IMUNES Based Distributed Network Emulator

Z. Puljiz and M. Mikuc
Faculty of Electrical Engineering and Computing/Department of Telecommunications, Zagreb, Croatia
zrinka.puljiz@fer.hr miljenko.mikuc@fer.hr

Abstract— **In this paper we describe a new version of our distributed network emulator that extends an existing kernel level emulator called IMUNES. IMUNES is based on a lightweight virtual machine concept and performs zero copying when packets traverse through the emulated topology. It works on a modified FreeBSD kernel and enables emulated nodes to use the standard UNIX applications. The main strengths of this tool are scalability, performance and high fidelity. We are developing a distributed network simulation to further increase the scalability by allowing parts of emulation to be deployed across a peer-to-peer emulator cluster. The decentralized management of the emulator cluster improves availability and robustness of the system. We provide support for a multi-user and multi-experiment environment to maximize the benefit from newly increased resources.**

Distributed emulation, scalability, clustering

1. INTRODUCTION

In network emulations the demands for resources are rising with the complexity of the emulated network. The resources available on only one machine are imposing the limit on scalability. As a result of dealing with this problem many powerful network emulators are designed as distributed systems [1], [9] and [11].

IMUNES is a network emulator that offers high scalability, performance and fidelity. Designing a distributed emulator based on IMUNES pushes the limits of scalability even further. IMUNES is a topology specification, management and GUI application. Core of IMUNES network emulation facility are kernel level lightweight virtual machines available in modified FreeBSD kernel. Distributing the simulation based on IMUNES implies the separation of topology specification and GUI application from the management utility, and extending management utility to provide the support for distributed simulations.

Our distributed simulator works on a cluster and fulfills the following:

- Improved scalability – this is the most important demand. We expect the scalability to improve proportionally to the number of distributed hosts.
- Improved robustness – the state of the cluster is kept up-to-date on each host in the cluster. Even in the case of a failure of any host in the cluster the state of the cluster remains available and accurate.
- Improved utilization – all the available resources of the cluster should be used in an efficient manner. This means allowing more than one person to use the cluster, and to emulate more than one topology at a time.
- Improved portability – allowing user operating on some other OS, to use the IMUNES emulation capabilities.
- Keeping the performance – we are preserving the simulation properties even in the case when the simulation is distributed among the hosts in the cluster.
- Improved availability – we are providing remote access to the cluster.

We have studied the centralized architecture that is used in some other distributed emulators [1] and [11], but in the end we decided to go for a peer-to-peer cluster. Our goal is to show that the proposed distributed network emulator is a powerful tool for simulation of large networks.

For distributing the topology we used standard graph partitioning methods also used graph partitioning methods like in [13].

The rest of this article is organized as follows: in the second section we give an introduction to IMUNES, its building blocks, kernel structures and the properties. The distributed architecture in presented in the third section. In the fourth section are the results and the conclusion, whereas the future work is described in fifth section. Related work is presented in the last, sixth section.

2. RELATED WORK

Many existing network emulators are designed as distributed systems. This is partially because the network emulation is very resource demanding so the need for higher scalability provoked design of the distributed emulators that provide more resources and the new resources can easily be acquired.

The architecture of ModelNet [9] consists of highly connected router core and edge nodes. In the smallest variant of the cluster the user needs at least two machines; one for a core router and one for an edge node. ModelNet is used for emulating internet like topologies through the set of pipes. The virtual machines that are the sources and the sinks of the ModelNet traffic are implemented in user space and reside on the edge nodes. The packets going from one host to another are using routes calculated in advance so no IP level routing is present, nor is dynamic routing available. IMUNES offers the traffic shaping ability as well, providing the same zero copying for the packet when it traverses the network but it also gives the opportunity for per hop and dynamic routing.

PlanetLab [1] also uses kernel level virtual machines. The management and slice authority (slice is a part of PlanetLab resources assigned to one experiment) are currently centralized and bind together, and on our distributed emulator management utility is distributed among all the cluster nodes. The IMUNES cluster concept currently is designed to operate only on a local network, where as PlanetLab does not have that kind of restriction. IMUNES offers private standard network variables for each emulated node, whereas PlanetLab nodes use raw sockets bind to certain ports for network virtualization. On each node in PlanetLab only one VM per user per experiment can be allocated.

NetBed [11] is designed as a large cluster of PC-s running FreeBSD or Linux operating systems for emulating large complex networks. The experiment is deployed using geographically distributed machines and links. WLAN links that can not be emulated by real links are emulated using dummynet traffic shapers. It has the centralized management utility.

3. IMUNES

3.1 Operating system support

IMUNES uses GUI for topology specification. For emulation, IMUNES provides a management utility that is responsible for mapping from GUI level object to kernel level objects. The kernel level objects used for emulation are virtual machines and netgraph nodes [5]. The virtual machine (VM) support is implemented in kernel space, unlike the support presented in [10]. The drawback of user level VM support is that for each emulated packet the user space network emulator has to pass the packet to the kernel by performing data copying and context switching, and the good side is the increased portability since a user level VM would work on heterogeneous operating systems. Our support for VM-s is implemented as a modification of a kernel and offers the full functionality of standard FreeBSD system. This modification does not degrade the working speed of the hosting FreeBSD system [14]. Virtual machines are interconnected in kernel space, so the packet traverses the network topology entirely in kernel space (Figure 1.). Passing the packet form one virtual machine to another is based on pointer forwarding and thus provides zero copying.

Virtual machines used in IMUNES are based on the clonable network stack and are accessible through the vimage utility. A clonable network stack is used to create a new network stack for each new virtual machine. This means that each VM has private network resources like routing table, network interfaces, network sockets, etc. Each VM also has a group of associated user space processes (BSD jail [2]), private CPU scheduling parameters and CPU usage accounting.

Vimage utility is used for crating and managing the virtual machines, while for connecting VMs we use the netgraph interconnection framework [5]. The netgraph nodes work in the kernel space just below the device drivers and just above the link layer processing. IMUNES management utility is used for mapping the user specified topology to the virtual machines and netgraph nodes.

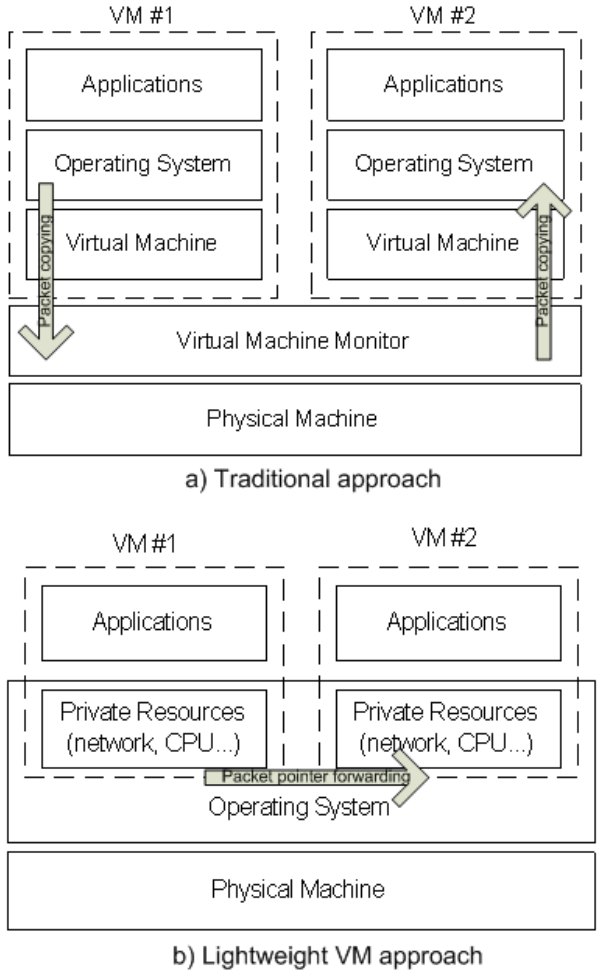


Figure 1. a) **Traditional virtualization model** (packet going from VM#1 to VM#2 is copied from user space to kernel space and vice versa) b) **Lightweight virtualization model** (Pointer to the packet is passed from VM#1 to VM#2)

3.2 IMUNES topology

IMUNES uses a GUI for drawing and viewing the desired network topology. The topology in IMUNES consists of two basic units: nodes and links. Nodes can exist independently, whereas links always connect two different nodes.

Nodes are further classified into link layer nodes and network layer nodes.

Link layer nodes are LAN switch, hub and physical interface and they provide just link layer functionality (passing the packet or dropping a packet). They are not implemented as virtual machines; they are just netgraph nodes.

Network layer nodes are the ones capable of network layer processing, and they are the nodes that are implemented in kernel as virtual machines. The IMUNES network layer nodes are host, PC and router. Network layer nodes have their own

instance of TCP/IP stack and they provide the full functionality of FreeBSD system. The difference between network layer nodes is in the booting process. The Host and PC do not forward packets and the routes are static, whereas the router is capable of packet forwarding using the routes obtained by dynamic routing protocols available through quagga [7] or xorp [12]. The Host normally starts standard network services, via inetd, unlike the PC. On virtual network layer nodes a UNIX shell can be opened and any available application can be started (Figure 2.).

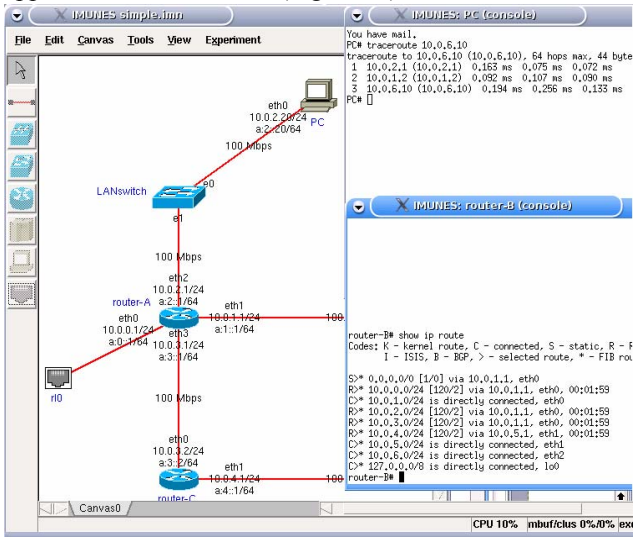


Figure 2. Standard UNIX shell opened for network layer nodes

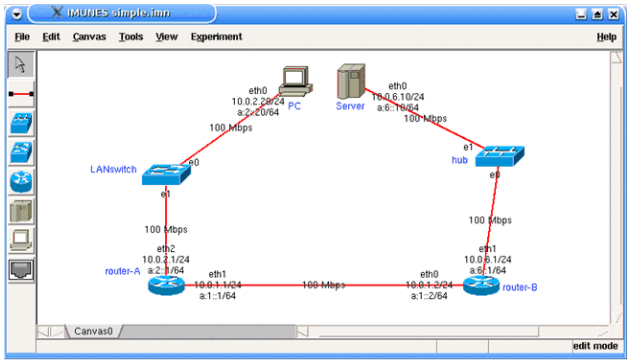
In IMUNES GUI only one link between any two nodes can be created. By default they present Ethernet type links, but support for serial links is also available. All links are presumed to be full duplex. Links are mapped to netgraph nodes as well as link layer nodes. The mapping between the topology and the appropriate kernel structures can be seen on the Figure 3.

Apart from just creating nodes and links in IMUNES, their properties can also be changed through the GUI. In this way we can emulate a network in which some nodes start DNS services and others start web severs, some links have high BER and some links have low bandwidth. The availability of standard network applications in IMUNES nodes and the network emulation provides a way for IMUNES to be used as a replacement for a general purpose testbed.

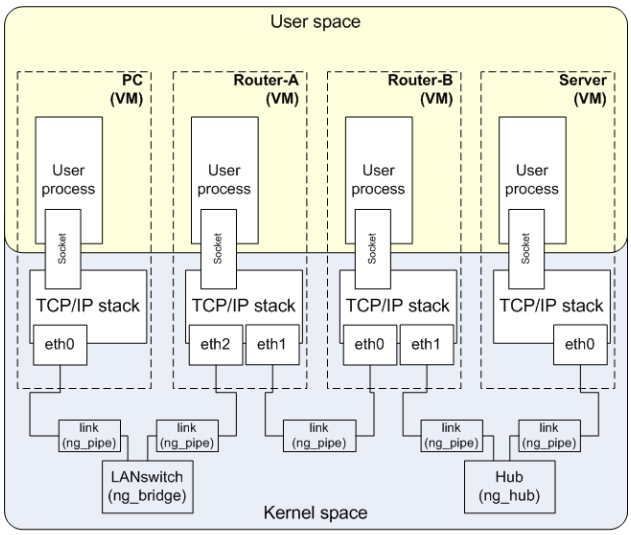
3.3 IMUNES properties

IMUNES is an acronym from an Integrated Multiprotocol Network Emulator/Simulator. Multi-protocol part of the IMUNES name is justified by the support for emulation of IPv4 and IPv6 traffic (in early stages even IPX) and the support for diverse routing protocols including the support for multicast routing using xorp [12].

The high scalability in IMUNES is achieved by using lightweight virtual machine model that does not drain the resources of the hosting machine [14]. The scalability is further improved by passing the packets from one VM to another simply by passing the packet pointer.



a) Topology as specified in IMUNES



b) Topology as deployed in kernel

Figure 3.

The fidelity is increased by the fact that all virtual machines are residing in the kernel, so passing a packet uses real system calls that are processed in only one context switching, the one form a user process to the kernel.

In IMUNES network layer nodes all standard UNIX applications and FreeBSD kernel calls are available, so the process of deploying a new network protocol tested on IMUNES is straightforward and does not require modifications of the code.

IMUNES is developed as open source project under GNU license, and is available form [3].

4. DISTRIBUTED EMULATOR

4.1 Architecture of the distributed system

The architecture of the distributed emulator is presented on the Figure 4. We have divided the IMUNES into a client part and a cluster part of the application. The client host keeps the client part. This part is platform independent since it is written in Tcl/Tk [8]. The experiment deployment is made on the cluster so that the client must know the IP address of at least one host in the cluster, the cluster host he will communicate with.

The cluster part is realized as a peer-to-peer cluster, where any cluster host maintains the constant TCP sessions with all the other hosts in the cluster.

When connecting to the cluster the client sends his user name for authorization. If the authorization passes the user can create a new experiment and view or modify the existing one.

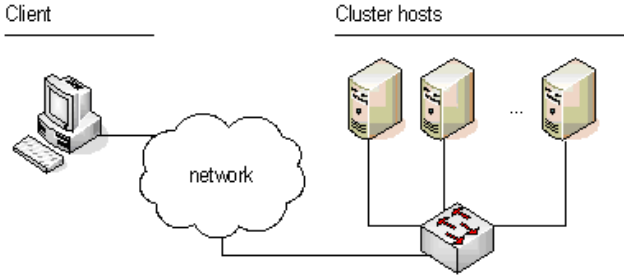


Figure 4. The architecture of the distributed system

The communication between the user and the cluster can be in clear text or encrypted. For the encryption SSH port forwarding mechanism with key authentication is used.

Client and cluster are communicating by sending messages. In most cases the messages represent commands from client to server for controlling the deployed experiment. The experiment in distributed IMUNES represents network topology that belongs to a specific user. User can have more than one experiment deployed at the same time. On the cluster a check is introduced to unable the user to change experiments belonging to others.

4.2 Cluster architecture

In the process of designing the cluster architecture we have considered centralized and a peer-to-peer cluster. While centralized approach would provide simpler management, it has inherent drawback that if the central host goes down the whole cluster goes down as well. So we decided to design our cluster as a peer-to-peer cluster, having two TCP connections between any two hosts in the cluster. This strong connectivity provides an efficient way of early finding that the host is down and it provides the way of monitoring the state of a cluster hosts. Any two cluster hosts are connected by two TCP connections providing a support for a bidirectional asynchrony communication, where each cluster host uses one dedicated connection for issuing the commands and receiving the results and acknowledges.

All hosts in the cluster are equal and all of them have to keep the state of the cluster. The state of the cluster constitutes of:

- List of all hosts available in the cluster and their loads,
- List of all the users in the cluster,
- List of all the VLAN identifiers used,
- List of all the experiments.

Each host in the cluster knows IP addresses and interface names of all the hosts in the cluster. Host additionally keeps the information about his client sockets used for communication with other hosts and his own identifier.

Each user in the cluster has a defined user name, and a list of all experiments belonging to that user. The user name must be unique and one experiment can belong to only one user.

VLAN (IEEE 802.1Q) identifiers are used for a link emulation that goes through the physical link. By using VLAN-s we are restricting the cluster to operate only on local area networks. The number of VLAN-s available on the network is limited, so only a finite number of emulated links can be deployed on top of the physical links. This limitation does not represent a problem since the number of VLAN-s is sufficient for any normal use of the cluster. The VLAN identifiers list keeps the track of all the used VLAN-s. We chose to use VLAN mechanism since adding a VLAN tag is not an expensive action with respect to processor time and memory used and adding a VLAN tag does not cause fragmentation of Ethernet frame.

The information about experiment include the experiment topology, the experiment starting time, the list of hosts on which experiment parts are executed and a list of used VLAN identifiers. Keeping the information about each experiment provides the support for multiexperiment system. The experiment in our distributed emulator presents a new building block.

4.3 Distribution of an experiment

The user experiment could be deployed on only one host in the cluster, but this would not increase the scalability. The idea is to allow the user to create a huge experiment and divide it so that each part of the experiment can be deployed on different host in the cluster. Dividing of topology can be done manually or automatically. For automate division the standard graph partitioning algorithms available through METIS [4] are used. This approach has been used also in some other network emulators [13]. We have constructed a graph from IMUNES topology in order to provide adequate input for METIS. This topology to graph mapping and all the preprocessing and postprocessing steps used are described in [6]. The resulting partitions can be further refined manually.

Partition identifier is assigned to each node in the topology. Links that are connecting nodes in different partitions are marked with local VLAN identifiers. The use of VLAN identifiers is justified by the fact that different experiments can use the same IP address space. Simple passing on to the network the packets as they are sent from originating nodes could cause the interference between different experiments. By using different VLAN identifier for any link connecting nodes simulated on different cluster hosts we have successfully removed any possibility of interference between different experiments.

On the distributed emulator all the nodes are deployed as they would be on normal IMUNES. Only exceptions are the links connecting nodes on different cluster hosts. Packets traversing this type of links are really going through the physical link that connects cluster hosts. The problem that occurs is how to simulate the properties of that link, so we used emulated links for traffic shaping on both cluster hosts. The control of the emulated properties, like bandwidth, is enforced before the packet leaves the cluster host and goes to the network. The delay is uniformly distributed on both

emulated links. Shaping of the links connecting nodes deployed on different cluster hosts is presented on Figure 5.

4.4 Properties of the distributed emulator

1) Scalability

The scalability of distributed IMUNES proportionally increases with the number of hosts in the cluster. By distributing the topology over the cluster we are using the resources of all cluster hosts. In the distributed emulation the number of kernel structures created remains the same as it would be if the emulation was executes on one machine. The only exceptions are links that connect nodes employed in different cluster hosts, for this type of links we use two kernel structures instead of just one.

The resources necessary for managing the cluster will increase with the number of hosts in the cluster, but we do not expect this rise to affect the emulation capabilities of the cluster since the cluster management utility is designed as a simple, resource undemanding application.

As a result we expect for the emulation to scale accordingly to the number of hosts in the cluster and their cumulative resources.

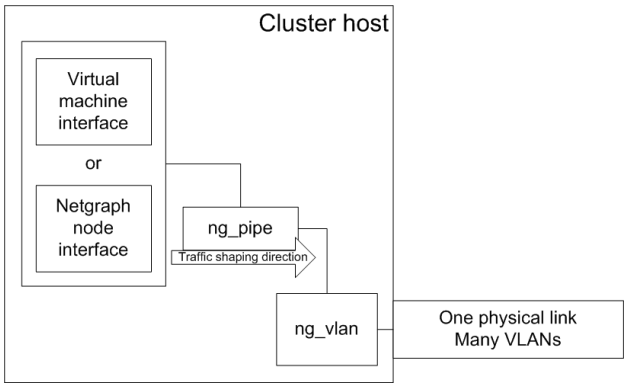


Figure 5. The traffic shaping direction of links that connect nodes deployed on different cluster hosts.

The limit on scalability still remains since we use VLAN identifiers that limit the cluster to be deployed only on LAN. But viewed from other perspective the employment of the cluster over the LAN provides faster links that increase the emulation capacity for the links that connect virtual nodes on different cluster hosts. It also provides easier maintenance of the cluster.

2) Robustness

We have achieved robustness by introducing redundancy in the system. The state of the cluster is maintained on each host in the cluster, so when one host in the cluster goes down the whole cluster doesn't suffer the consequences. Since the state of the cluster keeps the information of all the executed experiments, after the host fails the rest of the cluster knows which experiments, or parts of experiments are lost.

In the future we could offer the possibility of reassigning the parts of the topology that ware executed on a host that went down to another host or hosts in the cluster. This doesn't mean restoring the state of the experiment in the time of the failure; it just means restoring the state to the initial experiment state for that part of the experiment.

3) Utilization

When we talk about utilization we are primary thinking on improving the usability of available resources. So we designed a multiuser and multiexperiment support. Each user can access the cluster for execution of his experiments. The multiexperiment support means that different users can have different experiments (network topologies) and that one user can have many experiments. All this experiments can run at the same time on the cluster. Utilization of the cluster is further increased by the separation of client and cluster utilities. Client side can reside on the same machine as the cluster, but it can also be physically placed anywhere as long as it has a network connection to the cluster.

4) Portability

The portability is increased by separating client part and a cluster part of our distributed emulator. The client part is platform independent and provides an opportunity for users that do not work on modified FreeBSD kernel to use IMUNES emulation capabilities simply by accessing the cluster. The portability of IMUNES is currently increased by the use of VMware images and bootable CD.

5) Performance

We are trying to keep the performance of this distributed emulator in the same range as for IMUNES: This means providing the same scalability on the cluster hosts, and accurately emulating all the links that are going through the real links. The scalability is already discussed, so let us focus on the accurate emulation of links that connect nodes emulated on different cluster hosts.

Links connecting nodes residing on different cluster hosts are modeled for keeping the performance. The bandwidth check is performed before traffic reaches the physical link assuring that all the links that are using the same physical link receive the promised amount of bandwidth. The compensation for passing the emulated traffic through a real network is part of the future work and that's way we base our cluster on LAN.

6) Availability

Compared to centralized management peer-to-peer cluster offers higher availability. The four aspects of availability are improved. First, the user can connect to any host in the cluster to execute his experiments. Second, for a cluster to fail all the hosts in the cluster must fail, so the availability of the cluster is higher than the availability of just one host. Third, all the resources of all the hosts in the cluster are available for use. Separating client from the cluster is the last aspect of higher availability; the cluster is available to the user even when user works on different operating system.

5. RESULTS AND CONCLUSION

By implementing and testing the prototype of distributed emulator based on IMUNES we have received the following results. We have tested the cluster by creating one large experiment with N nodes that was deployed on one machine and on the cluster with two machines. The obtained results are presented on Figure 6. From graph we can see that on the two machine cluster the time for deployment of the experiment

reduces by the factor of two. This factor of time reduction is proportional to the number of cluster hosts. The deployment time fluctuations visible on graph are the result of the deployment of the cluster over two machines that ware not connected on the dedicated network, so the other traffic was also present. These fluctuations have the value around one second. The improvement of scalability is also visible, since on one machine we could employ 1024 IMUNES nodes, whereas on the cluster we have deployed 2048 IMUNES nodes.

In the table presented on Figure 7. we see the results of ping tested on live network, on standard IMUNES and on our distributed emulator. As we can see from the results the distribution of the experiment does not reduce the performance of IMUNES.

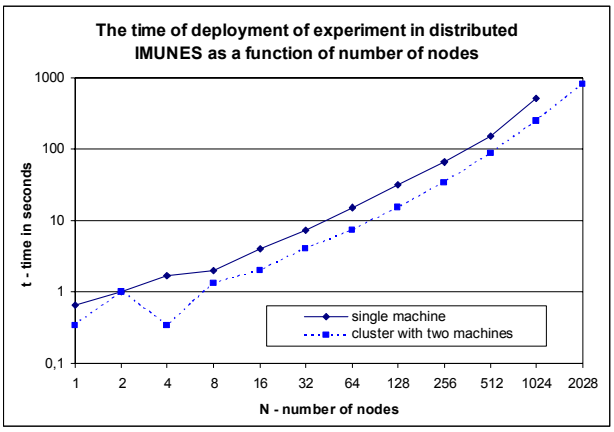


Figure 6. Test of distributed emulator scalability and performance

	Packet size	
	64KB	1480KB
Testbed	0,278ms	1,302ms
IMUNES	0,252ms	1,284ms
Distributed emulator	0,253ms	1,287ms

Figure 7. The results of Ping

All tests ware conducted on the commodity PCs, one with Celeron operating on 3GHz with 1GB of RAM and the other was Pentium P4 on 3,2GHz with 2GB of RAM.

The described concept of distributed network emulator is designed to satisfy the demands of scalability, availability, robustness, efficiency, portability and performance. This work is the work in progress any many things are currently hard coded, but we believe that in the future this concept will provide a stable, manageable and scalable distributed emulator.

6. FUTURE WORK

The future development would primarily deal with the user management since this part is currently hard coded. The cluster we developed and tested is a static cluster with hard coded members, but in the future we expect to develop a new version of the cluster that would support dynamical joining and leaving of the cluster hosts. We plan to implement additional support to provide an option for a cluster to operate on geographically distributed locations and in this way to further improve scalability.

REFERENCES

[1] Bavier, A., Browman, M., Chun, B., Culler, D., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., Wawrzoniak, M. *Operating System Support for Planetary-Scale Network Services*, *Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, 2004.

[2] FreeBSD jail
http://docs.freebsd.org/44doc/papers/jail/jail.html

[3] IMUNES download site
http://imunes.net

[4] Karypis, G., Kumar, V. *A fast and high quality multilevel scheme for partitioning irregular graphs*. *SIAM Jurnal of Scientific Computing,* 20(1):359-392, 1998.

[5] Netgraph framework
http://people.freebsd.org/~julian/netgraph.html

[6] Puljiz, Z., Mikuc. M. *Clustering Network Simulation: Graph Partitioning approach*, Proc. of ConTEL, 2005.

[7] Quagga
http://www.quagga.net/

[8] Tcl/Tk programming language
http://www.tcl.tk

[9] Vahdat, A., Yocum, K., Walsh, K. et al. *Scalability and Accuracy in a Large-Scale Network Emulator*. Proc. of OSDI, 2002.

[10] VMware
http://www.vmware.com

[11] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A. *An Integrated Experimental Environment for Distributed Systems and Network*s

[12] XORP project
http://www.xorp.org/

[13] Yocum K., Eade, E., Degesys, J., Becker, D., Chase, J., Vahdat, A. *Toward Scaling Network Emulation using Topology Partitioning*. in *Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, (MASCOTS)*. 2003.

[14] M. Zec, M., Mikuc, *Operating System Support for Integrated Network Emulation in IMUNES, Proceedings of the 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure* / ASPLOS-XI, 2004.