VIPNet: IPv6 Testbed

***Abstract* – *D*ue to the continuing growth of Internet, IP addresses have become scarce and valuable resources. A new Internet standard, IPv6, solves the problem of address scarcity because it has a very large address space (2^{128} addresses). Therefore many network environments need to deploy ipv6. Before deploying ipv6, we need to test how the applications will run exactly as if they were running in the real Ipv6 environment. With this requirement, Testbeds/virtual environments came into play. However there is a lack of testbeds to test and run ipv6 applications. We address this problem by proposing a Testbed called "VIPNet" to create ipv4/ipv6 network environments to test and deploy applications. Furthermore VIPNet can be used as a virtual learning environment to facilitate on-line learning.**

I. Introduction

Due to the rapid development of World Wide Web and increment in usage of Information Technology to cope educational, research, business, and many other fields, IPv4 has become a limited resource.

However, researchers managed to find a solution, called IPv6 or Internet Protocol Version 6 which supports 2^{128} unique IP addresses, a significant increase in the number of computers that can be addressed. Even though IPv6 has several advantages over IPv4, it is still not popular among the users. The major reason behind this is that there are very few testbeds to run and test their IPv6 applications before put them into use. Users cannot test how the applications would run in the new environment without using a proper testbed. This would lead very few people to deploy IPv6 in their networks and make IPv6 underutilization.

Furthermore in the field of education, there is an inclination towards using Virtual Environments to replace real systems. The main advantages we identified in this approach are its flexibility and cost effectiveness. For example, if we use virtual machines to facilitate learning activities in a school it will reduce the cost of managing resources such as class rooms and equipments. From the student's perspective, learning will become more flexible and less expensive. In the field of education we call these Virtual Environments as Virtual Learning Environments (VLE). Even though goal of these environments is to assist distance learning, nowadays it is also being used as a supplement for traditional learning environments. Additionally, most of the currently available VLEs are rich with features to let students to have theoretical knowledge rather than having practical knowledge on a particular subject, although practical knowledge also contributes to form ideas and theories.

Considering the problems and requirements, we propose a viable solution called "VIPNet" using virtualization technologies. In simply, VIPNet can be considered as a tool that can create a virtual IPv4 or IPv6 network with only using a single personal computer. Also it can create necessary application environments for deploying and testing applications which require an IPv4 or IPv6 network.

There exist previous solutions for the same requirement such as PlanetLab[1] which facilitates mass scale research testbeds to deploy and test applications. Since, PlanetLab is a globally distributed open network, using it for testing primary application will result compatibility issues and it would not be a feasible and effective solution. Furthermore, PlanetLabs' IPv6 deployment is yet to come. Therefore, our primary objective is not fulfilled with PlanetLab.

Also it should be mentioned that, since VIPNet can be used as a Virtual Learning Environment for course units such as Networking, Systems Administration and Security where students can apply and practically experience the concepts of subjects. Moreover, since VIPNet provides a comprehensive Graphical User Interface to create and manage the environment, even a novice user can easily setup required environment and manage it.

II. Background

A. *IP version 6*

Due to recent concerns over the limitations of the current pool of Internet addresses and the desire to provide additional functionality for modern devices, an upgrade of the current version of the Internet Protocol (IP), called IPv6, has been standardized. This new version, called IP version 6 (IPv6), resolves unanticipated IPv4 design issues and takes the Internet into the 21st Century.

The increased address space available under IPv6 could stimulate development and deployment of new communications devices and new applications, and could enable network restructuring to occur more easily. The redesigned header structure in IPv6 and the enhanced capabilities of the new protocol could provide significant benefits to Internet users, network administrators, and applications developers. IPv6 could also simplify the activation, configuration, and operation of certain mobile networks and services. Although there are several advantages in using IPv6, there are very few applications compatible with IPv6. This may happens due to the lack of availability of testbeds for application developers to test and run applications. Using the virtualization we are proposing a solution to build an IPv6 testbed.

B. *Virtualization*

Virtualization is a current hot topic which resulted from cloud computing. But it has many advantages over cloud computing. One of the major advantages in virtualization is server consolidation which uses server resources efficiently. Current virtualization techniques have enabled to run several virtual machines in a single physical machine as it is run in several physical machines. Most of the data centers are using virtualization to centralize its services into a few physical servers. The objective of server consolidation is to reduce the cost associate with the infrastructure and operations as well as increase the manageability of services.

Virtualization is also can be used in developing network testbeds which can be use as platform to application development and testing. This helps research community to use this as an alternative to highly cost testbeds. Virtualization also helps activities in e-learning. In general, virtual learning environments are being used as a teaching aid for courses such as system administration, network security. There are several free, open source tools as well as commercial tools to fulfill the above requirements. Virtualization technologies can be divided into many areas but in this paper we divide it into two branches as

1. Hypervisors
2. testbeds/virtual environment

1) *Hypervisors*

Hypervisor is a virtualization technique which allows multiple operating systems to run on top of a single computer. These hypervisors emulates the systems hardware to its guest operating systems. If we consider about the layers of the hypervisors which run on top of the hardware and the guest operating systems run on top of the hypervisor. Most of the time hypervisors are used to provide "infrastructure as a service" in cloud computing. Xen and KVM are two popular hypervisors.

a) *Xen Hypervisor*

Xen is a Virtual machine monitor which supports several microprocessor architectures such as x86, AMD64, etc.[2] . The core idea behind Xen is that, it provides a hardware abstraction layer and provide facilities to run several operating systems concurrently only by using hardware of a single computer system. Xen has been developed by University of Cambridge Computer Laboratory and now it is maintained by the Xen Community. Xen systems have three main components, namely, Xen Hypervisor, Domain0, and DomainU(s) or created domain instances.

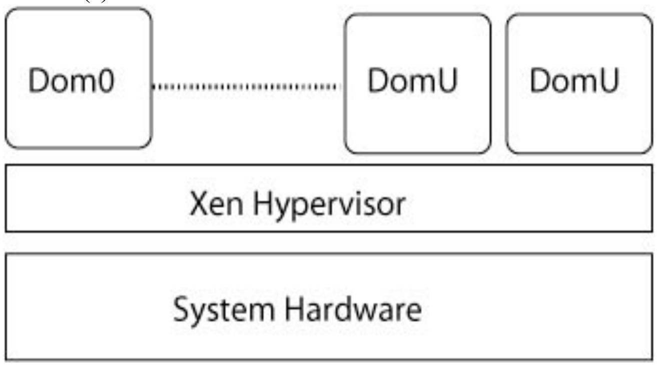


Figure 1: Xen Hypervisor structure

Figure 1 displays the basic Xen Hypervisor structure where Xen Hypervisor lays at the lowest level. Operating systems of domain components run on top of Xen Hypervisor. Dom0 has the operating system which boots with the Xen Hypervisor and only that particular operating system has access to the real hardware. The Dom0 act as the resource person for spawned domain instances identified as DomU(s). Dom0 provides a hardware abstraction for other domains and control the allocations of the resources according to the administrator preference. Therefore, the administrative user of the whole system should log in to the Domain0 to manage the other domains.

b) *KVM (Kernel based Virtual Machine)*

The main objective of KVM[3] project was to create a modern hypervisor and it was developed as a loadable kernel module which enables the linux kernel to act as a hypervisor. KVM was implemented after hardware support for virtualization was provided.

So KVM only supports hardware assisted virtualization and as a result KVM requires Intel VT and AMD-V to run. Guest operating system does not need modifications to run on KVM. KVM reuses the features implemented in the linux kernel such as memory manager, process scheduler, I/O stack etc. KVM was incorporated in to the linux kernel since 2007 and now KVM is a core part of the linux kernel.

c) *UML (User Mode Linux)*

User Mode Linux[4] project was started as a way to run Linux versions in a safer way which could be used to run software with bugs and to do experiments with new linux kernels without being problem to the main linux setup. In another words it runs linux kernel in user mode. UML uses a file on physical computers disk storage for virtual machine and it uses only the hardware which the user wants it to have. UML only supports linux as its guest operating system.

2) *Testbeds and Virtual Environments*

There are number of testbeds and virtual environments which are using UML as their core. Most popular examples are VNUML[5] and NetKit[6]. VNUML uses its own language which is based on XML to define its virtual network scenario. While Netkit also provides set of tools to manage configuration and launch a virtual network based on UML. It uses virtual hosts contain multiple network interfaces to create virtual routers which are connected using virtual switches. These routers and switches can be used to emulate the networks. There is another network tool called MNL[7] which can be used to create networks on top of Xen, VMware Server and user mode Linux. As previous tools this also uses a configuration file to create the virtual networks. Also it has its own language called MNL. There are few other similar tools such as UMLMON[8], Einar[9] Netbed[10] and PlanetLab[1].

III. System Setup

VIPNet basically use Xen as its base and it is used to create host machines network components. The following two sections will describe how Xen is being used to create host machine and network components. We opted to use Xen Hypervisor for VIPNet implementation. The main reason behind the decision is Xen is very mature and feature rich than other available open source hypervisors such as KVM. However, the proposed solution can be implemented regardless of the hypervisor technology. The concept is our main contribution.

a) Creating Host Machines

VIPNet runs on Domain0 and it has been implemented to create and manages all guest domains. These guest domains (DomU(s)) compose the virtual network together by acting as hosts, routers or switches. It is pretty straight forward to create a guest domain in Xen systems. This only requires a configuration file and a disk image for guest domain. Configuration file basically stores information such as memory allocation, network allocation, path to the kernel and ramdisk, disk partitioning information, etc. Disk image is the file system which provides necessary space for the guest operating system. Listing: 3.1 illustrates a basic configuration file with basic parameters.

```
memory     = ‘128′
root       = ‘/dev/sdaX ro’
disk       = ['phy:/dev/sdaX,sdaX,w']
name       = ‘vm-name’
kernel     = ‘/boot/vmlinuz-2.6.26-2-xen-686′
ramdisk    = ‘/boot/initrd.img-2.6.26-2-xen-686′
vif        = [ 'mac=00:16:3E:00:00:11' ]
on_poweroff = ‘destroy’
on_reboot  = ‘restart’
on_crash   = ‘restart’
```

Listing: 1

’Xen utils’ provides a set of utilities to manage Xen Hypervisor, Xen system and Xen guest domains. Initially, VIPNet uses these tools to create initial operating system image. Thereafter, VIPNet clones existing virtual machine disk images to spawn other virtual machines. This will drastically reduce the virtual network deployment time. However, if user need a different version of a virtual machine with different operating system, VIPNet can providing it in the similar manner.

b) Creating Network Components

Xen systems provide three main methods of network configurations. Namely, Bridged networking mode, Routed networking mode and Network Address Translation (NAT) networking mode. VIPNet has been implemented to use Bridged networking mode to deploy the virtual network. Using Routed or NAT network modes, it is not possible to create separate Local Area Networks (LAN). Therefore, we use Bridged networking mode for VIPNet. Xen Bridged networking mode uses Linux bridge-utils to create and configure software Network Bridge in Domain0. It is possible to create bridge for each and every network interface. Xen guest domains use these network bridges to connect to the virtual network. However, by considering the requirements of VIPNet, instead of using Linux bridge-utils, we opted to use open v switch to create network bridges for available interfaces. Further, open v switch has been used to create virtual switches of the VIPNet virtual network environment.

In VIPNet, routers has been created by attaching multiple interfaces to a DomainU(guest domain). It is possible to attache additional network interface for a particular domain by editing the configuration file. For example vif parameter of a DomainU with two network interfaces would look like
vif = [’mac=00:16:3E:00:00:11’, ’mac=00:16:3E:00:00:12’]

c) VIPNet Implementation

The GUI of VIPNet is being developed by using java language. libvirt Application Programming Interface(API) has been successfully used to communicate with the Xen Hypervisor. Libvirt is a free and open source API and management tool for managing virtual machine managers such as Xen, KVM, Virtual Box and VMware ESX. Libvirt is a C language library which has many other language bindings including Python. We opted to use libvirt over Xen API because of its comprehensive documentation and the flexibility to use the API with many programming languages. Further, libvirt API provides every feature which Xen API provides. Figure2 shows the layer that libvirt resides in the VIPNet design.

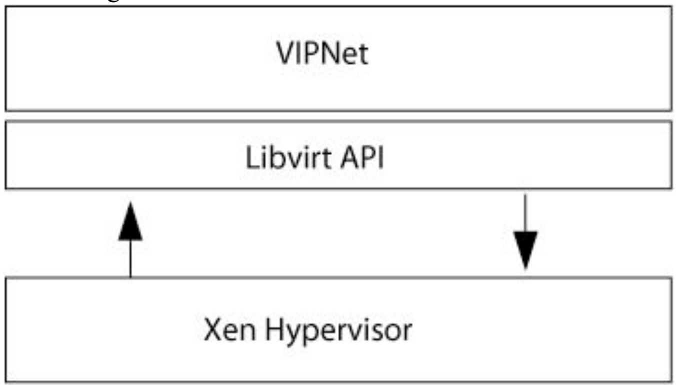


Figure 2: libvirt API

In VIPNet GUI, it provides feature such as defining and configuring virtual machines, start and stop virtual machines, etc. In addition to that, the GUI provides facilities to create switches when required by using open v switch. In addition to that, VIPNet uses Xen Store to dynamically change certain parameters of virtual machines such as network address. Xen Store is shared information storage among virtual machines. Figure: 3 illustrate how each domain and the Xen Hypervisor interacts with the Xen Store.

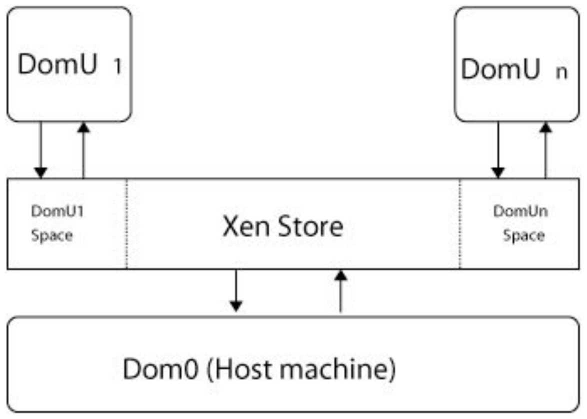


Figure: 3: how each domain and the Xen Hypervisor interacts with the Xen Store

By default all the configuration of the Xen domains are stored in this Xen Store. VIPNet uses this feature to dynamically configure the network interface(s) of virtual machines using a daemon program running inside each virtual machine.

As we described earlier, VIPNet provides feature such as defining and configuring virtual machines, start and stop virtual machines, etc. In addition to that, the GUI provides facilities to create switches when required by using open v switch. VIPNet basically consist of four main components as hosts, routers, switches and links to connect them. Figure 4 shows the main interface of VIPNet.

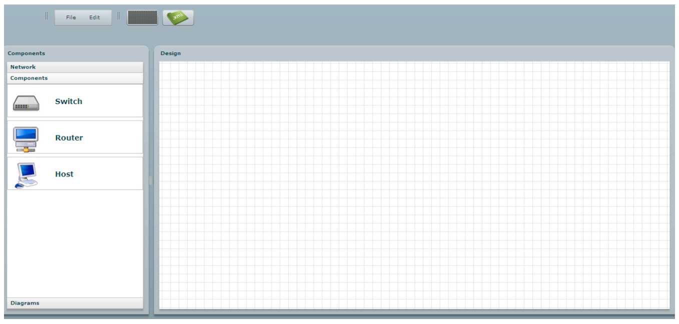
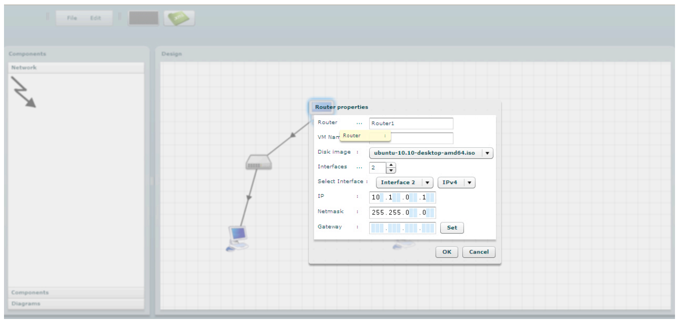


Figure 4: main interface of VIPNet

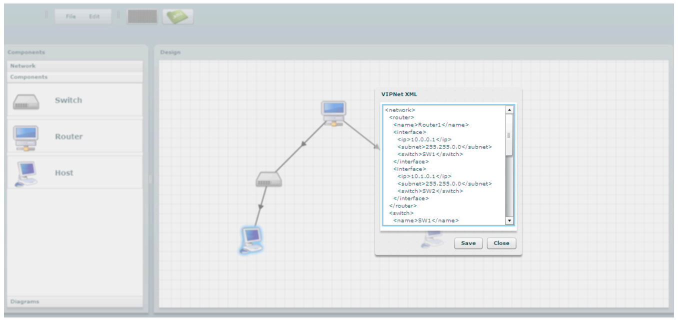
When user creates a host or a router, he has to provide the configuration parameters as shown in figure 4. Also when defining a router user needs to define the number of define the number of network interfaces it should have. Relevant disk image for the router/host will be cloned and use as its hard disk.



Router ... Router1
VM Name Router :
Disk image : ubuntu-10.10-desktop-amd64.iso
Interfaces ... 2
Select Interface : Interface 2 IPv4
IP : 10 . 1 . 0 . 1
Netmask : 255 . 255 . 0 . 0
Gateway : . . .

Figure 5: Providing configuration parameter for each host/ router

After drawing the network topology we can see the XML representation of the topology as in figure 5. After that this XML document will be parsed and through libvirt VIPNet calls Xen Hypervisor to start virtual machine. Virtual interfaces are then added into virtual switches as defined in the topology and the virtual network will be built up(figure 6).



```
<network>
  <router>
    <name>Router1</name>
    <interface>
      <ip>10.0.0.1</ip>
      <subnet>255.255.0.0</subnet>
      <switch>SW1</switch>
    </interface>
    <interface>
      <ip>10.1.0.1</ip>
      <subnet>255.255.0.0</subnet>
      <switch>SW2</switch>
    </interface>
  </router>
  <switch>
    <name>SW1</name>
```

Figure 6: XML representation of the topology

IV. Discussion

In this research we are proposing a solution to build an IPv6 testbed with GUI. We have used virtualization technologies to develop this. The proposed solution has several advantages than existing testbeds. Using the Xen virtualization tool would be the best option because past researches have been proved that Xen has better performance than any other virtualization tool [11]. Choosing Xen has increased the performance of VIPNet than choosing any other virtualization tool. Most of the existing testbeds and virtual environments like VNUML, NetKit support single Linux distribution. But VIPNet supports most of the linux distributions as well as it supports Windows if hardware support is available.

Most of the other testbeds use their own script rather than using GUI to create topology. But VIPNet use GUI as well as XML which is a universal language to create the network topology. This feature in VIPNet allows users to use the environment without learning about the environment or its scripting language. These features will make VIPNet more popular among the users than the existing testbeds. Also VIPNet can be used as a Virtual Learning Environment for course units such as Networking, Systems Administration and Security where students can apply their knowledge and practically experience the concepts of the subjects.

V. Conclusion

IPV6 has been invented as a solution to the IP crisis. Even though it has several advantages over IPv4, it is still not popular among the users due to lack of necessary infrastructure facilities to test IPv6 applications. As a solution to this problem, we propose a viable cost effective solution called "VIPNet" using virtualization technologies. VIPNet can be considered as a tool that can create a virtual IPv4 or IPv6 network with only using a single personal computer. Also it can create necessary application environments for deploying and testing applications which require an IPv4 or IPv6 network.

VIPNet has several advantages than most of the existing virtual environments and testbeds. Among them having good performance, user friendliness and ability to work with multiple operating systems are the major advantages. Since it has more enhanced features than the existing virtual environments VIPNet will become popular among the most of the users among the near future.

REFERENCES

[1] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman,“PlanetLab: An Overlay Testbed for Broad-Coverage Services,” ACM SIGCOMM Computer Communication Review, vol. 33, pp. 00–00, July 2003.

[2] xencommunity, “How does xen work?.” http://www.xen.org/files/Marketing/HowDoesXenWork.pdf, December 2009.

[3] Redhat, “Kvm - kernel based virtual machine,” November 2009.

[4] “The user-mode linux kernel home page.” http://user-mode-linux.sourceforge.net/.

[5] F. G. Márquez and D. F. Cambronero, “Distributed Virtualization Scenarios Using VNUML,” Architecture.

[6] S. Knight and N. Falkner, “How to Build Complex , Large-Scale Emulated Networks,” Proc. 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2010), vol. 40, 2010.

[7] J. Sechrest and K. Begnum, “The mln project.” Online. http://mln.sourceforge.net/index.php?page=Introduction.

[8] G. Stolpmann, “Umlmon - virtualization with user mode linux.” Online. http://www.gerdstolpmann.de/buero/umlmon.html.

[9] “Einar 1.0.” Online, September 2006. http://www.livedistro.org/gnu/linux/einar-10.

[10] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, “An integrated experimental environment for distributed systems and networks,” (Boston, MA), pp. 255–270, Dec. 2002.

[11] W. M. Fuertes and J. E. L. de Vergara, “A quantitative comparison of virtual network environments based on performance measurements,” Proceedings of the 14th Annual Workshop of HP Software University Association, 2007.