which router is their closest gateway. OPNET is sending out router advertisements, but they are malformed. As a result, real systems connected to OPNET still need to statically set addresses and gateways. Fig. 3(a) shows the OPNET format for the router advertisement. The router advertisement contains a subnet prefix value and length. OPNET is unable to properly translate the packet. Fig 3(b) shows the corresponding packet in Wireshark as being malformed. In OPNET, there is a way to manually set the prefix, but this is also not translated in a correct manner. It is not clear whether the router advertisement is malformed within the simulation and the simulated nodes understand the bad packet or if the packet becomes malformed due to processing by OPNET's System in the Loop module.

ISBN: 978-1-61208-199-1



```
* Packet Fields:
     Index Name          Size I Type               Value
     ----- ----------- ------ - ----------------- -------
         0 fields        448   structure          0x02F0C598
                 Type              int          Rtr Advertise    (8)
                 Rtr Lifetime      int          1800            (16)
                 Src LL Addr       int          0   (80)
                 Prefixes          structure                   (256)
                                  2005:0:0:2:0:0:0:0/64 (on-link)
         1 echo_packet      0   packet             -
```

(a) OPNET router advertisement containing subnet prefix information

```
Protocol      Info
ICMPv6        Router advertisement from 00:00:00:00:00:01[Malformed Packet]
ICMPv6        Router advertisement from 00:00:00:00:00:06[Malformed Packet]

⊞ Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
⊞ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 6, Src: fe80::9c84:2 (fe80::9c84:2), Dst: ff02::1 (ff02::1)
⊞ Internet Control Message Protocol v6
⊟ [Malformed Packet: ICMPv6]
  ⊟ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
      [Message: Malformed Packet (Exception occurred)]
      [Severity level: Error]
      [Group: Malformed]
```

(b) Wireshark capture of the resulting translated malformed packet with no subnet prefix information

Figure 3. Comparison of OPNET router advertisement with the translated packet in Wireshark

The second issue is that OPNET only accepts a small set of ICMPv6 message types and does so improperly. When first trying to communicate with ping messages (the first category of tests) between physical machines, the simulation would fail because it did not recognize the ICMPv6 message type. However, ping request and response message types are in the OPNET supported set. Further investigation in the code revealed that the top bytes were getting improperly set. This error could be caused by two things. One cause is the use of an improper type to handle the information. The second cause is that the translation function from real to simulated packet adds erroneous data. The fix for this issue was to mask the lower two bytes of the message type field in the simulation code for the IPv6 Neighbor Discovery process node. The specific code for this is:

```
icmp_pk_fields_ptr->message_type & 0xff.
```

The resulting value is a correct message type. Fig. 5 shows the simulation error message before making this fix.

Two smaller concerns are that the current translation functions do not support many IPv6 extension headers or any IPv6 routing protocols. These issues were discovered in the second category of tests. These concerns were known before doing the assessment from training classes provided by OPNET [10]. For full IPv6 support, extension headers are must because of many features inherent to IPv6. For example, IPv6 includes native support of IPSec. IPSec is implemented through extension headers, which are currently not supported by OPNET. Further investigation was done using Scapy [11], a packet manipulation tool. Packets using each extension header were created and sent through simulation. Fig. 6 shows an example of the hop by hop extension header packet being sent and OPNET's log message saying it is unsupported. It was found that no extension headers defined by RFC 2460 [9] were supported.

After fixing the message type and statically setting a gateway on the network interfaces, the tests did execute. Fig. 4 shows a graph of the IPv6 traffic received by the simulated router during the second category of tests. It clearly shows traffic is being translated into simulation. When the wget transfer finishes after the eight minute mark, the traffic received drops as expected. Both categories of tests produced similar graphs and results.

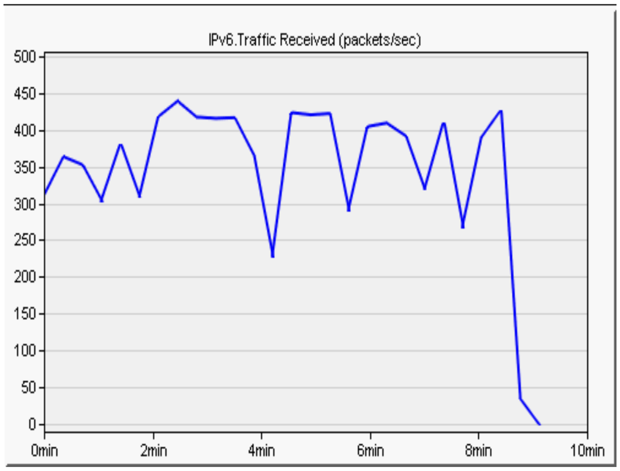


Figure 4. IPv6 traffic received by the simulated router

VII. Conclusion and Future Work

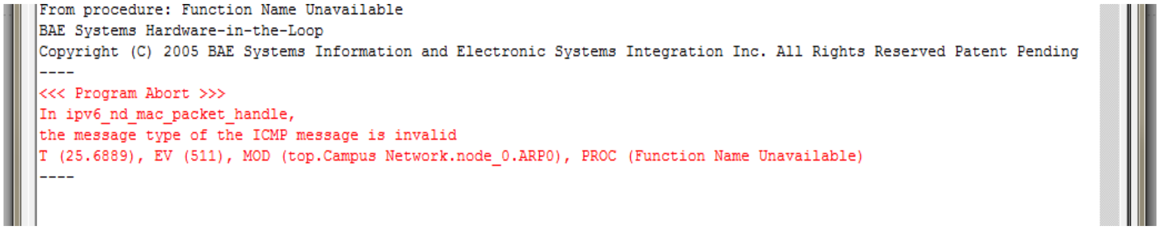
Out of the box, OPNET's System in the Loop is not yet ready to handle IPv6 simulation. Errors in handling essential fields of packets make it difficult to get the full potential out of the product. Complications that arise during the set up of network interface cards and virtual machines with System in the Loop can hinder the simulation and add an extra step to analyzing data.

Future work includes writing a new translation function for IPv6 packets that includes extension header support.

ISBN: 978-1-61208-199-1

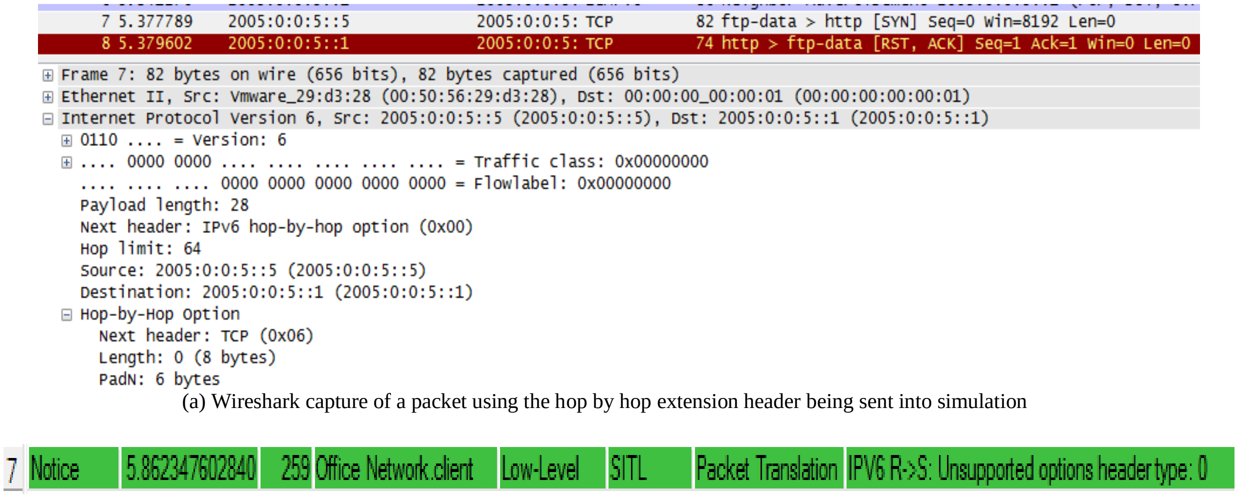
Further investigation into the malformed router advertisements needs to be done to see if a new translation function would solve this error. Using a different configuration to see the effect on translation delays is also planned.

With a translation function that supports the flexibility of IPv6 with its extension headers and fixes in the ICMPv6 message type handling, System in the Loop can be a viable simulation tool for network administrators.



```
From procedure: Function Name Unavailable
BAE Systems Hardware-in-the-Loop
Copyright (C) 2005 BAE Systems Information and Electronic Systems Integration Inc. All Rights Reserved Patent Pending
----
<<< Program Abort >>>
In ipv6_nd_mac_packet_handle,
the message type of the ICMP message is invalid
T (25.6889), EV (511), MOD (top.Campus Network.node_0.ARP0), PROC (Function Name Unavailable)
----
```

Figure 5. ICMP Message Type Error



```
  7 5.377789   2005:0:0:5::5          2005:0:0:5: TCP      82 ftp-data > http [SYN] Seq=0 Win=8192 Len=0
  8 5.379602   2005:0:0:5::1          2005:0:0:5: TCP      74 http > ftp-data [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

⊞ Frame 7: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
⊞ Ethernet II, Src: Vmware_29:d3:28 (00:50:56:29:d3:28), Dst: 00:00:00_00:00:01 (00:00:00:00:00:01)
⊟ Internet Protocol Version 6, Src: 2005:0:0:5::5 (2005:0:0:5::5), Dst: 2005:0:0:5::1 (2005:0:0:5::1)
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 28
    Next header: IPv6 hop-by-hop option (0x00)
    Hop limit: 64
    Source: 2005:0:0:5::5 (2005:0:0:5::5)
    Destination: 2005:0:0:5::1 (2005:0:0:5::1)
  ⊟ Hop-by-Hop Option
      Next header: TCP (0x06)
      Length: 0 (8 bytes)
      PadN: 6 bytes
```

(a) Wireshark capture of a packet using the hop by hop extension header being sent into simulation

```
7 | Notice | 5.862347602840 | 259 | Office Network.client | Low-Level | SITL | Packet Translation | IPV6 R->S: Unsupported options header type: 0
```

(b) OPNET log output of unsupported packet

Figure 6. Attempt to send a packet using the hop by hop extension header into OPNET from a live machine

REFERENCES

[1] "OPNET Modeler" [Online] Available: http://www.opnet.com/solutions/network_rd/modeler.html, Accessed on 23 January 2012.

[2] Kaplan, G.; , "Simulating networks," Spectrum, IEEE , vol.38, no.1, pp.74-76, Jan 2001 doi: 10.1109/6.901148 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=901148&isnumber=19336G.

[3] "ns-3: IPv6 Class Reference" [Online] Available: http://www.nsnam.org/doxygen/classns3_1_1_ipv6.html, Accessed on 25 January 2012.

[4] "OMNeT++ IPv6 Suite" [Online] Available: http://www.omnetpp.org/omnetpp/doc_details/2137-ipv6suite, Accessed on 25 January 2012.

[5] "OPNET: IPv6 for R&D Specialized Model" [Online] Available: http://www.opnet.com/solutions/network_rd/simulation_model_library/ipv6.html, Accessed on 12 December, 2012.

[6] M. Aziz, M. Islam, and M. Khan, "Throughput Performance Evaluation of Video/Voice Traffic in IPv4/IPv6 Network," in Internation Journal of Computer Applications, vol. 35 no. 2, pp. 5-12, December 2011.

[7] D. Le, X. Fu, and D. Hogrefe, "Evaluation of Mobile IPv6 Based on an OPNET Model," unpublished.

[8] Green, D.; Mayo, R.; Ranga Reddy; , "IPv6 Application Performance Characterization Using a Virtual/Live Testbed," Military Communications Conference, 2006. MILCOM 2006. IEEE , vol., no., pp. 1-4, 23-25 Oct. 2006 doi: 10.1109/MILCOM.2006.302398 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4086641&isnumber=4043248

[9] S. Deering and R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, IETF RFC 2460, December 1988; http://www.ietf.org/rfc/rfc2460.txt

[10] "OPNET Training" [Online] Available: https://www.opnet.com/training/index.html, Accessed on 23 January 2012.

[11] "Scapy" [Online] Available: http://trac.secdev.org/scapy, Accessed on 1 February 2012.