SANDIA REPORT
SAND2009-5996
Unlimited Release
Printed September and 2009

Simulated, Emulated, and Physical Investigative Analysis (SEPIA) of Networked Systems

David P. Burton, Michael J. McDonald, Uzoma A. Onunkwo,
Thomas D. Tarman, Vincent E. Urias, Brian P. Van Leeuwen

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2009-5996
Unlimited Release
Printed September 2009

Simulated, Emulated, and Physical Investigative Analysis (SEPIA) Of Networked Systems

David P. Burton, Michael J. McDonald, Uzoma A. Onunkwo,
Thomas D. Tarman, Vincent E. Urias, Brian P. Van Leeuwen

Sandia National Laboratories
PO Box 5800
Albuquerque, New Mexico 87185

Abstract

This report describes recent progress made in developing and utilizing hybrid Simulated, Emulated, and Physical Investigative Analysis (SEPIA) environments. Many organizations require advanced tools to analyze their information system's security, reliability, and resilience against cyber attack. Today's security analysis utilize real systems such as computers, network routers and other network equipment, computer emulations (e.g., virtual machines) and simulation models separately to analyze interplay between threats and safeguards. In contrast, this work developed new methods to combine these three approaches to provide integrated hybrid SEPIA environments. Our SEPIA environments enable an analyst to rapidly configure hybrid environments to pass network traffic and perform, from the outside, like real networks. This provides higher fidelity representations of key network nodes while still leveraging the scalability and cost advantages of simulation tools. The result is to rapidly produce large yet relatively low-cost multi-fidelity SEPIA networks of computers and routers that let analysts quickly investigate threats and test protection approaches.

ACKNOWLEDGMENTS

The authors would like to acknowledge that the work, which produced the results presented in this paper, was funded by the U.S. Department of Energy under Sandia's Laboratory-Directed Research and Development (LDRD) Program.

We also acknowledge the contributions of the following individuals, who influenced the direction and provided focus for this research:

LeAnn Miller (Project Manager)

Chuck Villamarin

Bernie Clifford

Michael Johnson

Bill Cook

Tan Thai

David Harris

Peter Sholander

Nancy Durgin

Vic Romanelli

Ben McBride

John Mulder

Andrew Othling

We are also grateful for the laboratory and experimental support provided by the following individuals:

Jayme Lara

Marshall Daniels

Table of Contents

Table of Figures

1. Introduction

One need not look far to develop an understanding that computer network security has become an important topic world-wide. Most people have awakened to the fact that adversaries can move their virtual presence, through software across networks like the Internet and into individual computers. Not fully appreciated is that sophisticated adversaries can even utilize software to reconfigure the network fabric itself. Once reconfigured, adversaries can potentially produce wide-spread denials of service.

Fortunately or not, the techniques that adversaries can use to cause this sophisticated mischief are often complex and arcane. The fortunate aspect is that this requisite sophistication keeps the techniques away from most adversaries. The unfortunate aspect is that it also keeps their discovery and their ability to develop appropriate mitigations away from most security researchers. Worse is the fact that many of these vulnerabilities must be studied at scale. For adversaries willing to break the law, the Internet itself has become a favored testbed. For security researchers wishing to do no harm, the number of choices for larger scale vulnerability studies is limited. This research is pointed at removing the limits and giving those security researchers new ways to discover, study, and address these vulnerabilities.

To review, the most widely-used technique that network researchers use for deep analysis is hardware-based. Here, researchers purchase and configure networks from physical equipment that they have purchased or built. They then instrument the networks using traditional network diagnostic equipment and connect computers to the networks to generate appropriate traffic. While very accurate, this approach is problematic for two reasons. First, the equipment can be very expensive to acquire, configure, and maintain. Second, instrumentation and experimentation can be problematic. It is difficult to correlate traffic events that move across the network and, as a result, difficult to roll up studies and generate system-level information.

Network researchers also use simulation extensively. There are numerous simulation tools in existence today for studying network issues. In comparison to about a decade ago, these current tools have sophisticated capability and increased accuracy. They make it very easy to correlate events across the networks and generate system-level information. Unfortunately, few have reached a new realm of reality in network experimentation that allows researchers to effectively evaluate various implementations and especially study threats and vulnerabilities at scale.

As for example, network simulation tools such as OPNET are designed in part to allow engineers and researchers to understand how network algorithms perform at scale. Analysts can implement and deploy these algorithms on networks of simulated devices, trace messages that the devices send between one another, and collect statistics on the resultant traffic and other delays.

Until recently, these simulation models were divorced from the implementations. At the implementation phase, engineers re-code the simulation models into the various deployment languages and then test the implementations on physical networks of these devices. In practice, the simulation and implementation codes are invariably different. For example, implementation codes often get refined and features get added without being simulated and hence the simulation models and implementations branch in capability. As a result, it is difficult to get accurate predictions from the models alone. In the case of vulnerability analysis, this difference limits the number of vulnerabilities that researchers might discover through the simulation models alone. As a result, the vulnerability researchers traditionally turn to the implementations for their analysis with the cost of limiting the size and diversity of the networks that they can analyze.

The dynamics of this problem-solving environment are changing. Most importantly, a few network simulation model vendors have begun to offer so-called System in the Loop (SITL) interfaces that allow researchers to pass real network traffic into their simulated networks [OPNET]. Additionally, improvements in emulation and virtualization have made it possible for researchers to build relatively large networks of emulated or virtualized devices on a small number of computer platforms [CISCO-SIM, CHETNET-EMULATORS, DLINK-EMULATORS]. Additionally, researchers have made a variety of advances in representing the application portions of the networked systems. These advances range from automation tools to represent individual users on corporate and other networks [ROSSEY-02] to simulation systems to represent virtual control system environments [MCDONALD-08].

This research embraced and built upon these advances to develop what we term advanced Simulated, Emulated and Physical Investigative Analysis (SEPIA) environments. SEPIA tools allow analysts to *rapidly* and *cost-effectively* analyze complex network security issues. The research made the following key advances:

1. It extended upon OPNET's SITL tools for allowing real traffic to pass through the simulated networks, by developing new techniques that allow complex real and emulated systems to interoperate with their simulated counterparts.
2. It extended upon existing emulators by developing hypervisors that allows researchers to launch and manage connected networks of emulated network devices from a single application.
3. It executed a series advanced proof of principal experiments that, in addition to showing SEPIA's value, provided the researchers with new insight into vulnerabilities associated with network configurations and protocols, such as the Internet's Border Gateway Protocol (BGP)
4. It developed a new understanding of how the simulations models within these SEPIA environments will scale.
5. It developed tools to automatically configure SEPIA testbeds for rapid implementation.

Key aspects of this research have been published [VAN LEEUWEN-2008], [VAN LEEUWEN-2009]. The remainder of this paper describes these advances in more detail. In addition, it provides a background on the BGP algorithm, which was an essential

technical issue in the work. Finally, it describes experiments and performance tests we performed to validate the approach and test its limits.

2. Background

Simulated, Emulated and Real

For the sake of clarification, we define briefly the terms simulated nodes, emulated nodes, and real nodes. Here, s*imulated* refers to the nodes represented through simulation tools. These nodes generally use stripped down or abstracted implementations of the software running on virtualized hardware and hence easier to setup for typical cases. *Emulated* nodes on the other hand use real software, for instance Cisco IOS (originally Internetwork Operating System), but run on emulated or virtualized machines. Real nodes are obvious – the real software running on real hardware.

It is noteworthy that this report uses the words Emulation and Virtualization synonymously. The technical difference between emulation and virtualization is that while emulation uses machine-code translation to implement "machine within a machine" functionality, virtualization simply uses containment and indirection to implement the same apparent result. As this implementation difference is not significant at the system scale the terms are often blurred. Examples of emulated or virtualized systems include VMWare, QEMU and Dynamips [CISCO-SIM]).

Simulated vs. emulated vs. real nodes and experiments have varying degrees of difficulty and cost involved in procurement, configuration and deployment. Simulation node and experiment configuration typically takes the shortest time when compared to the time-intensive and often high monetary expense involved in real-hardware configuration. However, simulation models typically do not implement all functions and incorporate abstractions of the device being modeled. Models used in simulations must also be configured to represent the devices being modeled as closely as possible. The process of obtaining configuration parameters of implemented network devices and transcribing the equivalence into simulation models can be difficult, tedious, and time-consuming.

Many of today's network critical analysis involve the use of custom-made testbeds from real hardware components. These are typically an expensive and time-consuming to deploy and use, but nevertheless required for critical missions. In some of these scenarios, a number of simulation runs are performed before the real network is built. However, the ability to rapidly test prototype network devices is still a major challenge. In many cases, the simulation program code needs to be developed to simulate the devices in question. These codes, sometimes buggy, typically do not depict an accurate picture and feedback from simulations is used to re-work the simulation code. This process is time consuming and inefficient to the extent that deployed network setups are not well tested.

SEPIA attempts to bring together the best of these approaches. Analysis questions like "What happens if we change a component or subnet of an already implemented network?" can be answered without modeling the existing network when SEPIA is used. In this case the implemented network is interfaced with the modeled or emulated devices or subnets and experiments can be performed with high-fidelity results and without replicating the existing network in a simulation. Studies of this type are invaluable for a rapid network testing and deployment. The emerging SEPIA technique thus has the advantage of inserting prototype devices into well-tested simulated networks.

The Border Gateway Protocol (BGP)

Border Gateway Protocol (BGP) is the widely used routing protocol for internet and VPN networks today. The study of BGP has attracted great interest over the years because of its ubiquitous nature for internetworking [FEMSTER-04], [FEMSTER-07]. In a network, correct configuration of the BGP protocol at the router(s) is essential to allow for intended functioning of the network as well as providing access to external networks. Improper configuration of BGP has resulted in unintended network behavior, a recent example being the Pakistani incident unintentionally denying access to YouTube's website [BROWN-08].

BGP comes in two forms – external BGP (eBGP) and internal BGP (iBGP). The focus of this report's work is with eBGP, which defines the policy of network access between different Autonomous Systems (ASes) as opposed to iBGP that typically reside for an AS and can be implemented using the common Open Shortest Path First (OSPF) routing protocol. The BGP protocol implementation can be captured using a finite state machine diagram as shown in Figure 1.

In BGP terminology, any two routers that interface with each other using the BGP protocol are known as BGP peers. In every peer-to-peer session, each router maintains a variable that indicates which of the states in Fig. 1 they are currently in. During initialization, every BGP router goes into the first state, the IDLE state. This is the state responsible for initiating Transmission Control Protocol (TCP) connection with peers and setting up resources for the BGP connection. Upon successful initialization, the peers go into the CONNECT mode, where the underlying TCP connection completes. If this state completes successfully, the state variable changes to the OPEN_SENT mode. If unsuccessful at the CONNECT state, the peer goes into ACTIVE mode, where the peer is reset and the process of establishing a connection is repeated. From the OPEN_SENT state, the BGP peer goes into OPEN_CONFIRM where the peer waits for successful delivery of KeepAlive messages before entering the final connected state of ESTABLISHED. There are more details on how the BGP transitions and uses these states to recover from failed paths and make choice on routing paths, but these are beyond the scope of this work. In synopsis, each BGP router maintains a state variable that indicates its current status with respect to the six modes in Fig. 1, and its next

course of action (depicted by the directed edges of the finite state machine) from its received messages.

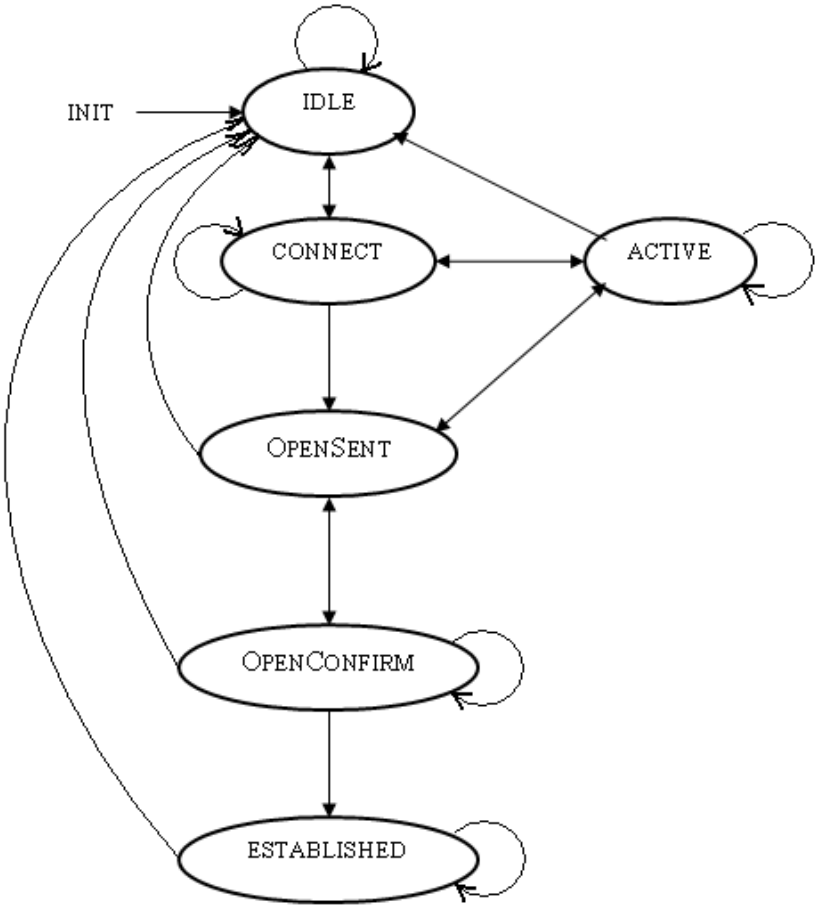


Figure 1: BGP finite state machine model

3. Emulation

A full featured router virtual machine was identified as a necessary SEPIA capability. Important features included capable of booting a vendor's binary, the ability to forward packets and connect-ability to other router virtual machines running both locally and remote machines. Though other router virtual machines, such as NISTnet [DAWSON-2000], existed that had many of these features, Dynamips [CISCO-SIM] was identified as the only router virtual machine capable of booting a vendor's binary. The first versions of Dynamips virtualized the Cisco 7200 platform and booted IOS Version 12.2(25) as shown in Figure 2.

Manually configuring each virtual appliance for large experiments is time consuming and prone to error. Thus, a hypervisor was developed that allows an end user to specify virtual machine configurations, deploy them onto an emulation host node and connect the virtual machines into a desired network topology. The hypervisor, called VirtualNet, was written in the Python scripting language. VirtualNet recieves an eXtensible Markup Language (XML) file which specifies the network topology and configuration information. VirtualNet then launches the corresponding virtual network appliances. The use of XML configuration files allows for easily restarting of experiments, sharing of experiments between different users and the creation of experiments for users less experienced with VirtualNet. Once the virtual appliances have been instantiated, VirtualNet allows the user to dynamically interact with them. This dynamic interaction is useful for empirically investigating network behavior. VirtualNet is designed to be modular such that support for new virtual appliances can be added or functionality for existing virtual appliances can be extended easily. For instance, adding Simple Network Management Protocol (SNMP) support to the base code required only minor changes. VirtualNet was similarly extended to receive an XML file exported from OPNET (see the Experiment Creation Tools section for more details).

Because Dynamips can emulate the behavior of a Cisco router it has served as the primary router virtual machine. Yet Dynamips' packet forwarding performance is severely limited. Vyatta [www.Vyatta.com] produces a Linux based router that boast performance that competes with hardware routers. A virtual router appliance based on Vyatta affords significantly better performance but at the cost of less vendor specific behavior. There is a recent trend in industry of virtualizing network appliances. For instance, Cisco has partnered with VMWare and released a virtual switch [CISCO-NEXUS]. Likewise, Vyatta has partnered with Citrix for deployment on Amazon's web services [NEUMANN -09]. Thus, virtualized network appliance that were once thought to only serve as testbed devices are emerging in production environments.

A core feature for SEPIA is the ability to construct experiments using a hybrid combination of simulated, emulated and physical devices. Likewise, the ability to choose from a variety of emulated devices is useful. Each emulated device has tradeoffs to consider. For instance, an experiment that required higher throughput in the routing core could use a Vyatta virtual appliance whereas another experiment interested in Cisco specific behavior could use a Dynamips virtual appliance running the Cisco IOS image. Furthermore, an experiment requiring both high throughput and Cisco specific behavior in different portions of the experiment could assign each type of virtual appliance accordingly. The strength of the SEPIA approach is the flexibility to put available resources where it is most appropriate.

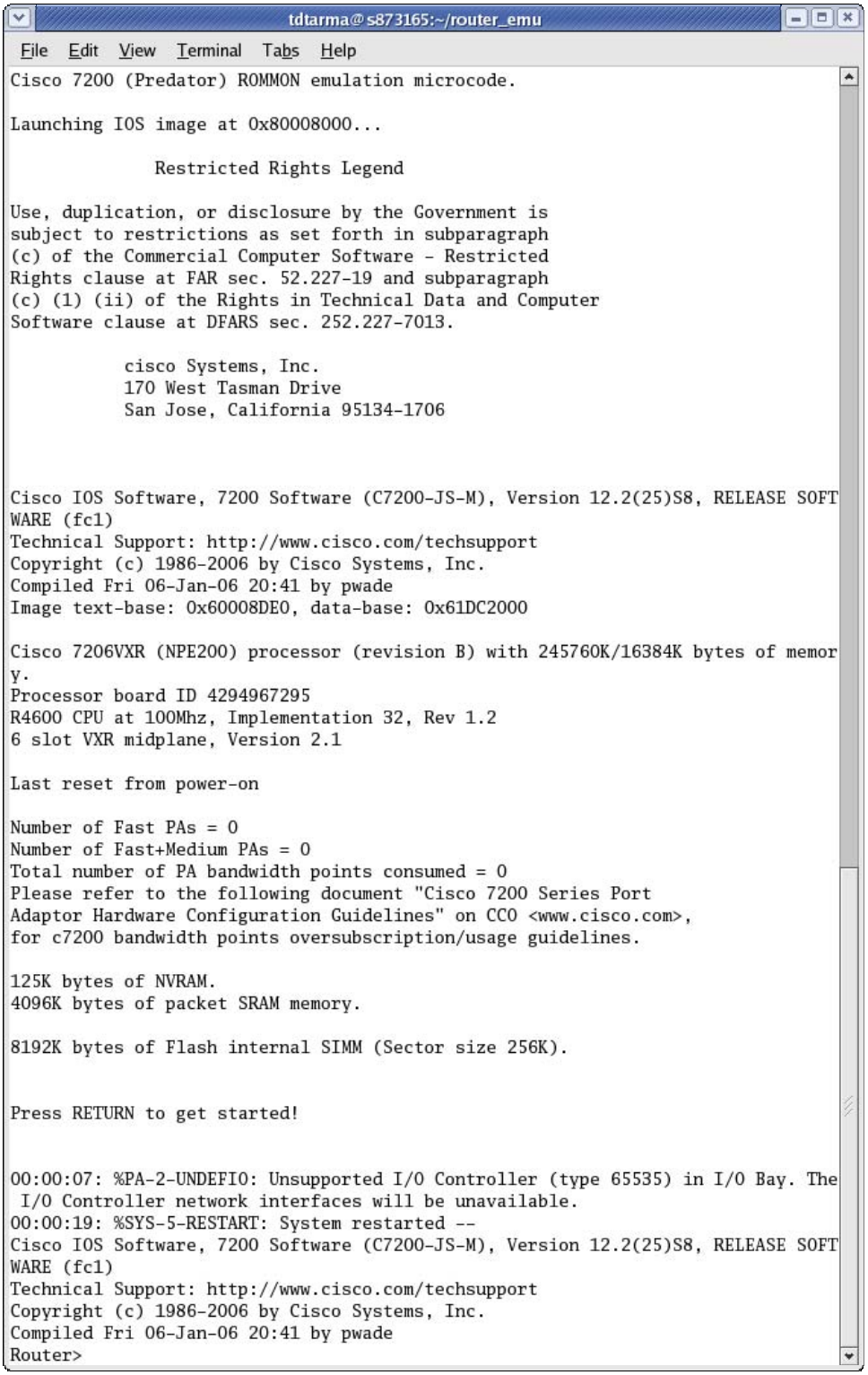


Figure 2: IOS boot screen from Dynamips virtual router

4. Network Simulation

In comparison to a decade ago, current simulation tools have sophisticated capability and increased accuracy. Unfortunately, not many of these tools have reached a new realm of reality in network experimentation that allows simulation to interoperate simultaneously with real network traffic. OPNET, a key supplier of network simulation software, has notionally facilitated this area of great interest in the research world. A means of interfacing real network traffic with simulation traffic gave birth to OPNET's system-in-the-loop (SITL) interface. SITL uses the winpcap library for Microsoft Windows machines and the libpcap library for UNIX-like machines to interface live traffic packets between simulation and the real or emulated nodes. This advancement creates a new realm for researchers to study and evaluate network performance.

Continuing with our interest in analyzing BGP in large scale networks we developed extensions to the OPNET Modeler SITL interface. The objective was to create a hybrid testbed comprised of real, emulated, and simulated devices. The hybrid testbed provides for rapidly re-configurable experiments with high accuracy.

Since SITL as provided by OPNET is not capable of the functionality required for our BGP analysis significant extensions to SITL had to be developed. SITL in its current form supports a limited set of protocol translations thus it is expected that continued development of SITL extensions will be necessary as experiments with different objectives are required. Thus the first network simulation task was to create a methodology for developing SITL translation functions. This methodology proved effective in the development of the BGP experiment translation functions.

Extension to SITL translation code

The current SEPIA environment employs OPNET Modeler with the SITL interface to provide the simulation partition of experiments. The SITL interface provides various layers of translation between real and simulated packet networks. SITL as provided by OPNET is readily available to enable a simulated network to pass real Internet Protocol (IP) traffic between two real networks. This requires SITL to translate packet data at Layer-3 and below of the OSI network protocol stack. In this case SITL ignores packet data at Layer-4 and above. However, for experiments that require interactions between real and simulated protocols at Layer-4 and above custom development is necessary to create necessary translation functions. Regarding the BGP analysis, significant SITL extensions are necessary which are described in the context of the BGP experiment.

Our demonstration experiment requires that real and/or emulated devices employing BGP protocol interface and interact with simulated devices employing BGP. This required functionality that the standard SITL capability does not support. Specifically,

BGP relies on TCP to exchange BGP control information with its peers. This required the development of SITL translation functions for both TCP and BGP.

SITL Translation Function Development Methodology

In experiments that incorporate both simulated and real or emulated devices translation functions are necessary to interface packet or datagram between to the two domains. During experiment development the protocols that are expected to pass between the simulated and real domains must be identified. These protocols must have translation functions available or translation functions must be developed. OPNET SITL currently has a limited set of protocol support that includes [opnet doc]:

- Ethernet,
- Address Resolution Protocol (ARP),
- Internet Protocol version 4 (IPv4),
- Internet Control Message Protocol (ICMPv4),
- Internet Protocol version 6 (IPv6),
- Internet Control Message Protocol version 6 (ICMPv6),
- User Datagram Protocol (UDP),
- Open Shortest Path First (OSPF),
- Routing Information Protocol (RIPv2).

Our interest to analyze BGP operation with a hybrid experiment required developing SITL translation functions for both BGP and TCP. Our methodology includes accessing the IETF protocol standard documentation to identify the information exchanges and data formats the protocol should follow. The standard definition implementation may vary which requires the translation function developer to examine the implementation of both the simulated and the real protocol. Our development methodology includes simulated and real packet sniffers to examine protocol control message content.

In our development activities and experiments the use of both simulated and real packet sniffers proved very valuable. On our real nodes we include Wireshark, a packet analysis tool, to examine the actual packets targeted for the simulated nodes or those packets having just been translated from simulated to real. Without a packet sniffer packets that may have had an incorrect field or incorrect checksum would be quietly dropped and troubleshooting would be very difficult. Likewise a packet sniffer in the simulated network proved very useful during debugging. Unfortunately Modeler does not include a packet sniffer that can promiscuously listen and parse TCP packets. For our development, we created a simulated sniffer that captured TCP packets and parsed their fields. The values from each field were displayed on the debug console for examination.

The translation function development also includes an initial network model done entirely in simulation. This initial network model provides us with a view of the messages exchanged among BGP peers strictly in simulation. Next we replace a part of

the simulation model with a SITL interface and configure the SITL interface with the appropriate filters. The SITL interface is configured to interface through the OPNET Modeler host to the real or emulated network device. This simulated and hybrid experiment should pass traffic that is similar to that passed by the simulation-only experiment. Next, using the simulated and/or real packet sniffers the packets can be translations can be verified.

A feature of the simulation is a *Reachability Analysis* capability that is used on the scenario to identify the routing protocols used on each of the routers and the path a packet will follow through the network. This feature is very useful during the translation function debugging and validation process.

Continuing with our methodology we verify connectivity after the real or emulated component has been merged with the simulation. The first step of this part of the methodology it to obtain successful Internet Control Message Protocol (ICMP) Pings between the simulated and real/emulated nodes used in the experiment. Successful Ping tests demonstrate at least partially correct configuration of the SITL interfaces and real/emulated interface.

SITL Translation Function Development

First, we created a custom TCP translation function for SITL. One objective of this function is to translate a TCP packet originating at the real node into an OPNET Modeler TCP packet. The Modeler TCP packet had its fields populated with fields taken from the real TCP packet. Each step of the TCP connection process must be translated. Another objective of the translation function is to translate TCP packets from simulated to real. The simulated packet fields are mapped into a real TCP packet. For this translation the TCP checksum has to be calculated and included in the real packet so that the real receiving node will not drop the TCP packet. Note that in both directions, our translation function is limited to TCP packets that are comprised of a single TCP segment. In the simulated to real direction, the Modeler segmentation-and-reassembly functions are used to access the parameter fields.

As with most modeling and simulation tools, OPNET is interested in reducing simulation run-time by incorporating efficiency methods in their models. In the case of TCP, the model has a TCP connection discovery process that is facilitated with a *key* field. In the model, each TCP socket is assigned a key value that expedites the discovery process. Thus real TCP sockets that interface with simulated TCP sockets must also have *key* values. To map real TCP sockets to simulated TCP sockets (and their associated *keys*) an extension to SITL was developed. The SITL modification was implemented as an additional process in the SITL node as shown in Figure 3.

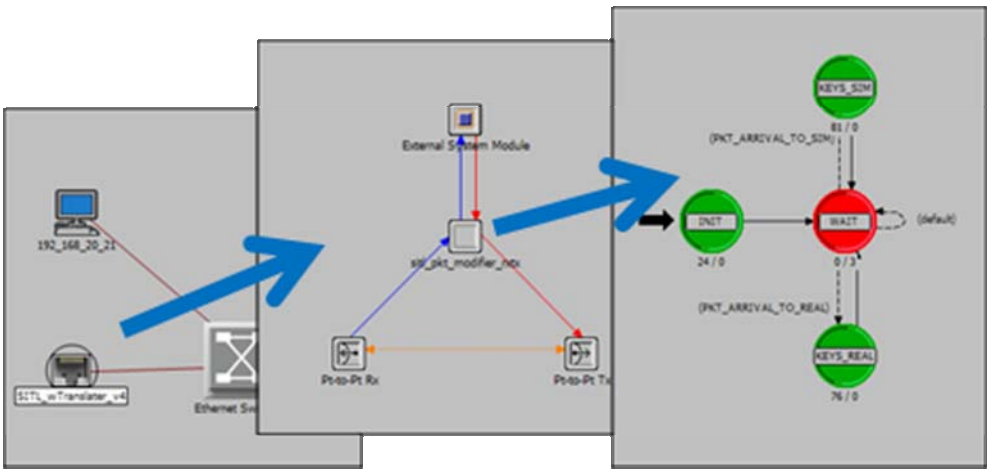


Figure 3: SITL interface node extension

Another simulation efficiency technique used in Modeler is an IP address/interface mappings for the modeled network. If the target IP address does not exist in the model network and thus not in the IP address table, simulated TCP will not attempt to connect to this IP address. When interfacing to real hardware the interface may exist outside the simulation and the TCP model should attempt to connect to this interface. To mitigate this issue the OPNET TCP is modified to bypass this test if SITL interfaces are used in the simulation.

In our development process we created a custom application process module to validate the TCP translation function operation. This process module, illustrated in Figure 5, is interfaced to the TCP layer model provided by OPNET and interfaced to a workstation node, illustrated in Figure 5Figure 4, for testing. The objective of the custom application-layer process module was to establish a TCP connection and transfer packets with a real or emulated node for testing of the translation function. After a successful TCP connection was accomplished, the custom application-layer module would receive the payload from the TCP packet and print it to the Modeler debug console and create a different application-layer payload to send back to the real-node that originated the connection. In this TCP experiment, the real-node was running a simple Python script that originated a connection and sent a text message. When the script received a reply, it treated it as a text script and displayed the text on the real-node monitor. These experiments were performed prove correct TCP connectivity between simulated-nodes and real-nodes.

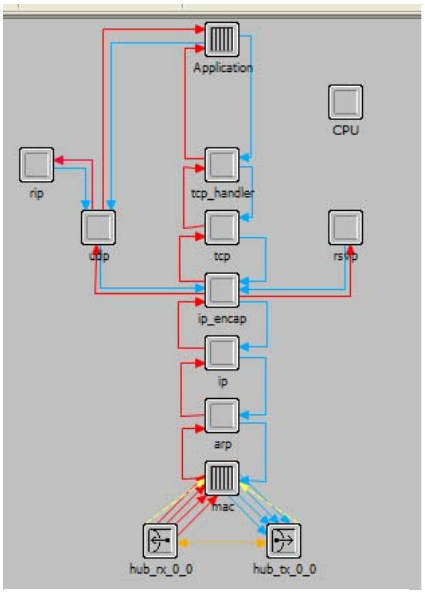


Figure 4: Custom workstation for testing TCP

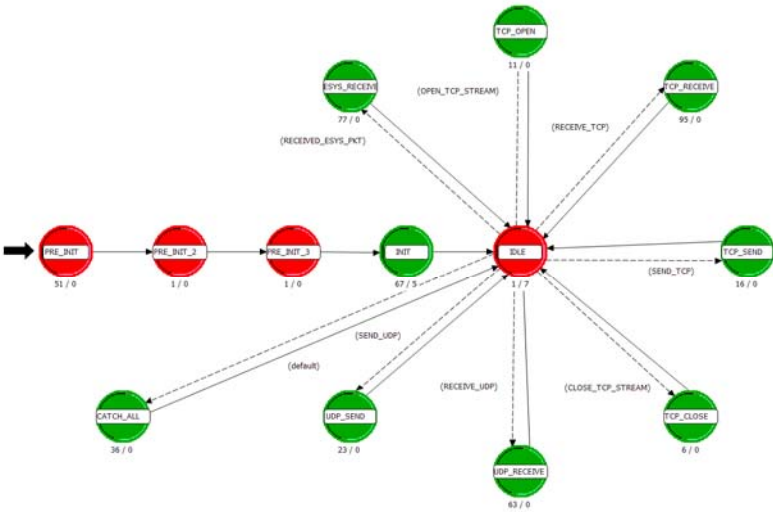


Figure 5: Application-layer process module

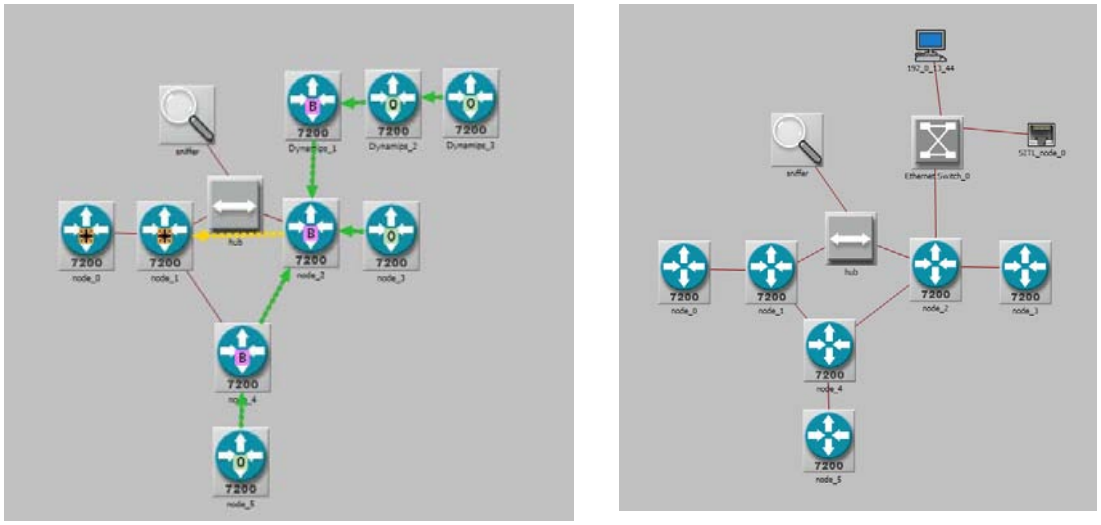
The various modifications to the existing OPNET Modeler models are performed by reviewing the model source code. OPNET does not provide documents describing the model source code. The SEPIA design team has minimized any changes to OPNET model code to maintain a forward compatibility with Modeler version releases. However, a number of changes were necessary and are well documented so that the modifications can be rolled into version updates.

Next, our translation targeted the actual BGP parameter fields carried by the TCP packets. We developed functions to translate the BGP-specific packets embedded in the TCP packets. OPNET Modeler includes an implementation of BGP as described by the Internet Engineering Task Force (IETF) Request for Comment (RFC) and includes BGP control messages as described by the standard. However, OPNET's implementation has nuances that must be addressed for their BGP implementation to be compatible with real or emulated device BGP. The following issues must be addressed when interoperating OPNET BGP with real-device BGP:

- The BGP packet type numerical value is transposed in the OPNET model. The standard defines an UPDATE message as Type 2 while OPNET defines the message as Type 3. The NOTIFICATION message has a similar issue.
- OPNET's definition of BGP Path Origin Size must be changed to reflect an 8 bit value.
- A problem also exists in OPNET's table access of BGP routes while creating an UPDATE message. The table access function was modified to increment correctly in the BGP model.

The BGP translation function development illustrates the steps of our methodology. The first step consists of establishing the model network entirely in OPNET Modeler. This allows us to view the messages exchanged among BGP peers strictly in simulation. The diagram depicting the network experiment is shown in Figure 6.

In the second step, we port a critical section of the simulation to the Dynamips Cisco router emulator. This step enables the verification of the translation function built for proper TCP exchange and subsequently BGP connectivity using our TCP and BGP translation functionality. The diagram for this step is shown in Figure 6 A. Note in this figure the routers marked "Dynamips_1", "Dynamips_2", and "Dynamips_3" in the top right corner of the diagram have been replaced by a single SITL interface. The interface transfers packets through the simulator host computer's Ethernet interface that connects to an emulated set of Cisco 7200 series router in Dynamips. The emulated routers are configured similar to to the routers in the simulation-only network. Note that in Figure 6, a custom-designed simulated "sniffer" node is built to enable us to view control packet data resulting from BGP.



A) Simulation-only **B) Combined simulation and emulation**

Figure 6: Two Versions of BGP Simulation Models analyzed for comparison

Reachability analysis is used on the scenario to identify the routing protocols used on each of the routers. In Figure 6, the routing protocols are indicated by the letter on each router. In our scenario, the BGP routers are marked with the Letter "B" while the OSPF routers are marked with the Letter "O". Next hop paths with a direct physical connection are shown as green arrows. In the scenario, the next hop BGP path without a direct physical connection is shown as a yellow arrow. In Modeler, right clicking each path will indicate the routing protocol used to create that path.

5. Experiment Creation Tools

This research developed an important and capable experiment environment to help perform analysis of communication networks and networked information systems. The experiment environment leverages existing capabilities where possible and our developments enable the combing of real, emulated, and simulation into a single environment. The environment can be used for a broad range of network analysis that can be performed with experiments constructed with an unlimited variety of configurations. The experiment configurations may be made up of all emulated devices or combinations of real, emulated, and simulated devices.

In a typical experiment, we expect analysts to design several variations of their networks under study as well as the implementations. Researchers must be able to readily develop, refine, and switch between configurations. To facilitate this work flow, we developed tools to facilitate rapid experiment creation of SEPIA models. The resulting experiment creation tools provide a mechanism to quickly create experiments of complex systems. Because the tools are largely automated, their use also helps eliminate the possibility of errors resulting from human input.

Figure 7 shows the resulting experiment flow produced by such a coupling with the end product of a hybrid analytical environment. Initially, the process begins by a user designing an experiment in the OPNET Graphical User Interface (GUI) which we extended to facilitate marking nodes that are to be emulated. Figure 8 shows how this GUI is used to specify which nodes will be simulated, which will be emulated and which will be real. Once each node is configured, the user exports the experiment as an XML file which corresponds to Experiment.xml in Figure 7.

The XML file exported from OPNET is then digested by a Sandia-designed module that separates the simulation portions from the emulation portions. The products of this module are two new files: one that describes the simulation nodes (Simulation.xml in Figure 7) and the other that describes the emulation nodes (Emulation.xml in Figure 7). One of the more formative challenges at this stage is in keeping track of the network relationships that corresponded to various network layers. For instance, the XML file exported from OPNET has different sections to describe Layer 2 relationships and Layer 3 relationships. Reconciling Layer 2 and Layer 3 connectivity requires deducing implicit relationships.

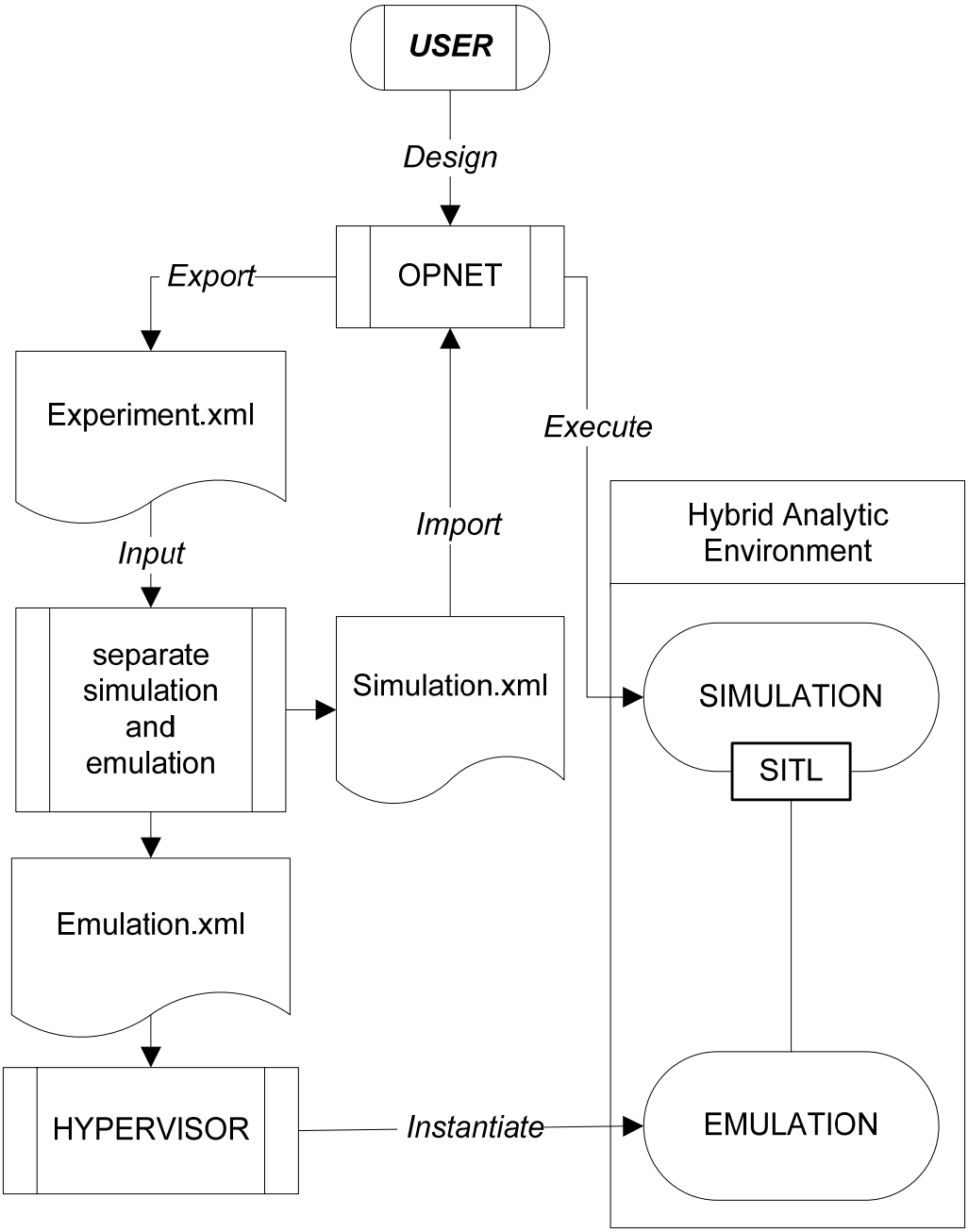


Figure 7: Experiment Flow

A technical implementation challenge at this stage of the process is in how to map the SITL interfaces to the emulation nodes. Our implementation creates separate SITL interfaces corresponding to each connection between a simulation node and an emulation node. One alternative would be to link all of the emulation nodes to one SITL with connectivity to each simulation node connected through a hub to the SITL. The disadvantage to this approach is that the SITL would not be able to filter traffic for a particular portion of the simulation network. Also, if a hub were used the potential of significantly altering the original network topology designed by the user increases.

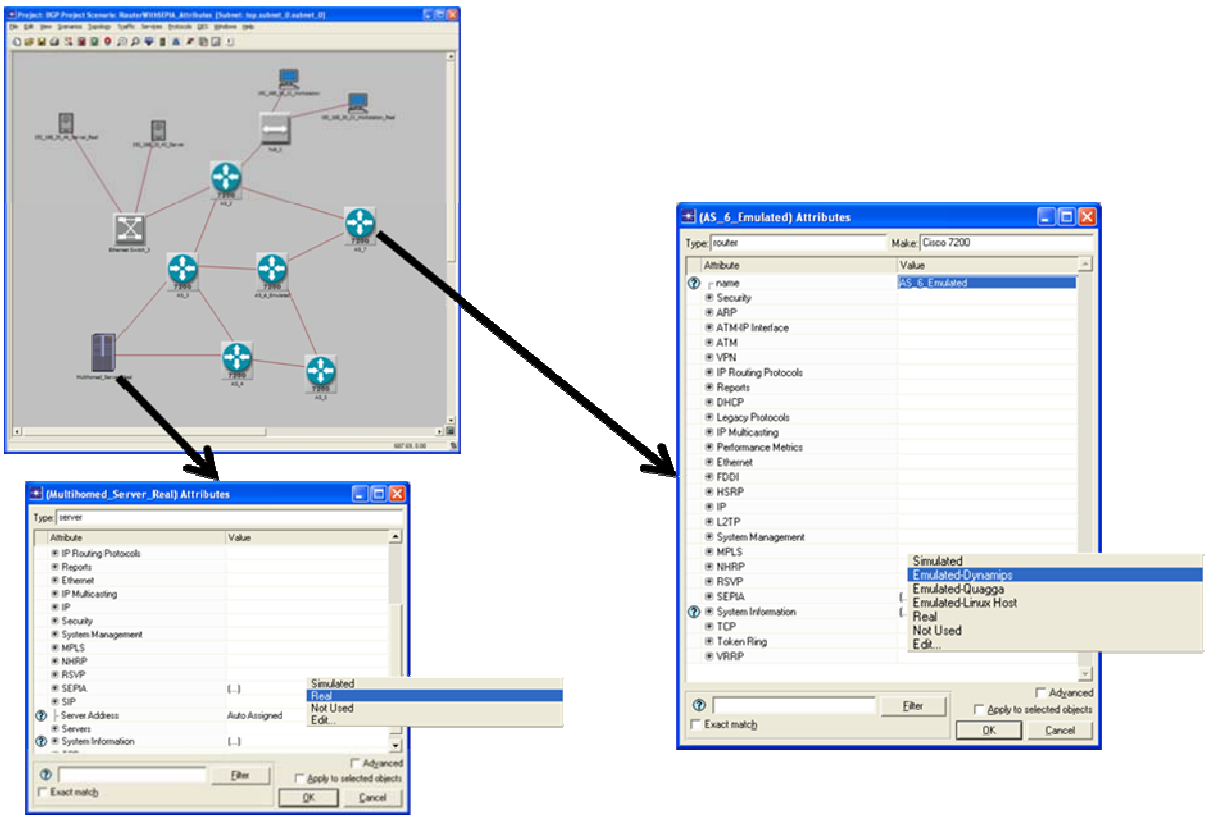


Figure 8: Configuring nodes the experiment configuration GUI

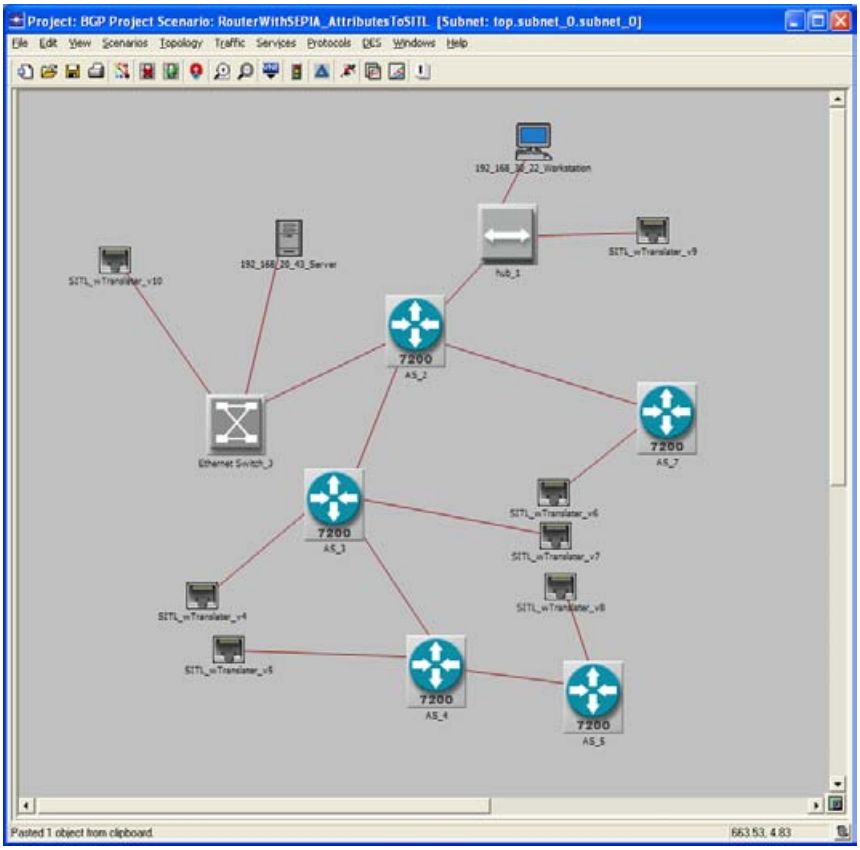


Figure 9: Resulting OPNET simulation model.

The simulation file (Simulation.xml) produced by the separation module is then imported back into OPNET. Figure 9 shows an example of an OPNET model loaded from such a file. Once OPNET executes this experiment it functions as the simulation portion of the hybrid experiment with simulation and emulation interactions facilitated by the SITL interface.

A hypervisor program was produced that receives as input an XML file. The hypervisor first parses the imported XML file (Emulation.xml) creating directories and configuration files needed for each emulation node. Next, the hypervisor instantiates each emulation node in uniquely identified UNIX screen sessions. The identifier is used for real-time interaction with each emulation node during the experiment. The real-time experiment interaction is useful for both investigative analysis of the experiment as well as debugging of the experiment. Figure 10 shows an example of a hypervisor running two Dynamips emulators.

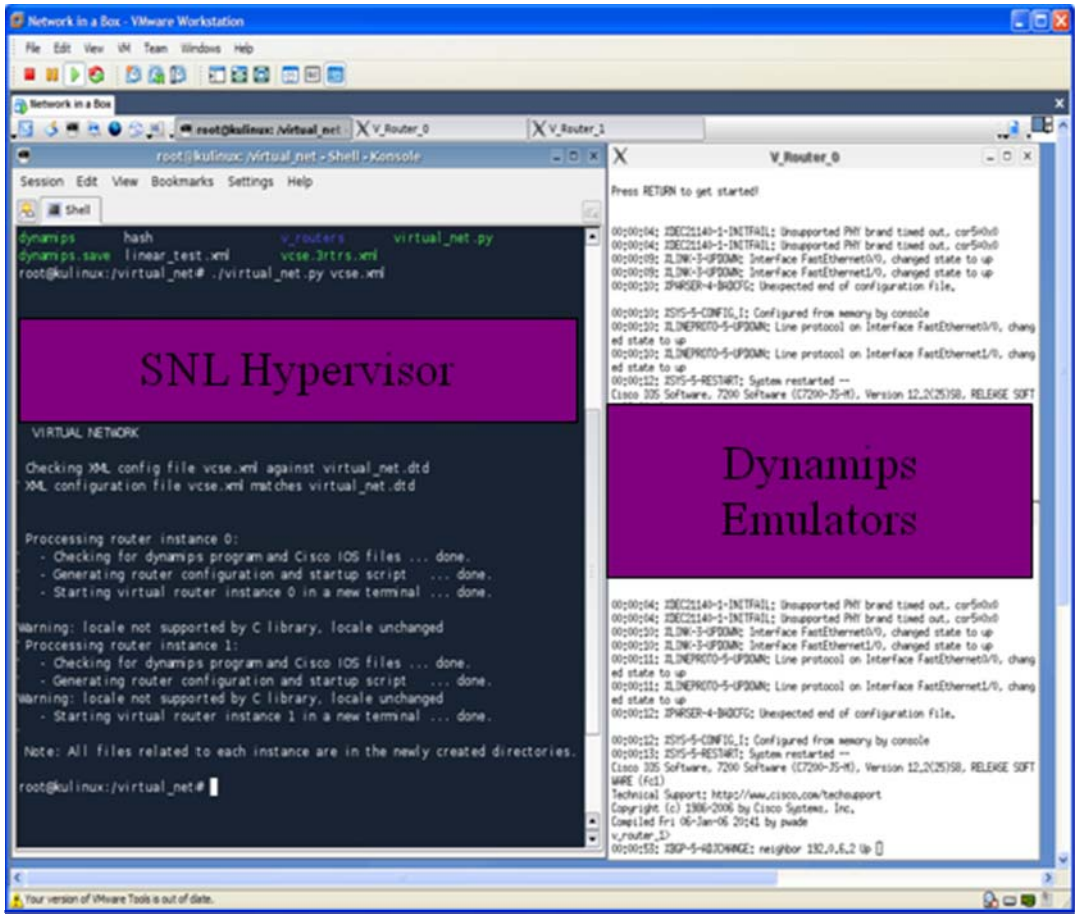


Figure 10: An SNL Hypervisor running two Dynamips emulators

6. Simulation Runtime Estimation

This section discusses simulation runtime estimation, which is used to predict real-time performance of the simulation partition of the SEPIA experiment. It describes algorithms and methods that effectively estimate whether or not particular classes of simulation can run at real-time on the supporting compute platforms.

We found that the key characteristics that affect runtime are

- Number and type of simulated device nodes,
- Number of SITL interfaces,
- Filter level of SITL interface,
- Degree of connectivity,
- Protocol usage, and
- Expected traffic loads

In our current studies, we identified traffic loads along with number and type of network devices as primary parameters for work-load estimation. Our goals are to

- Estimate whether the simulation computing platform can meet the real-time constraints required by the experiment;
- Estimate the amount of computing resources needed to perform a specified simulation within real-time constraints.

Note that the simulation performing faster than the real-time does not create a problem as we can automatically induce delays as needed. The reverse, simulation lagging real-time, is the main problem of concern.

In an illustrative set of experiments, we created a series of basic network models that included two SITL interfaces and a simulated *device under test* – the simulated devices were (a) straight-through crossover cable, (b) Ethernet switch, and (c) Ethernet router (CS7200). These experiments are shown in Figure 11.

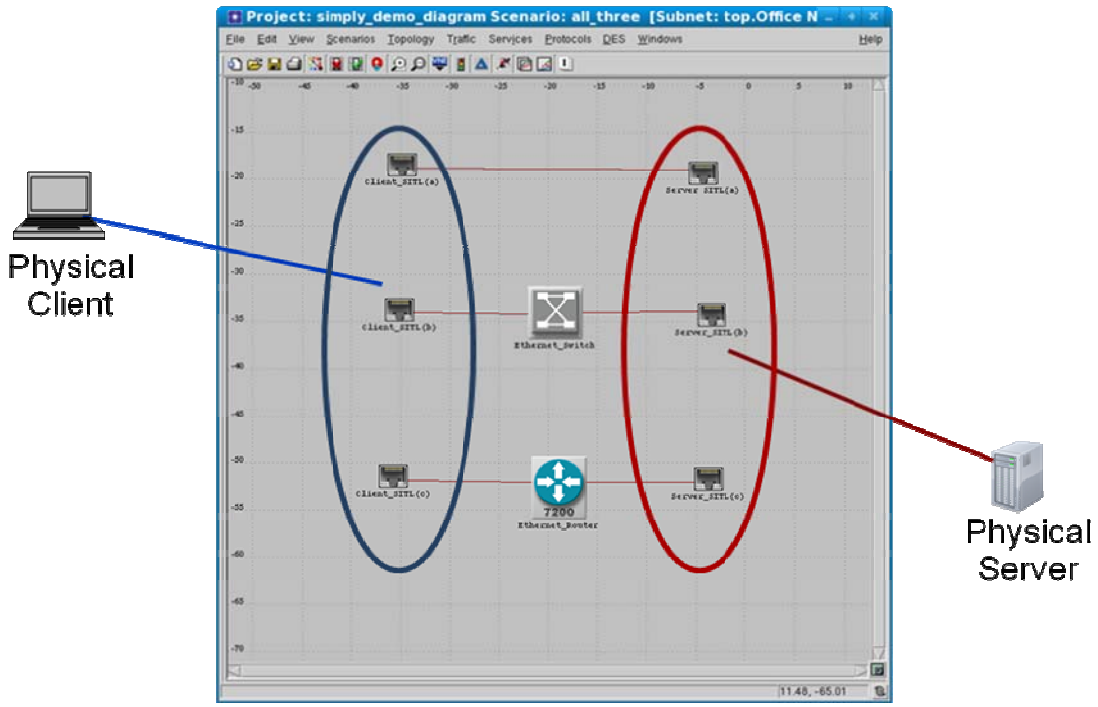


Figure 11: Experiment setup with three client-server scenarios – straight-through, single switch, and single router

In the first experiment, a UDP data stream is generated by a physical client application outside of the simulation and transported through the simulated network to a physical server application (as shown in Figure 11). The in-coming real packets are converted to simulation packets and vice versa as the traffic flows from the client to the server. IPerf is the application used to generate and sink the traffic. Using trial-and-error, the data traffic rate is increased to just below the point where the simulated network begins to drop UDP datagrams (stress testing to less than 0.5% data loss) and key performance data is collected. Figure 12 illustrates the maximum bandwidth variation with UDP datagram size. The graph illustrates that the network saturated primarily on an increasing packet rate as opposed to an increasing data rate within the limits tested. In other words, the simulated network along with the SITL interfaces is limited by its ability to forward packets as opposed to being strictly limited by the bit transmission rate.

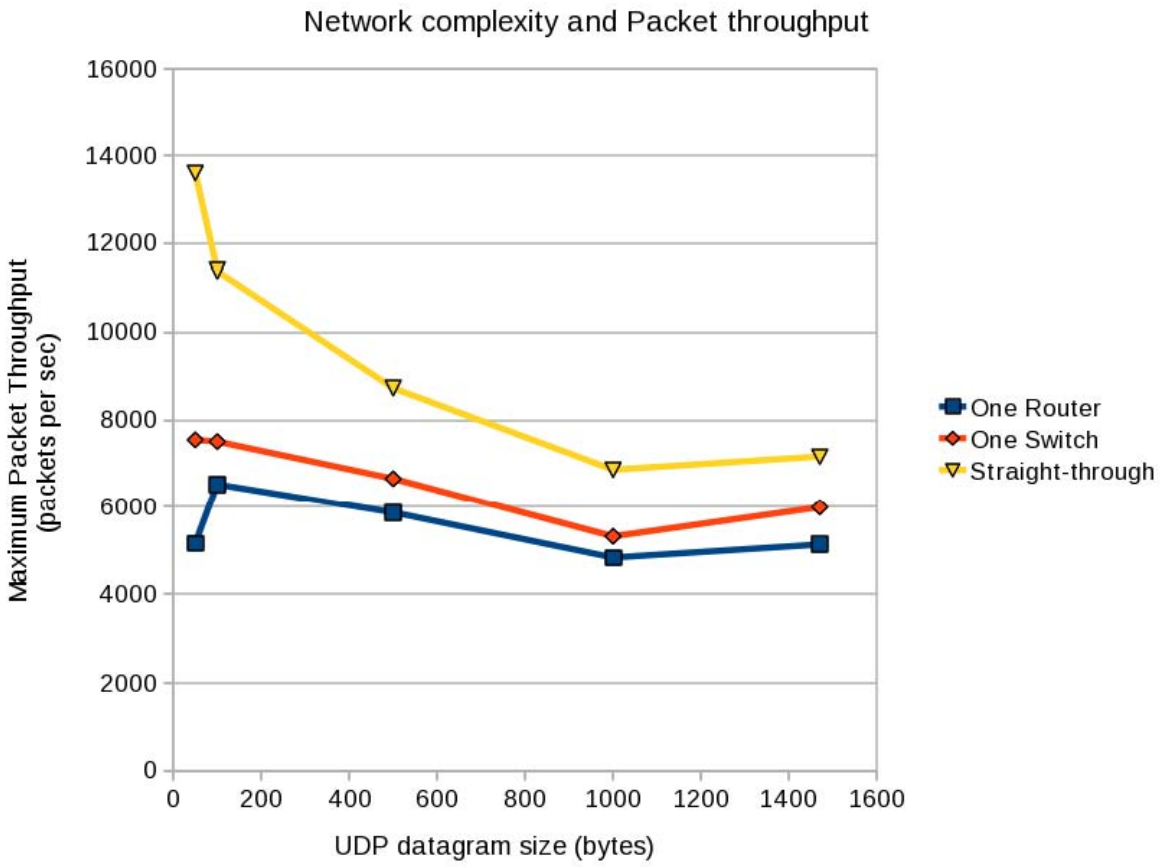


Figure 12: Maximum packet rate vs. datagram size

In another illustrative experiment, we maintained the same network models but set out to obtain the variability in computation cost as dictated by real traffic throughput within OPNET. Figure 13 illustrates the three simulation topologies' computation cost as measured by Central Processing Unit (CPU) events-per-second. In this experiment, we fixed the UDP datagram size to 1470 bytes and increased the UDP data rate to 80 Mbits/sec. From the plots, we see that the cost of two SITL interfaces grows linearly to a maximum of 40,000 events/sec at a data rate of 80 Mbits/sec. The cost of the switched network and the routed network also grow linearly but at a ratio of about 2 and 2.5 times faster than the SITL interfaces alone. Beyond 65 Mbits/sec, the routed network began experiencing more than 0.5% packet loss. This indicates that the simulation computation platform is unable to simulate the network model transmitting the experiment data stream in real time without significant data loss.

Understanding where the experiment fails to perform at real time enables the experimenter to appropriately configure their experiments. For example, they might limit network traffic, network size, or other complexities. Alternatively, they might use bigger computers or break the simulation model into multiple models for distributed analysis.

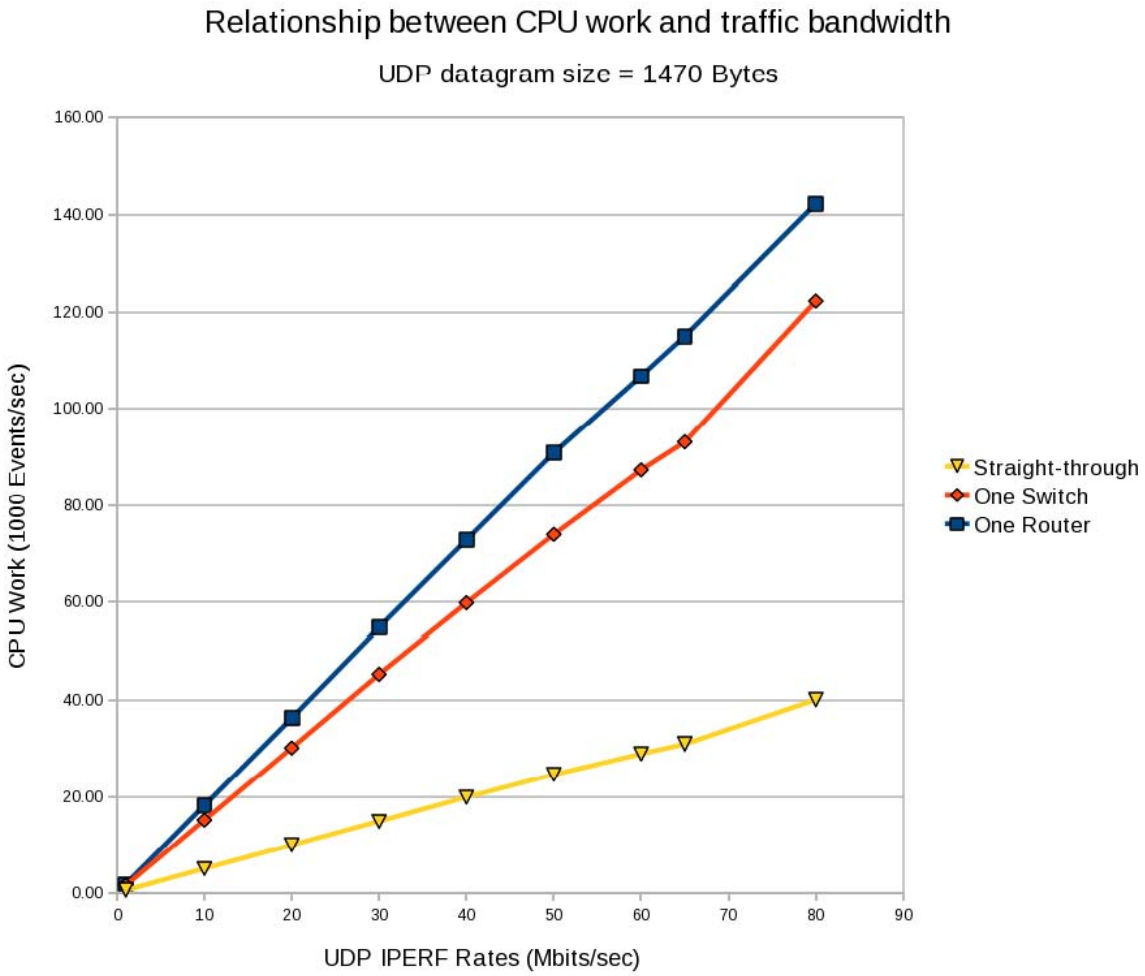


Figure 13: Simulation platform CPU utilization as affected by the modeled network data throughput

In Figure 13 the CPU Work is measured in simulation events-per-second. This metric is used since simulation computing platforms have varying CPU performance. More powerful CPUs can support larger simulation event rates and thus can simulate more complex network models in real-time.

Using our simulation performance for predicting work load

In our experiments, we have ascertained that the measure of CPU work load in terms of events per second is important for knowing how much real traffic a given network size can support. However, it is also true that all events are not equivalent. For example, the event associated with a SITL node converting real packets to simulated packets involves more memory management and have more overhead than that associated with generating a simulated packet within the simulation. One implication of this fact is that CPU work predictions need to be made with these network aspects taken into consideration.

For experiments specified in OPNET, there is a maximum number of events per second supportable by a given CPU platform. However, this maximum number is different for when an experiment deals with real traffic (due to SITL's need to do real-to-simulated packet conversion) and for when an experiment is purely simulated (neither real traffic nor SITL interfaces). Due to the nature of SEPIA, our network models will typically have SITL interfaces and hence deal with real traffic translations. The maximum number of events supportable depends on the architecture of the simulation platform as well as ongoing processes within the machine.

In the current platform used, we have an Intel Pentium D processor at 3.2 GHz with 2.0 GByte Random Access Memory (RAM) memory running Linux 2.6.29. We stress tested this machine using the network model in Figure 11 and determined the maximum CPU work with real traffic to be about 150,000 events per second. This means that within network models with real traffic, our platform will stop supporting real time simulation when the events generated exceed 150,000 events per second without having more than 0.5% loss in data. Figure 13 above showed that the CPU work necessary to simulate a Cisco 7200 router within OPNET grew linearly with the bit data rate coming into the simulation. In other words, if we call the CPU work in thousand events per second, Cw, and the bit data rate in Mbits/sec, D, then

$$Cw = a(D)$$

With measurements from our experimental data, this turns out to be Cw = 1.32*D (due to router) and Cw = 0.50*D (due to SITL). This means that with the maximum CPU work, the maximum bit rate we can support in the given platform with less than 1% data loss is (150/(1.32+0.5)) = 82.4 Mbits/sec. This is close to the trial and error estimate of 65 Mbits/sec.

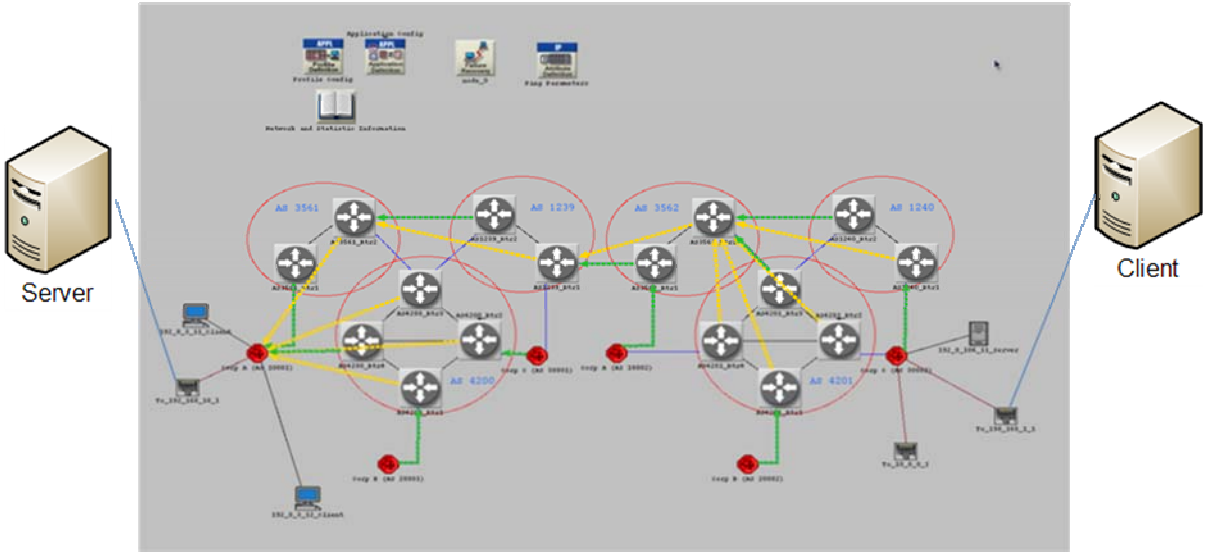


Figure 14: Enterprise network comprising of six BGP ASes and 10 routers between Client and Server

An experiment was created to demonstrate our estimation capability. Figure 14 illustrates a model of an enterprise network. The enterprise network consists of three SITL interfaces (only two were actually used based on their filter settings), twenty-one routers, three simulated machines, and Digital Signal (DS3) links between Autonomous Systems (AS) which are circled in red on Figure 14. In the enterprise network, the routers used within the ASes are atm4_ethernet2_slip8_gtwy models as opposed to CS_7206_6s_a2_ae8_f4_tr4_slip16 models. Once again, we use IPerf to generate UDP datagram streames between the physical client and server nodes with a datagram size of 1470 bytes. However, from our experiment shown in Figure 13, the CPU work, Cw, is expected to be a linear function of the incoming traffic. However, we still need to determine the coefficient of variability (slope) as the new router model in the enterprise network was different from the CS7200.

To estimate the slope, we ran a sample initial case with 1 Mbps of incoming traffic through the enterprise network. This resulted in an average CPU work of 7,300 events per second. With this in mind, we determined the slope as 7.3; this means that the relationship is Cw = 7.3*D. Since there are 10 routers going from one SITL to the next (that is, the links in "green" along the topmost routers plus one for each subnet in "red octagon"), the cost of each atm4_ethernet2 router in thousand events per second is Cw = (7.3-0.5)/10*D = 0.68*D, which is about 2.7 times less than a CS7200 router model. Table 1 below shows the predicted performance and the actual results of simulation.

Table 1: Performance prediction of enterprise network in Figure 14

Iperf Rate (Mbps)	Projected Performance (1000 events/sec)	Actual Performance (1000 events/sec)
1	7.3	7.3
10	73	76
20	146	150
30	219	160

Table 1 illustrates the correlation between projected and actual performance and hence predictability in the network. However, at 30 Mbits/sec, we were only generating 160, 000 events per second as opposed to 219 Mbits/sec expected. At first sight, this appears as a discrepancy between the actual CPU work and the projected performance. However, at 30 Mbits/sec, the simulation platform could no longer support the real traffic as illustrated in the third subplot in Figure 15. The subplot shows the lag between the blue (received traffic at SITL) and purple (sent traffic from SITL) lines. In addition, feeding the network at the rate of 30 Mbits/sec resulted in a 30% data loss and a CPU usage of 99%. This means that the computation has become the bottleneck and the simulation could no longer keep up with real time. Furthermore, we can give a stronger affirmation that the maximum bandwidth our enterprise network (Figure 14) can support is (150/7.3) = 20.5 Mbits/sec with less than 0.5% data loss, which corresponds to the observed data in Table 1. If we extend the experiment and replace the routers in the enterprise network with Cisco 7200 models, we can expect that the maximum real traffic rate supportable will be 150/(1.32*10+0.5) = 10.9 Mbits/sec.

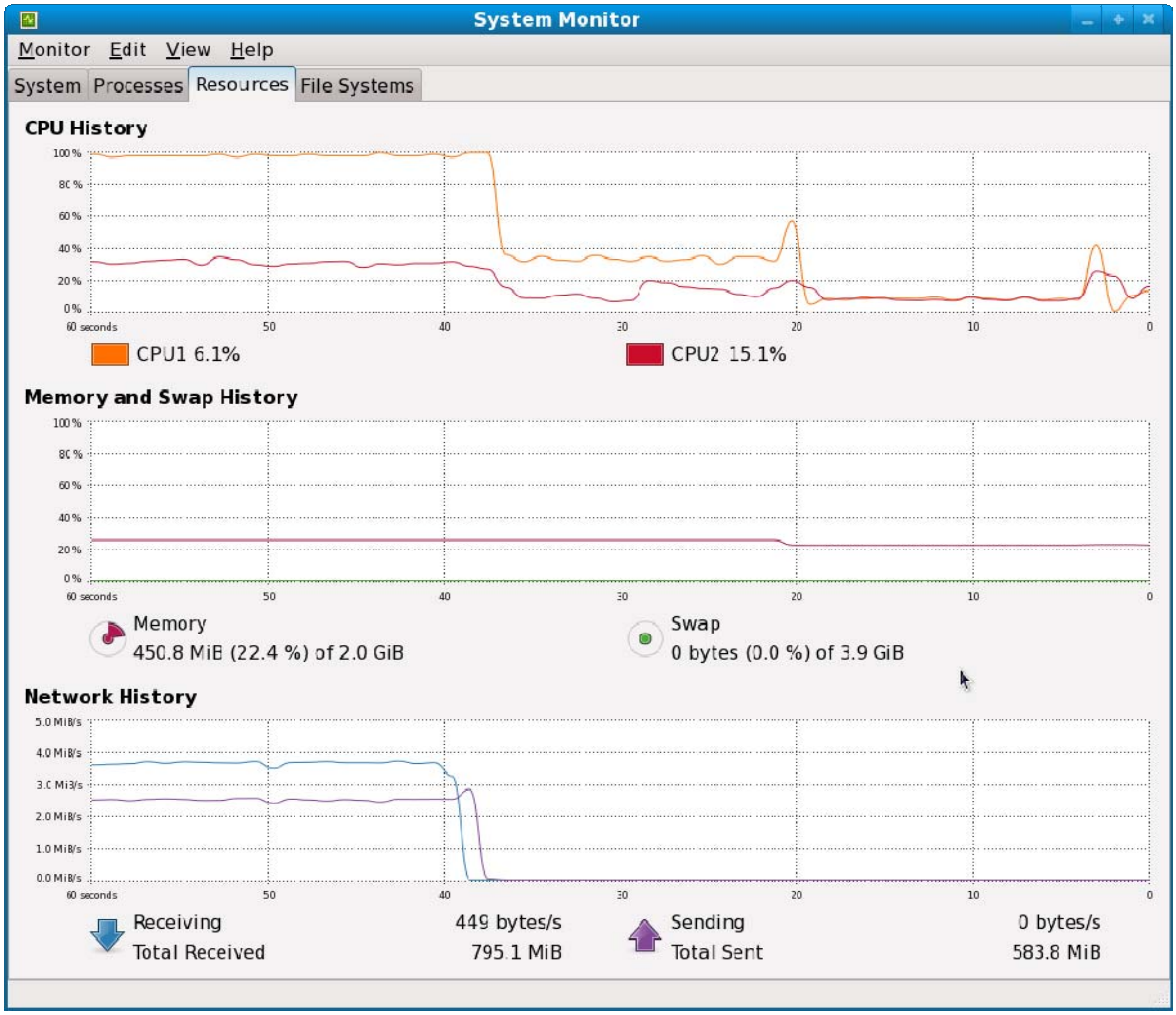


Figure 15: CPU, memory, and network interface usage due to enterprise network simulation at 30 Mbps real traffic

7. SEPIA DEMONSTRATION EXPERIMENTS

Twice in 2008 BGP vulnerabilities received international attention. The SEPIA environment proved useful in analyzing both incidents. These BGP vulnerabilities center on the preference that BGP puts on a longer prefix over other routing metrics. When coupled with a lack of validation for BGP updates, these vulnerabilities can have significant consequences. These vulnerabilities have been known to the network community for some time and solutions have been proposed [HEPNER-09].

Investigating the events, analyzing the effects of various configurations and intimately understanding such scenarios are very important to computer network security. Yet without a suitable testbed such analysis is simply speculative. Conducting such an experiment on the actual Internet is not a responsible option and since assembling several real nodes is cost prohibitive and time consuming constructing a testbed comprised of only real nodes is not a reasonable solution. The remainder of this section describes these issues in details.

Experimental Testbed

Creating network experiment testbeds within corporate network environments is difficult. Researchers need significant latitude to test their experimental networks. Network operations security personnel need to maintain corporate network integrity. Unfortunately, the research often features activities that can disrupt the operations or security of the larger networks. Thus, it is inadvisable to simply house these research networks within the corporate networks. At the same time, corporate networks offer valuable features. In particular, they allow researchers, who often work in a physically distributed environment, to better work together and share data. The need is for a testbed architecture that can be located within the corporate networks but is otherwise isolated from the corporate networks. We call this isolation idea the enclave concept.

Many technologies were considered in developing our enclave architecture. An important facet of this research is the ability to demonstrate that the physical, emulated and simulated network devices are communicating with one another. Confirming that packets are actually being transmitted and received requires a basic and lightweight infrastructure. The infrastructure does not need to be fully featured in order to demonstrate the capabilities of SEPIA. For instance, a lightweight web server can be used instead of a fully featured web server that can manage thousands of users. The lightweight nature of the services lends itself towards the use of virtual machines.

An additional requirement is that the experiments need to be easy to run remotely. This includes the ability to conduct demonstrations in a comprehensible, easy, reproducible and remote manner. Often, various elements of the environment use complex GUIs that cannot be readily used over basic (e.g., telnet) connections. Furthermore, it is important that disruptions within the underlying research networks do not disrupt the

researcher's access to the nodes. This all requires a solution that allows researchers to remotely log into each virtualized machine on the network without using the research network itself for that access.

The resulting architecture was developed through many develop, field, then test iterations. Ultimately, two technologies, Virtual Private Networks (VPNs) and VMware ESX, rose to the top and solved almost every problem. The final solution utilizes a split-tunnel VPN and VMware ESX-based system. Figure 16 illustrates a high-level topology of the final architecture. It shows how the remote clients can access the Enclave through the VPN and gain access to an experiment hosts (which can be physical or virtual) to interact with the network experiment.

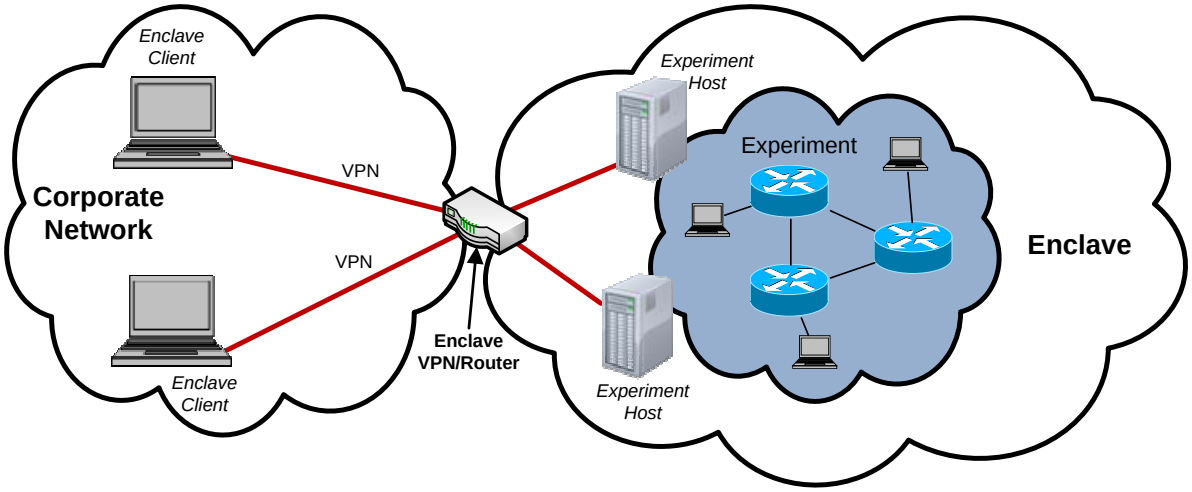


Figure 16: Testbed Topology

Our experiment testbed uses a VPN Split-Tunnel for researcher access. This solution is only appropriate for research networks that do not test malware. By using a split-tunnel, an authenticated VPN user is able to access both the corporate network and a local LAN (where the experiment is running) using a single physical network connection. This connection is facilitated through a VPN client software application. The benefit of using this technology is that the user is able to connect to corporate resources such as Web sites, File Transfer Protocol (FTP) sites, code revision control systems and etc., while being able to conduct experiments on the private network. One advantage is that it releases some bottlenecks and conserves bandwidth as Internet traffic does not have to pass through the VPN server. However, with some experiments, a split-tunnel may not be sufficient to isolate the network from damage, since the VPN is accessible through the corporate network.

ESX is a VMware "bare-metal", enterprise-level virtualization product that provides a platform for virtualized infrastructure enabling management and access of virtualized resources. ESX provides the ability to abstract many of the services and technologies being used. It also ESX provide the ability to quickly snapshot, backup, and rapidly reconfigure both the host and the network, thus providing the ability of rapid development without significant overhead to the end user. In the testbed, the user did

not know whether or not the service they were using was virtual or physical, nor did it matter.

The ESX virtual switch addresses many important Layer 2 needs, such as the need for creating bridges, virtual interfaces and taps. An additional benefit of using ESX was the ability to demonstrate the capability of the laboratory in a far easier fashion. In contrast Virtual Network Computing (VNC), virtual interfaces, X-Forwarding and Remote Desktop Protocol (RDP), which we tested, did not integrate well to provide the needed capabilities as an easy-to-use system.

Another feature of ESX is that it allows experimenters to consolidate significant experiments to a single physical box. The combination of virtual switches, with virtual vlans and multiple Ethernet ports enables the users to create an entire experiment within a single machine with entirely virtualized appliances and conduct the demonstration with access to physical network devices and an OPNET server. The virtual appliances are able to move from one virtual platform to another with limited overhead, as compared to physical hardware which does not provide that flexibility.

ESX also provides the ability to script and manage machines via CLI, the ESX Application Programming Interface (API). This provides a method for quickly instantiating and erecting large network clouds within the VMware infrastructure. Through the use virtualization, moderately large network clouds are possible, which can provide differing levels of fidelity, ranging from low to high, depending on a projects needs. The portability and remote accessibility enables other projects to utilize and leverage the tools and technologies already developed.

To summarize:

- The use of ESX provided a method of creating virtual networks that communicated on differing network segments, however, the VMware client was able to start, stop, reboot, and suspend the virtual machines in the experiment outside of the guest operating system. If for some reason the network connection was lost, it was possible to start the virtual machine and fix the issues.
- The physical routers were connected to physical machines via a terminal cable. If for some reason during the experiment, Secure Shell (SSH) access to the interface was lost, there would still be access through the terminal via the physical host.

This is just a limited list of examples, there are numerous examples illustrating that the environment had redundant, distinct methods of accessing the resources on the testbed.

BGP Studies

This research investigated two BGP-related vulnerabilities in detail. The first is the BGP Hijacking vulnerability related to the Pakistani-YouTube denial of service. In this scenario, a participant on a BGP network falsely advertises their router as having a

primary route to another party's network. This mis-advertisement is then propagated throughout the network with the effect that most network traffic is redirected to the false advertiser's network. In the case of the Pakistani-YouTube event, Pakistan's false advertisement of YouTube's addresses caused YouTube traffic from around the world to be incorrectly directed to Pakistan with the result being that users could not access YouTube until the situation was corrected.

Using simulated nodes, emulated nodes and a single real node an experiment was conducted in which an IP Hijacking scenario was performed. The resulting experiment served to more intimately understand IP Hijacking and investigate of possible detection and mitigation methods. The experiment description is illustrated in Figure 17.

The topology in Figure 17 was chosen because it demonstrates the vulnerability with very few ASes. Additionally, the hijacked route was purposefully longer to demonstrate that BGP favors length of prefix over number of hops. The consequences of BGP's favoring prefix length over other routing metrics result in the ability to hijack an IP address space that may results in a much longer path.

At the center of Figure 17 is a screen shot of the simulation segment of the experiment. The simulation segment includes multiple routers each configured as a unique AS. Attached to one simulated router is a Dynamips emulated router that provides connectivity to a real Hypertext Transfer Protocol (HTTP) server described as the Unintended Server. Attached to a second simulated router is a real Cisco 7200 series router that provides connectivity to a real HTTP server described as the Intended Server. A final connection to a simulated router is the real workstation acting as the client. The real and emulated routers connect to the simulated network via multiple SITL interfaces running our custom developed translation functions. Also note that in Figure 17 multiple custom-developed *sniffer* nodes are included to enable us to view special statistics about the BGP flows through the network.

The experiment begins with the real, emulated, and simulated routers establishing network connectivity with TCP between the multiple ASes. The TCP connections are necessary for the routers to exchange BGP route information. As the experiment progresses the demonstration network router protocols converge and the client node can access the Intended Server and is able to download an HTML webpage. During the experiment an operator causes a BGP configuration change at the emulated router that results in a longer more specific prefix for the Local Area Network (LAN) hosting the Unintended Server. The emulated router propagates updated BGP information to its neighbor router (a simulated router). The neighbor router receives the updated BGP information, reconfigures its routing tables, and further propagates the BGP information to its neighbors. The result of the BGP information updates throughout the network is that the client's traffic is redirected to the Unintended Server.

The second vulnerability studied here is the BGP Man-in-the-Middle (MITM), as recently described at Defcon [PILSOV-08] and Blackhat [HEPNER-09]. In this vulnerability, adversaries configure their BGP router to cause traffic from most places

on the network that is destined to or from the victim's network to pass through the specially configured router. Adversaries can then use this mis-direction to create a vantage point where they can observe traffic or even insert their own traffic into the message streams.

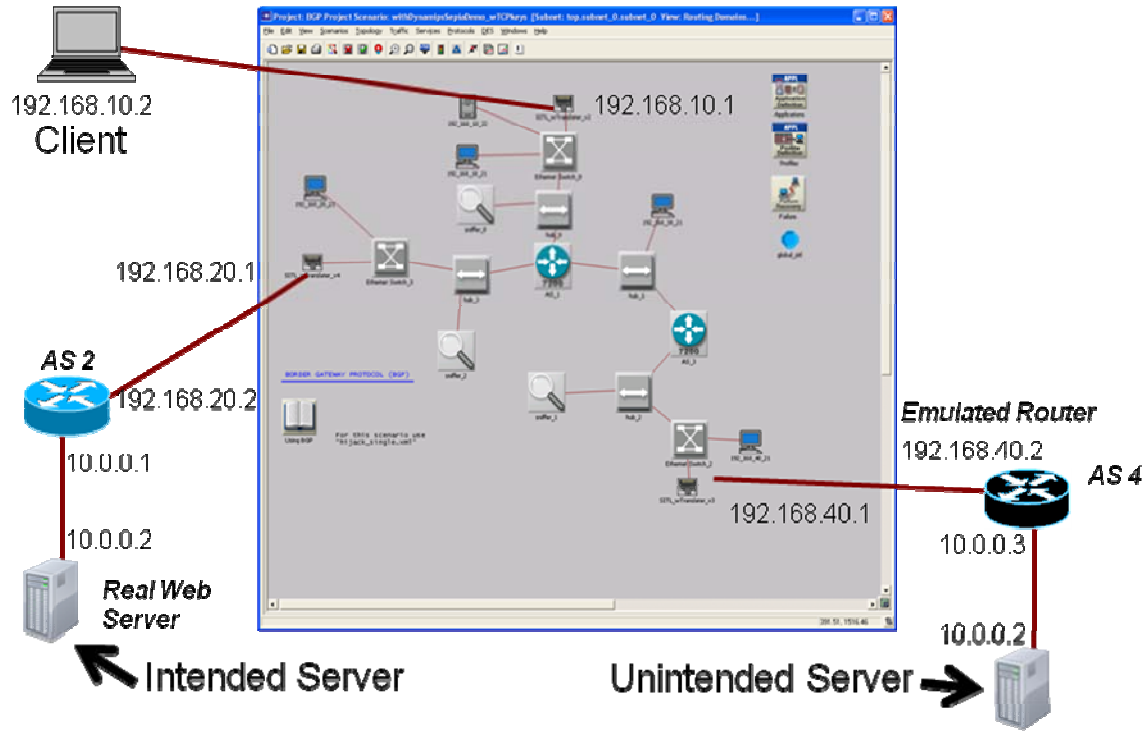


Figure 17: BGP demonstration network incorporating simulation, emulation, and real elements

Experiment Execution

In these experiments the simulation runs at real time and is set to run the duration of the experiment. The simulated routers are configured prior to launching the simulation and, unlike the real or emulated routers, the simulated routers cannot be reconfigured via a terminal during the simulation execution. However, the simulated routers behave faithfully to BGP when receiving or propagating route or other configuration updates.

The BGP experiment demonstrates that a platform where an information security analyst is empowered to thoroughly investigate potential vulnerabilities in an appropriate environment. Furthermore, exploration using the SEPIA environment allows the analyst the freedom to pursue otherwise undocumented theoretical scenarios that may produce similar consequences as the IP Hijacking scenario. This explorative activity could then facilitate proactive development of mitigation tactics, detection techniques and more robust protocols. Instead of being limited to speculative analysis and reactionary patching the SEPIA environment allows for empirical observation of potential scenarios and proposed strategies.

8. Conclusions

This report has shown that SEPIA provides an ideal approach for analyzing modern computer network security issues. It has shown how simulated, emulated and physical nodes can be combined within a single network model and how researchers can use these models as if they were real networks. In addition, the report describes how many of the modeling processes can be automated and networks can be configured and instantiated quickly. Finally, it has shown ways to predict model performance so that the analysts can understand the limits to which their models can be used and the ways that they might extend these limits.

The paper showed how several tools combine to make these networks possible. It showed that OPNET's System in the Loop (SITL) proves vital for rapid reconfiguration of large network experiments with high fidelity. It also showed how networks of emulated and real devices could be configured as XML files and instantiated through centralized hypervisor codes. Finally, it showed how these network environments could be fielded within enclaves to allow researchers to share the capability without risking network leakage.

The paper described example analyses using our simulated, emulated, and physical experimentation capability to assess BGP reachability. This included experiments to evaluate known vulnerabilities in Border Gateway Protocol (BGP); an extensively used routing protocol in the Internet and the Global Information Grid (GIG). Each experiment utilized direct interoperability between simulation nodes and emulated Cisco routers. Real and emulated Cisco routers running actual installations of the Cisco Internetwork Operating System (IOS) were interfaced to an OPNET Modeler simulated network. Emulated and real BGP routers are peers of the simulated BGP routers which communicate via Transmission Control Protocol (TCP) connections to share information about routes and Autonomous System (AS) reachability. They showed how the resultant capability provides a means to perform experiments on specific network device implementations operating in a larger simulated network. The experiments can be configured and performed in much less time and at far less expense than complete hardware experiments.

While this report provides a significant groundwork, further research remains in the area of estimating simulation run-time. As computing platforms become more powerful more complex networks with increasing scale can be simulated in real-time. Better methods are needed to estimate the part of an experiment to simulate in real-time are necessary to efficiently and effectively create functional experiments.

The SEPIA environment provides relatively high fidelity representations of key network nodes while still leveraging the scalability and cost advantages of simulation tools. Sandia National Laboratories applies SEPIA to its mission of enhancing computer security used in critical government and commercial applications.

9. References

[CISCO-SIM] Cisco 7200 Simulator (AKA Dynamips)
http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

[CISCO-NEXUS] Cisco Nexus 1000V Series Switches
http://www.cisco.com/en/US/products/ps9902/

[CHETNET-EMULATORS] http://www.chetnet.co.uk/ems/linksys.htm

[DLINK-EMULATORS] http://www.dlink.com/support/faq/?prod_id=1457

[NEUMANN -09] Melissa Neumann "Vyatta Press Release: Vyatta Raises $10 Million in Series C Round Led by Citrix," Belmont, CA – 2009-06-09
http://www.vyatta.com/about/press_releases.php?id=61

[OPNET] OPNET Technologies, Inc. www.opnet.com

[BROWN-08] Martin A. Brown, Rensys Blog, "Pakistan Hijacks YouTube," February 24, 2008
http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml

[DAWSON-2000] Terry Dawson, "NISTNet: Emulating Networks on Your Own LAN," 06/22/2000
(http://linuxdevcenter.com/pub/a/linux/2000/06/22/LinuxAdmin.html)

[FEMSTER-04] N. Feamster, J. Winick, J. Rexford, "A model of BGP routing for network engineering," *ACM Sigmetrics - Performance '04*, June 2004.

[FEMSTER-07] N. Feamster, J. Rexford, "Network-wide prediction of BGP routes," *IEEE/ACM Transactions on Networking*, April 2007, pp. 253-266.

[HEPNER-09] Clint Hepner, Earl Zmijewski (Renesys Corporation), Defending Against BGP Man-In-The-Middle Attacks, Black Hat DC 2009, Arlington, Virginia, February 2009 (http://www.blackhat.com/presentations/bh-dc-09/Zmijewski/BlackHat-DC-09-Zmijewski-Defend-BGP-MITM.pdf)

[MCDONALD-08] Michael J. McDonald, Gregory N. Conrad, Travis C. Service, Regis H. Cassidy, "Cyber Effects Analysis Using VCSE: Promoting Control System Reliability," Sandia National Laboratories Report # SAND2008-5954, Printed September 2008

[PILOSOV-08] Alex Pilosov and Tony Kapela, Defcon 16, "Stealing The Internet: An Internet-Scale Man In The Middle Attack," Las Vegas, NV, Aug 10, 2008

http://www.google.com/url?sa=t&source=web&ct=res&cd=1&url=https%3A%2F%2Fwww.defcon.org%2Fimages%2Fdefcon-16%2Fdc16-presentations%2Fdefcon-16-pilosov-kapela.pdf&ei=EmKuSrnyKYXQsgO0xMXtBA&usg=AFQjCNHvUCtybNpDA5GCTOI1cObBUUr_fw

[ROSSEY-02] Lee M. Rossey, Robert K. Cunningham, David J. Fried, Jesse C. Rabek, Richard P. Lippmann, Joshua W. Haines, and Marc A. Zissman, "Lincoln Adaptable Real-time Information Assurance Testbed," Lincoln Laboratory, Massachusetts Institute of Technology, 244 Wood Street, Lexington, Massachusetts 02420-9108, IEEE Report 0-7803-7231-X/01/© 2002 IEEE (http://www.ll.mit.edu/mission/communications/ist/files/2002_IEEE_Aero_LARIAT.pdf)

[VAN LEEUWEN-2008] Brian Van Leeuwen, Uzoma Onunkwo, Michael McDonald, "BGP Analysis using System-in-the-Loop (SITL) Testbed," OPNETWORK, Aug 2008, Washington, DC.

[VAN LEEUWEN-2009] Brian Van Leeuwen, David Burton, Uzoma Onunkwo, Michael McDonald, Simulated, "Emulated, And Physical Investigative Analysis (SEPIA) Of Networked Systems," MILCOM-2009, October 2009, Boston MA.

DISTRIBUTION (Electronic Copies):

1	MS0806	Timothy Berg, 9336
1	MS0672	David P. Burton, 5629
1	MS1209	Bernie Clifford, 5096
1	MS1235	Gregory N. Conrad, 5631
1	MS0807	William R. Cook, 9530
1	MS1235	Marshall Daniels, 5633
1	MS0671	Jennifer M. DePoy, 5628
1	MS1206	Nancy Durgin, 5632
1	MS0932	Terri L. Galpin, 9515
1	MS1235	David Harris, 5632
1	MS0672	Robert Hutchinson, 5629
1	MS1206	Michael H. Johnson, 5625
1	MS0621	Benjamin McBride, 5632
1	MS1235	Michael J. McDonald, 5633
1	MS0621	LeAnn Miller, 5636
1	MS0671	John Mulder, 5628
1	MS0806	Uzoma Onunkwo, 9336
1	MS0672	Andrew Othling, 5641
1	MS0549	Michael H. Pendley, 5632
1	MS1235	Steve Rinaldi, 5633
1	MS0620	Victor C. Romanelli, 5636
1	MS1330	Peter Sholander, 5349
1	MS0620	Thomas D. Tarman, 5632
1	MS0621	Tan Thai, 5625
1	MS0933	Vincent E. Urias, 9515
1	MS0672	Brian P. Van Leeuwen, 5628
1	MS1235	Charles Villamarin, 5633
1	MS0899	Technical Library, 9536