Design and Implementation of XBurner

Toshiyuki Miyachi
National Institute of Information and Communications Technology
Asahidai 2-12, Nomi, Ishikawa, Japan
miyachi@nict.go.jp

Shinsuke Miwa
National Institute of Information and Communications Technology
Asahidai 2-12, Nomi, Ishikawa, Japan
danna@nict.go.jp

ABSTRACT

Experiments involving evaluation of new network technologies require realistic network environments. Network traffic is one of the important elements in these environments and many products and approaches are available to generate realistic traffic. However, for efficient evaluation, users should be able to configure their traffic to possess the requisite user-defined characteristics. The existing traffic generators often do not allow users to use their application software in conjunction with the generated traffic. Therefore these generators cannot satisfy user's requirements in some cases.

We propose XBurner—a platform that can be used to generate mass traffic using a number of actual and native software applications on virtual PCs. The environment in this platform is developed using AnyBed and XENebula and is controlled by SpringOS. In this paper, we describe the design and implementation of XBurner.

1. INTRODUCTION

Environments used in network experiments are composed of several elements[1]. Network traffic is one of the important elements required to implement a realistic environment, because there are various types of services in live networks and they constantly sending their traffic through the network. However, all services that may influence a target service cannot be hypothesized; therefore evaluations of the services should be performed using realistic network traffic. Moreover, the behavior of network application software is manifested in a form of network traffic; thus, introduction of traffic is a method for pretending experimental environment is larger and more complicate without using a large number of actual nodes.

A number of approaches have been proposed to generate network traffic, such as reproduction of observed traffic[2] and creation of traffic according to predefined patterns[3, 4, 5, 6]. These methods are widely employed for network-technology assessments such as performance evaluations and network analysis.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EMUTools Workshop 2013, March 05-07
Copyright © 2013 ICST 978-1-936968-76-3
DOI 10.4108/icst.simutools.2013.251709

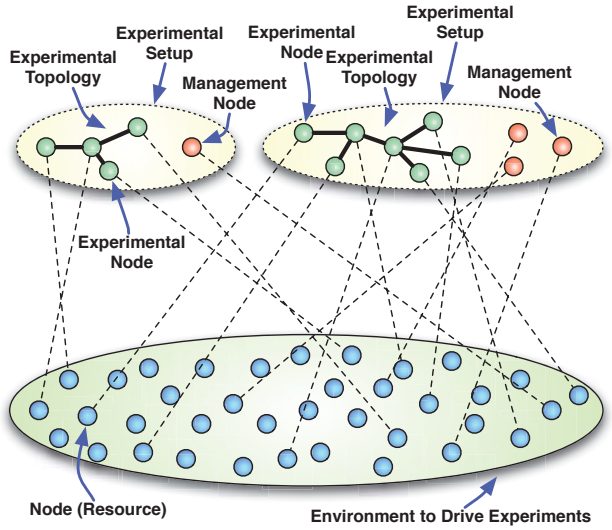

Figure 1: Architecture of Network Experiments

However, the existing traffic generators are often designed to evaluate several protocols and they cannot generate traffic by new services or introduce user-defined traffic patterns.

We implemented a platform —XBurner— that can run native software on many actual nodes and OSes. This platform can enable users to run many of their software applications on large-scale and complex topology and allow users to devise their own traffic patterns.

In this paper, we describe the design and implementation of XBurner.

2. TRAFFIC IN NETWORK EXPERIMENTS

In this section, we explain roles of network traffic in network experiments. First, we describe elements of experimental setup. Experimental setup is a environment served for conduction a network experiment, including the elements of management and experimental nodes. Figure 1 shows the architecture of network experiment. Multiple experimental setups can be built on one environment to drive experiments such as network testbeds and software simulators.

Then, we describe the current applications of traffic generators and the limitations of these generators.

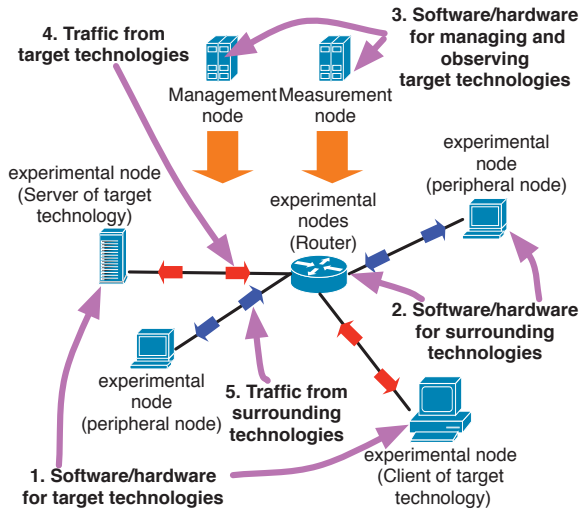

Figure 2: Elements of the Experimental Setup

2.1 Elements Constituting the Experimental Setups

A number of elements constitute the experimental setup. In this subsection, we elucidate the nature of these elements.

Software/hardware for target technologies

This is the main element in the experimental setups. This element includes hardware such as PC, switches, links and software such as OSes and the application software required to test the target technologies. The purpose of the experiments is to observe behavior of these target technologies.

Software/hardware for the peripheral technologies

There are many elements that seem to have no relations to the target technologies in live networks. In order to evaluate the target technologies accurately they should be introduced the experimental environment, but there may be some unknown relationships between the target technologies and these elements before performing experiments.

Software/hardware for managing and observing the target technologies

Although numerous types of data are required during experiments for evaluating target technologies, some types of data cannot be observed on the nodes for the target technologies. Moreover, additional software and hardware should be installed to control the software and hardware remotely.

Traffic from target technologies

Network elements generate traffic, and this is one of the most important factors in understanding their behavior or identifying problems associated with these elements.

Traffic from peripheral technologies

We have already described the surrounding elements that provide various services in a live network. These elements also generate traffic and it might affect the target technology or their traffic.

Figure 2 shows these elements on an experimental setup. As we have already mentioned, two types of network traffic are observed in an experimental setup. One is derived from the target technologies and the other is from surrounding technologies. The previous one is of course important but traffic from peripheral technologies is also important because they may have some influences to the target and sometimes there is very little knowledge on the relationships between new technologies and existing technologies. Therefore, surrounding technologies that have no apparent relationship with the target technologies should also introduced into the environmental setup.

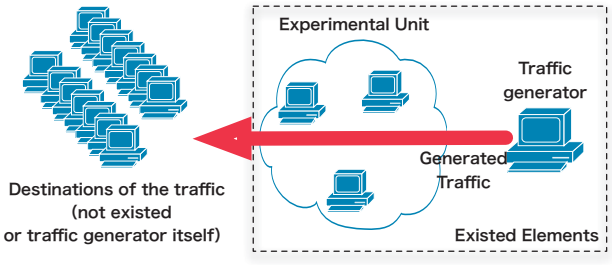

Figure 3: Reproduction-type Traffic Generation

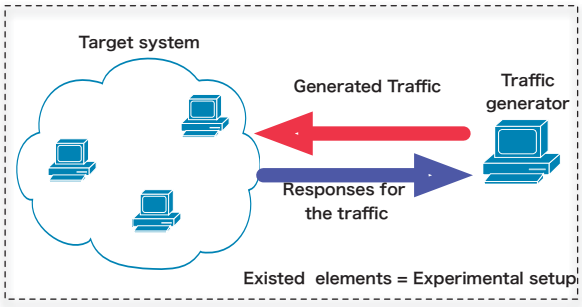

Figure 4: Communication-based Traffic Generation

The behaviors of a service on the network are manifested in a form of traffic, so introducing imitated traffic from peripheral elements that have various parameters is sufficient in some network experiments to simulate/emulate large-scale and complicated experimental environment.

2.2 Generic Use Case of Traffic Generators

Traffic generation can be performed by two approaches; reproduction of a prepared pattern or creation of actual negotiations between senders and receivers.

The first approach can be used to create background traffic or perform benchmarking of some types of services such as routing daemons. The traffic generator sends out traffic to non-existent receivers or to itself. Users may connect their target systems between the traffic generator and the destinations then they observe the behavior of the targets. In this type of traffic generation, there is no communication with the destination nodes because they don't have actual stateful peers. Thus, this approach cannot be used to evaluate protocol steps or perform benchmarking procedures that require communication between the peers. Figure 3 shows a typical application of this kind of traffic-generation approach.

The second approach creates connections and exchanges information between senders and receivers. Therefore, it can be used to evaluate the protocol steps and perform every type of benchmarking. Figure 4 indicates this type of gen-

erators. These generators make connections with the target systems and change their behavior according to the response provided by the target system.

2.3 Issues Associated with Existing Traffic Generators

The traffic generators described in the previous subsection have certain limitations.

Reproducing-type generators cannot introduce user-defined traffic patterns easily, because they generate traffic according to their own database, which often cannot be edited by the users. Moreover, in such generators, traffic generation using the users' software implementation is also difficult.

Communication-based generators are appropriate for evaluating the protocol steps of the target technologies. However, even in these generators introduction of new services and user-defined traffic patterns is not easy, because many of these generators are implemented for their own target services. Moreover, mass traffic cannot be created easily using these generators, because they manage protocol states and assemble new packets according to the response of the peers.

Currently many traffic generators support traffic-pattern definition and assemble packets. However, user-defined packets and traffic patterns may be different from specifications of services. In such cases, some of these generators cannot build "broken" packets or incorrect patterns of protocol sequences.

3. DESIGN OF XBURNER

To solve the problems described in the previous section, we designed XBurner to allow users to generate configured traffic according to their requirements.

We design XBurner as a platform to run general application software to generate traffic in order for meeting user's requests. It provide base system to run user's software on many general actual nodes with realistic and large-scale topology.

In this section, we list the requirements for XBurner.

3.1 Requirements

Firstly, we show the prerequisites for ensuring that XBurner can provide flexible environments to its users.

Capability to run user's own application software

To introduce newly developed application software and user-defined traffic patterns, we designed XBurner to be capable of running application software implementations for generic OSes. This feature also enables XBurner to run the existing traffic generators that can be run on general OSes.

Introduce realistic link characteristics to the environment

Packets on the network may possess complex characteristics, because they come through various links that have each characteristics including delay, bandwidth, jitter and so on. We provide a similar environment for the users who require realistic traffic.

Enable adjustment of the link characteristics in an experiment

Each link state on the live network changes continuously; therefore, users should be able to change the link characteristics in these experiments. XBurner can satisfy these requirements.

Introduce "Templates"

Facilitation of traffic generation is important for general users. Templates for typical traffic have been devised to help such users.

In the following sections, we describe the design and implementation features that can satisfy these prerequisites.

3.2 Functions of XBurner

In the previous subsection, we have presented a summary of the requirements for providing flexible environments to the users. The following preconditions should be satisfied to meet these requirements.

Foundation Builder

XBurner employ general hardware or OSes such as Linux and FreeBSD for traffic generation, thereby allowing users to introduce their own application software and traffic-pattern programming. XBurner also allows users to introduce several types of existing traffic generators that run on generic OSes such as Harpoon[2] and tcpreplay[7]. In addition, dummynet[8] and netem[9] can be used to change the link parameters on these OSes.

Foundation builder is a function to build environments consists of nodes on which general OSes can be run for running these software.

Burner Application Trigger

Users will generate their traffic from generic software, therefore, control methods are required for management of the software on the experimental nodes. In XBurner, the relevant function for this aspect provides user interfaces to run and terminate application software on the experimental nodes. Typically, the users should manage there application software and link emulators.

The ability to perform remote management of application software on each experimental node is called Burner application trigger.

Neighbor Interferer

The behavior of some network technologies can be affected by the topology of the network. A topology-associated problem can be identified only by running the target technology on many topologies. Therefore, XBurner provides functions to build several types of network topologies.

Neighbor interferer enables users to introduce realistic network topology.

Burner Factor Organizer

Intuitive and simple user interfaces will help users effective usage of XBurner.

Burner factor organizer provides easy interfaces for controlling the above functions.

4. IMPLEMENTATION OF XBURNER

XBurner employs existing technologies to realize the objective stated above. In this section, we explain these technologies and their implementation.

4.1 Technologies

In this section, we describe the technologies which are employed by XBurner.

4.1.1 Foundation Builder

Now many actual-node-based testbeds such as as Emulab[10], StarBED[11][12], VM Nebula[13] and XENebula have been proposed and implemented. In addition, laboratories and companies may often have small actual-node-based testbeds for their experiments. These testbeds contain a number of physical or virtual nodes that can run generic OSes. XBurner employs these actual-node-based testbeds to run actual software implementations on generic OSes. In particular, XBurner employs XENebula and StarBED-like actual-node-based network testbed for realizing foundation builder to run actual application software.

XENebula is a Xen[14]-based platform for building large-scale network experimental environment. It distributes Xen disk images onto physical nodes and configures host OSes to run in virtual nodes. Then, it initiates startup process of the virtual nodes. XENebula performs the following functions:

- Allocation of virtual nodes to physical nodes according to requested performance for virtual nodes
- Distribution of disk images and configuration files for virtual nodes to physical nodes
- Initiating virtual nodes
- Stopping virtual nodes and cleaning up configurations on physical nodes

StarBED is a network testbed that has over one thousand physical PC nodes just for network experiments. Every physical nodes on StarBED have at least two network interfaces to connect management network and experimental network(s) separately. The interface connected to management network is configured statically to access experimental node any time to configure these node including when OS installation or software setup. This management connection enables users to configure experimental network as they want. The experimental network is basically formed by utilizing 802.11q VLAN instead of changing physical cable connections.

4.1.2 Burner Application Trigger

SpringOS[11] is a software suite for controlling experimental nodes including PCs and network switches. It is mainly used in StarBED, but it can run on other environments such as small PC clusters that have the same physical state as StarBED.

SpringOS performs the following functions:

1. OS installation into experimental nodes
2. Configuration of switches to build an L2 topology using VLAN
3. Node configuration for networking
4. Conducting scenarios by executing external tools on experimental nodes

We used function 4, because XENebula and AnyBed perform functions corresponding to function 1 and 3 for virtual nodes. Currently, because of VLAN ID shortage for building large-scale environments, XENebula builds the environment using a flat network. In this case, switches configuration with function 2 isn't required, but this function is useful for creating small environment that can be built with a maximum of 4000 VLANs.

SpringOS slave software for conducting experimental scenario must be run on experimental nodes, and it invokes external tools according to user-defined scenarios. SpringOS also has function to synchronize experimental nodes by message passing. These two functions provide capacity of conducting scenario, executing external tools to run native application software on experimental nodes and slave software sends message to its master software when it needs synchronization with other experimental nodes. The scenario description of SpringOS is described in [15]. This capability also enables us to control traffic generation using native software by execute them on experimental nodes.

We made template scenarios for XBurner; these scenarios simply execute external tools that have following roles:

1. Download required native application software and tools to configure them
2. Configure these software
3. Generate traffic by starting them
4. Stop them to finish generating traffic

This template just execute simple scripts for each steps. So users can replace these scripts to others that describe user's scenario of traffic generation. If users want to make more detailed configuration, they can replace SpringOS scenario itself.

So this SpringOS function can be used as Burner application trigger and as Burner factor organizer.

4.1.3 Neighbor Interferer

AnyBed is a tool that can be used to create configurations for the quagga[16]-routing daemon according to the route information. It can generate configurations from topology databases such as CAIDA AS Relationships[17] dataset.

Using AnyBed and XENebula, users can run many virtual nodes and emulate a large-scale L3 network that is actually routed by dynamic-routing protocols.

We built Inter-AS-emulating environment with XENebula and AnyBed on StarBED by using the CAIDA database.

Figure 5 illustrates a conceptual image of the architecture for building an environment using XENebula and AnyBed.

As we have already mentioned, we employed existing link emulators such as dummynet and netem. Because these emulators run on generic Linux and FreeBSD, we introduce these emulators in the experimental nodes and control their parameters by SpringOS. Users can also adjust traffic characteristics by adjusting the timing of starting their application software and the number of application software.

AnyBed provides complex topology and link emulators generates various link characteristics on each network interface on experimental nodes. XBurner employs the combination of these tools to reproduce a real environment.

Figure 6 shows the overall schematic representation of XBurner.

When entire traffic must be injected into a single system or link for evaluating target system, XENebula environments should be separated and connected them with the target system at a single point. In this case, we can build multiple XENebula environments using different IP address ranges. However, original AnyBed and XENebula don't support IP address range specification, so we extended the application

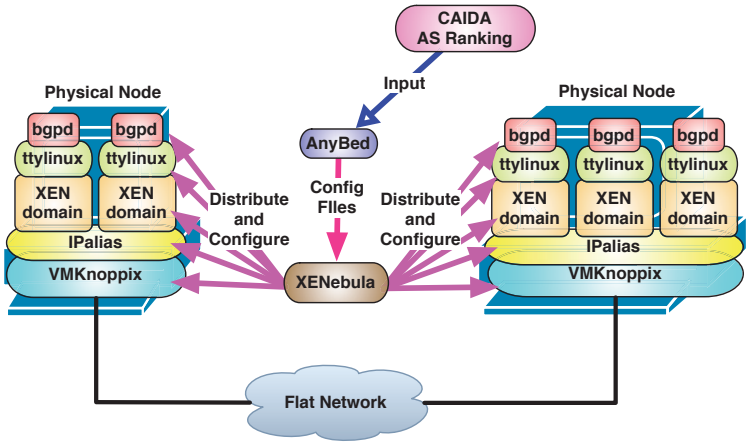

Figure 5: L3 Topology Building with XENebula and AnyBed

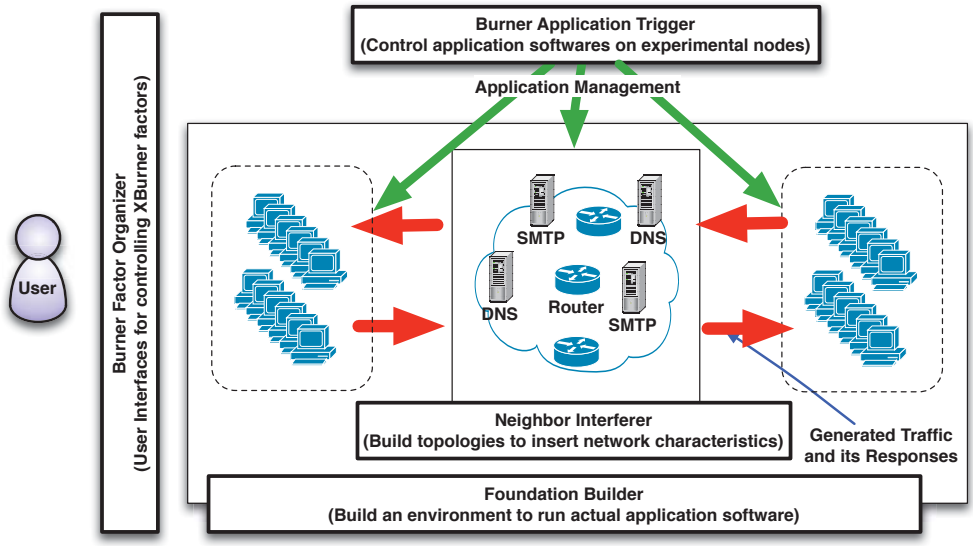

Figure 6: Traffic Generation with XBurner

Table 1: Requirements and its Solutions

Function	Solution
Foundation Builder	XENebula and StarBED
Burner Application Trigger	SpringOS
Neighbor Interferer	AnyBed
Burner Factor Organizer	Preparation of template scenarios

Table 2: Specification for StarPOD

Part	Specification	Number
Node	DELL R200 Pentium(R) E2200 2.20 GHz 4 GByte Memory 2 * 160 GB 7,200 RPM SATA HDD 2 * Gigabit Ethernet	12
Switch	Alaxala AX2430S	1

of AnyBed and XENebula. We also remake XENebula to use KVM[18] because now it is included in major Linux distributions.

Table 1 shows the required functions and the solutions for these requirements.

4.2 Implementation

To evaluate our proposal, we used XBurner for simple traffic generation. We built a large dumbbell topology using 200 virtual nodes on 8 physical nodes; this topology is illustrated in Figure 7. We built two Inter-AS KVM-based networks that were connected via a physical PC router. Each network was built using AnyBed and XENebula with the CAIDA dataset, and each network contained 100 virtual nodes. We used StarPOD, which is a demonstration environment for StarBED with have the same conceptual physical topology with StarBED. In this case, the management network is used for controlling physical nodes, and the management network for virtual nodes and the flat BGP network were built on the experimental network. Therefore, the experimental network was separated into multiple networks on the basis of IP address ranges (not in L2). The specification for the nodes is provided in table 2.

To build this environment, we created a Tiny Core Linux[19] diskimage for XBurner, which including the slave software for SpringOS and additional hard disk space. SpringOS requires the resource file of the environment, which includes the management IP address, MAC address and so on; therefore, we wrote a simple script to generate the database by using ssh and management IP address lists of the virtual nodes created by AnyBed at the time of allocation.

After building the topology, we generate traffic using our SpringOS scenario, which can be used as a template in many experiments. This scenario involves the following steps:

1. Check node status
2. Copy required files from management node to the virtual nodes
3. Configure application software
4. Start application software
5. Stop application software

We made tarballs containing scripts that perform these roles for servers and clients, and the virtual nodes obtained this tarball and extracted the scripts as a part of scenario. When the virtual machines receive messages to perform other steps, the slave software for SpringOS executed the downloaded scripts to show the appropriate behavior.

We observed about 300 to 370 Mbps traffic on the physical PC router(Figure 8). HTTP clients (`wget`) got 1 Mbyte-size

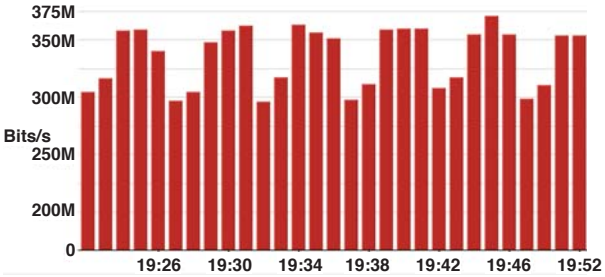

Figure 8: Measured traffic on the beside of the physical router

files from servers (`apache`) repeatedly and we didn't tuneup the performances of servers and clients in this experiments, therefore the system will show much more performance when users configure the application software appropriately or use more physical nodes. The router should be replaced with a target system, in the case of practical evaluations.

This experiment shows that XBurner can run as a platform for traffic generation. Further, we prepared a KVM disk image for XBurner, this image can be used for various traffic-generation, because the scenario can download the specific files includes required scripts for each experiment and users only must replace the tarballs for their requirements.

5. CONCLUSION

Introducing network traffic into experimental environment is indispensable to evaluate target technologies accurately. Experimenters may want to use native application software for real networks to generate traffic or make their original packets and traffic pattern for the evaluation. In this paper, we described XBurner, to satisfy these request by using generic OSes on which native software run. It also provide large-scale network characteristics. The evaluation using native traffic will enable the following processes:

- Using multiple implementations of one specification including implementation bugs and behaviors that is not described in the specification
- Implementation of user-defined packets and traffic patterns
- Statefull evaluation of target system

Generating mass traffic using actual application software is not easy, because such generation requires many nodes for running these application software, and these application software isn't designed for traffic generation. Therefore, we proposed and implemented XBurner, which enables users to run a number of actual application software using many actual nodes.

XBurner employs existing technologies such as XENebula, AnyBed, SpringOS, and link emulators. We developed the following features to supplement these technologies:

- AnyBed and XENebula extensions to specify the IP address range for the environment
- Template writing for typical usage of XBurner

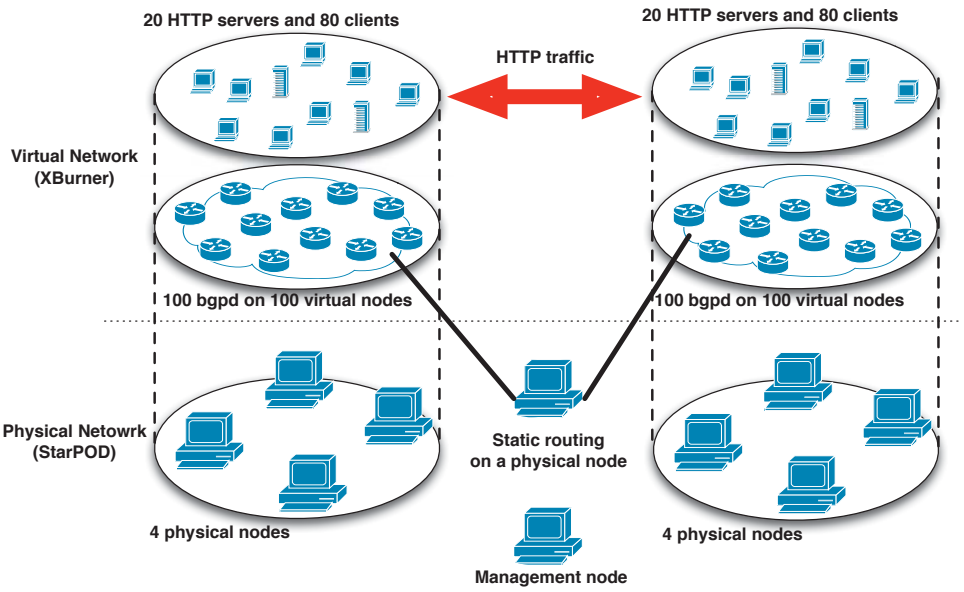

Figure 7: Experimental Environment

In this paper, we have described the design and a sample implementation of XBurner. The implementation shows the effectiveness of this platform. However, XBurner is merely a platform for generating traffic using native application software. The most important point of consideration is the kind of traffic that is required for evaluating the target technologies. We intend to use XBurner as a tool to develop appropriate traffic model for several situations, such as those involving background traffic, and routing technologies and so forth.

6. REFERENCES

[1] Toshiyuki Miyachi, Shinsuke Miwa, Ken ichi Chinen, and Yoichi Shinoda. On the Nature of Network Experiments —Issues to Automate Network Experiments—. In *The 20th IFIP International Conference on Testing of Communicating Systems and the 8th International Workshop on Formal Approaches to Testing of Software (TESTCOM/FATES2008) Short Paper, TESTCOM/FATES 2008 Supplementary Proceedings*, June 2008.

[2] Joel Sommers and Paul Barford. Self-configuring network traffic generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, October 2004.

[3] Spirent Communications. `http://www.spirentcom.com/`.

[4] Agilent N2X. `http://advanced.comms.agilent.com/n2x/`.

[5] Antonio Pescape Alessio Botta, Alberto Dainotti. Multi-protocol and multi-platform traffic generation and measurement. In *INFOCOM 2007 DEMO Session*, May 2007.

[6] BreakingPoint Elite. `http://www.breakingpointsystems.com/products/breakingpointelite`.

[7] Tcpreplay. `http://tcpreplay.synfin.net/trac/`.

[8] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.

[9] S. Hemminger. Network emulation with netem. In *Australia's National Linux Conference*, April 2005.

[10] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. pages 255–270, Boston, MA, December 2002. USENIXASSOC.

[11] Toshiyuki Miyachi, Kenichi Chinen, and Yoichi Shinoda. StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In *International Conference on Performance Evaluation Methodologies and Tools (Valuetools) 2006*, October 2006.

[12] Toshiyuki Miyachi, Takeshi Nakagawa, Ken ichi Chinen, Shinsuke Miwa, and Yoichi Shinoda. StarBED and SpringOS Architectures and their Performance. In *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011)*, April 2011.

[13] Shinsuke Miwa and Hiroyuki Ohno. A Development of Experimental Environments "SIOS" and "VM Nebula" for Reproducing Internet Security Incidents. In *Journal of the National Institute of Information and Communications Technology*, volume 52 numbers 1/2, pages 23–34, October 2005.

[14] Ian Pratt. Xen and the art of open source virtualization. In *the Proceedings of the Linux*

Symposium Volume Two, July 2004.

[15] Ken-ichi Chinen, Toshiyuki Miyachi and Yoichi Shinoda. A rendezvous in network experiment — case study of kuroyuri. In *International Conference on Testbeds and Research Infrastructures for the Development of Networks and CommunitiesąÊTridentcomąË*, March 2006.

[16] Quagga Software Routing Suite. `http://www.quagga.net/`.

[17] CAIDA. AS Relationships. `http://www.caida.org/data/active/as-relationships/`.

[18] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the Linux Virtual Machine Monitor. In *Proceedings of the Linux Symposium*, pages 225–230, 2007.

[19] The Core Project —Tiny Core Linux. `http://distro.ibiblio.org/tinycorelinux/`.