Design of a Real Time Transaction Processing Monitor (TPM) Benchmark Testbed

MARIA LUISA CATALAN, DENNIS A. LUDENA R., HIDENORI HUMENO, MASAYOSHI ARITSUGI
Graduate School of Science and Technology
Kumamoto University
39-1, Kurokami 2-chome, Kumamoto City 860-8555
JAPAN
{maru123, dennis}@st.cs.kumamoto-u.ac.jp
aritsugi@cs.kumamoto-u.ac.jp

Abstract: - The Transaction Processing Monitor (TPM) is the most-used middleware in different e-commerce systems from large enterprises to medium and small businesses available in the internet. Due to its growing popularity, the necessity for a more efficient TPM performance is now the major concern between the developers and researchers. The need for a high-end benchmark platform for a TPM is at present very vital to meet the high performance needs of online transactions. In addition to the performance characteristics of the TPM, we also have to ensure the security of the transactions. In this paper, we perform a detailed analysis of the current software packages available for this application. And therefore, we propose a secure, isolated, and highly configurable environment using the real-time emulation capabilities of NS2 and the virtualization capabilities of Xen, in order to provide a testbed with the characteristics and behavior of a real network.

Key-Words: - Virtualization, Emulation, Networking, Transaction Processing, Benchmark, Modelling

1 Introduction

The TPM (Transaction Processing Monitor) is at present the most used middleware for online transactions processing systems. Due to the great increase of using online transactions around the world, the need to develop a more efficient TPM system is becoming an enormous concern among developers.

TPM handles thousands of transactions coming from different sources and geographical locations, all trying to get the desired information without errors in the shortest amount of time. For this reason, an urgent need for a new and efficient middleware system that can perform and manage the task more efficiently is now in great demand [1, 2].

However, the required budget to develop a test platform is high, and the flexibility is limited [3]. In addition, most of the variables are controlled, which makes the platform too much predictive in its behavior.

Back in our first approach (shown in Figure 1), we determined the best environment to realize the creation of a test platform based on virtualization technology. The main parts of this system are:

- PC which will handle the Database
- PC which will handle the TPM system
- Virtualized users

For the virtualized users, we use Xen which has special kernel that allows mainly only two types of virtualization: The Paravirtualization and Fully Virtualization [7, 8].

The virtual users can send Ethernet frames to and from the simulation. This approach has the advantage of using real TCP/IP stack. In the case of Xen's paravirtualization feature, we have a direct access to the hardware [9, 10]. But the major limitation that we faced was the required memory for each user.

This condition did not allow us to create more users in order to benchmark the system. Figure 1 shows the architecture of our previous study, the three PC Linux-based systems.

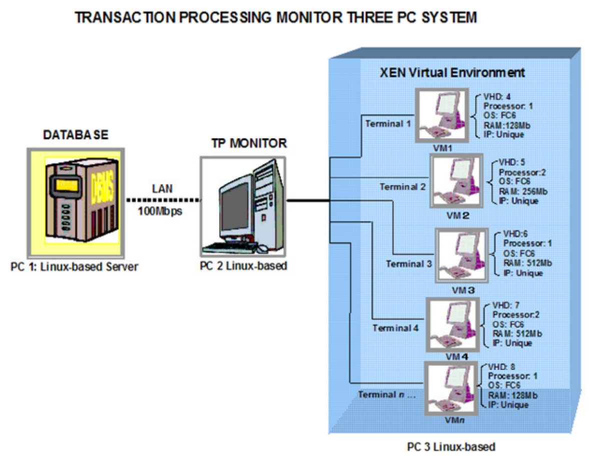

Figure 1. Desired architecture [1]

As for the network emulation tool, we are going to use NS2. NS2 is very well known among developers and researchers around the world because of its capabilities to test new network protocols.

2 TP (Transaction Processing) Monitor

A TP monitor is a complex middleware program designed to manage the execution of a transaction. In the event a client initiates a transaction, the TP monitor sends the transaction to the database depending on the type of request and sends back a response.

The core concept is a transaction which strictly considers the ACID requirements [11].

An acronym which stands for Atomicity (all transactions are either completely committed, or are not done at all), Consistency (the transactions transforms into a new correct state), Isolation, (the series of transaction stages must not be visible to other transactions), and Durability (once the transaction commits, the results should be preserved despite any failures).

The types of jobs performed through the TP monitor are: process management, transaction management, and client/server communication management.

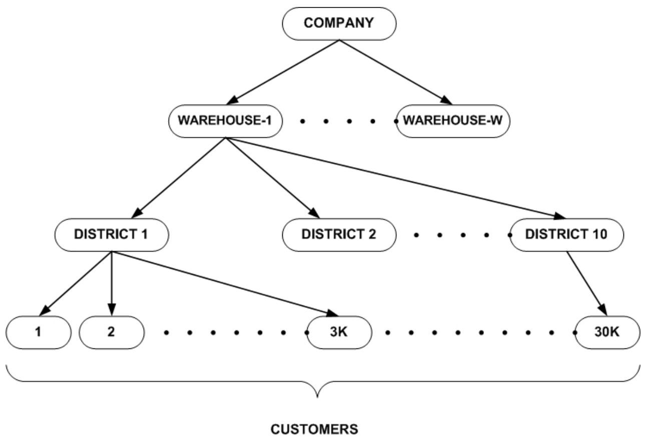

Figure 2. Hierarchy of the TPC Business environment [11].

2.1 The TPC-C

TPC-C stands for Transaction Processing Performance Council which is an on-line transaction processing (OLTP) benchmark. TPC-C is more complex than previous OLTP benchmarks, such as TPC-A because of its multiple transaction types, more complex database and overall execution structure. TPC-C involves a combination of five concurrent transactions of different types and complexity, either executed on-line or queued for deferred execution.

The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC).

The characteristics of the TPC-C are:

- The simultaneous execution of multiple transaction types that span a breadth of complexity
- On-line and deferred transaction execution modes
- Multiple on-line terminal sessions
- Moderate system and application execution time
- Significant disk input/output
- Transaction integrity (ACID properties)
- Non-uniform distribution of data access through primary and secondary keys
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

Figure 2 shows the general model of the usage of TPC-C system.

3 Virtualization Technologies

Virtualization is now a hot research topic for both servers and desktops due to its standard software application wherein multiple virtual machines can be held on a host PC [12]. Through virtualization, experiments with routers and networks also become possible, as well as between the host computer and the virtual laboratory without risking the real environment [13].

Here, we provide a high level overview of the most popular methods of virtualization [14], the Full virtualization and the Paravirtualization.

3.1 Full Virtualization

Fully virtualization is the most popular method supported by VMWare and Microsoft (Virtual PC). For desktops, this means running the virtualized operating system over a fully installed operating system like Windows or Linux. Inside the virtual machine application, a user is isolated and can create different virtual machine configurations (guests) where each of the guests can have their own virtual devices, including drivers, hardware and peripherals.

Users can install an operating system and applications into each of the virtual machines created inside the full virtualization solution. Administering one of the guests is just like administering a single operating system.

3.2 Paravirtualization

Paravirtualization solutions, like the Xen Virtual Machine Monitor (VMM) requires a special operating system installation [14]. In Linux, this is simply a customized kernel and management software piece. This method, like full virtualization provides secure isolation between the virtual machines. Xen requires the installation of a Xen-capable Linux kernel that can act as the control operating system. This controls the paravirtualization layer that resides between physical devices and guest operating systems.

The main technical difference between this method and the full virtualization comes in the paravirtualization layer, which connects the I/O device between different guests and provides direct driver access to the guests.

Paravirtualization layers can provide access to direct hardware resources, while full virtualization solutions provide access to virtualized device drivers only, thus lacks the support to some of the latest hardware features.

Paravirtualization can also be used to provide device access to operating systems that the native drivers might not have for these devices available. Each of the guest operating system installations are full Linux installations with their own devices, file and storage requirements, etc.

Table 1 shows a comparison of the different virtualization methods.

	Method 1: Full Virtualization	**Method 2: Paravirtualization**
Benefits	• Works with existing operating system (OS) installations • Unmodified guest OS's • Full isolation between guest OS's	• Access to direct devices and resources (USB2) • Open Source solutions can be modified by anyone • Low price • Better performance compared to full virtualization • Full isolation among guests
Constraints	•Proprietary Solutions controlled by the Virtualization companies •Price •No direct device driver access	• Requires modified host and guest OS's • Not as fully supported and developed as the commercial solutions

Table 1. Virtualization technology comparison summary [15]

Name	Guest OS(s)	Guest OS SMP available	Drivers supported guest OS available	Guest OS speed relative to host OS	License
HyperV	Supported drivers for Windows 2000, Windows 2003, Windows 2008, Windows XP, Windows Vista, Linux (SUSE 10 Released, more announced)	Yes	Yes	Native drivers IO is non-emulated for better IO performance. However, substantial performance loss on some workload (network and and disk intensive especially).	Proprietary (Free with Windows Server 2008)
User Mode Linux	Linux	?	Special guest kernel + modules required	Near native (Runs slow as call calls are proxied)	GPL Version 2
Virtual PC 2007	DOS, Windows, OS/2, Linux (SUSE, Xubuntu), Open Solaris (Belenix)	No	Yes	Near native with Virtual Machines additions	Proprietary (Free from Jul 2006)
Virtuozzo	Various Linux Distributions, Windows	Yes	Compatible	Native	Proprietary
VMware Server	DOS, Windows, Linux FreeBSD, Netware, Solaris, Virtual Appliances	Yes (2 – way)	Yes	Up to near native, substantial performance loss on some workload (network or disk intensive specially)	Proprietary (Free)
VMware Workstation 6.0	DOS, Windows, Linux FreeBSD, Netware, Solaris, Darwin, Virtual Appliances	Yes (2 – way)	Yes	Up to near native	Proprietary
VMware Player 2.0	DOS, Windows, Linux FreeBSD, Netware, Solaris, Virtual Appliances	Yes (2 – way)	Yes	Up to near native, substantial performance loss on some workload (network or disk intensive specially)	Proprietary (Free)
Xen	Linux, Solaris, Windows XP & 2003 server (needs vers. 3.0 and a Vanderpool or Pacifica–capable CPU), Plan 9	Yes	Not required with the exemption of the networking drivers where a NAT is required. A modified guest kernel or special hardware level abstraction is required for guest OSs.	Up to near native speed, substantial performance loss on some workload (network and disk intensive specially)	GPL

Table 2. Virtual Systems Comparison[18, 19]

	Customizable	Modularity	Friendly GUI	Scalability
NS-2	Yes	No	No information available	Yes
PDNS	Yes	Yes	Yes	Yes
OPNET	No information available	No	No information available	Yes
NetSim	Yes	No	No information available	No information available
GTNETS	Yes	No	No information available	No information available
CNet	No information available	No	No information available	No information available
Simnet	No information available	No	No information available	No information available

Table 3. Comparative chart among the network simulators

3.3 Xen System

Here, Xen 3.0 is used because of its high end features, Linux friendly environment, and high VMM performance [7]. In addition, it has distinctive characteristics in relation to paravirtualization management like, better performance compared to the fully virtualization environment such as; easy facility for storage area needed during the installation process, and the capability to work in an independent operating system and network [16]. In this work, we use CentOS 5.0 Final as a host system.

One of the advantages of using a virtualized environment is its easy management during configuration. In a real multi-user environment, manual work is necessary to change the configuration of terminals.

With virtualization, we can create and store groups of virtual machines or virtual environments in different storage devices like: RAID Hard Disk, SATA/IDE Hard Disk, or DVD, in less time and effort.

Table 2 [18, 19] provides a comparison of the different characteristics of Xen against some of the most popular virtualization packages. The following are some of the applications where these packages are used: Hobbyist, Developer, Business, Enterprise server consolidation, Hosting service separation, Security Isolation, Research, Tester, etc.

Moreover, Xen is capable of creating a small network inside the host PC. In this network can be used: routers, switches, and different network devices. This capability is vital for our purposes because we need to create several scenarios, where various network topologies are included.

As we can observe in Table 2, Xen contains several advantages compared with some of the popular virtualization packages in the market.

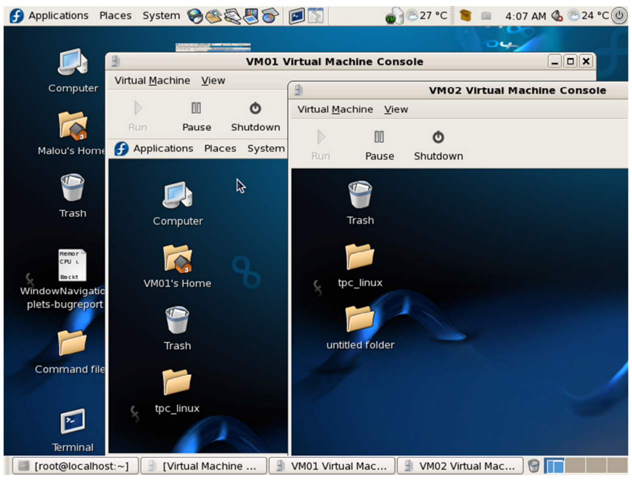

Figure 3. Example of virtualized users [1]

4 Network Emulation

As we consider in the previous study [1], we need to create an emulated network environment where the packages created in several terminals will be transported in this network.

This emulated network will have the following characteristics:

4.1 Customizable network environment

The system is capable of creating several small, medium and large network topologies, like: wireless, ATM, Ethernet, etc.

4.2 Capability to receive traffics generated from another real or virtual terminals

The system is capable of receiving traffics generated from other terminals. These terminals could be real PCs or from a virtual environment. In our particular case, we create a SOHO (Small Office Home Office) Network. The traffic from this network will be applied to the simulated network.

4.3 User Friendly GUI

The system has the basic GUI features so that we will be able to observe the results in a graphical way. And also, perform some configurations through the GUI.

4.4 Scalability

The system has the capability to increase the number of nodes or modify the topologies in the network without difficult or complicated procedures. There are several packages in the market that can meet the requirements. In order to select the most suitable package for our purposes, we make an extensive study comparing the main characteristics of these systems.

Analyzing the Table 3 results, we can observe that PDNS (Parallel Distributed NS) could be the best alternative for our purposes. But the problem in this case is the support of this network emulated software. PDNS was developed by the College of Computing, Georgia Institute of Technology [22], and the last modification or update was made in March 2004. So we can conclude that the system is not update or at least that the update process was stopped for any reason, which makes the system not reliable for new kernel applications. In this case in order to keep the system as economic as possible, we decide to use NS2. NS2 is the widest network emulation package. It has an official support from the developers. The GUI is not as rich as other packages but there are some third party applications that offer more interesting GUI for NS2.

5 The NS2 Network Emulator

NS is a discrete event simulator directed to the networking research. NS provides several simulation environments like: TCP/IP, wireless, and different tools to test protocols under research [5, 6]. NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI [4].

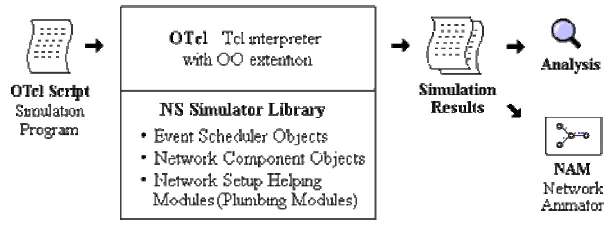
OTcl Script
Simulation Program
OTcl Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules(Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Figure 4. NS2 Architecture [6]

As shown in Figure 3, in a simplified user's view, NS is an Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup. In other words, to use NS, the programming is done in OTcl script language.

To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library. Then, tells traffic sources when to start and stop transmitting packets through the event scheduler.

Because of the discrete event nature of NS2, the scheduler become the vital part of the system. Through this system, we will be able to know exactly when to start a service or procedure, and when to stop them. Through the tracking of the simulation times, the scheduler is capable to activate the events that were previously defined in the OTcl file.

The event scheduling also allows us to generate several channel characteristics pre-defined in the OTcl script, like: delays, errors, packet loss and more. This feature is very relevant to our development purposes, since the TPM is a TCP/IP client server application. We need to find and develop the most "close to the real" environment. Through this, the tested events will have all the realistic design considerations.

Figure 5 shows the NS2 architecture. Depending on the user level, the simplest one could stand in the bottom corner where it can run OTcl based applications. While, an advanced user can exploit some of the C++ / OTcl capabilities to generate complex simulations. All the components together make the NS2. In other words, NS is an extended Tcl interpreter with network simulator libraries.

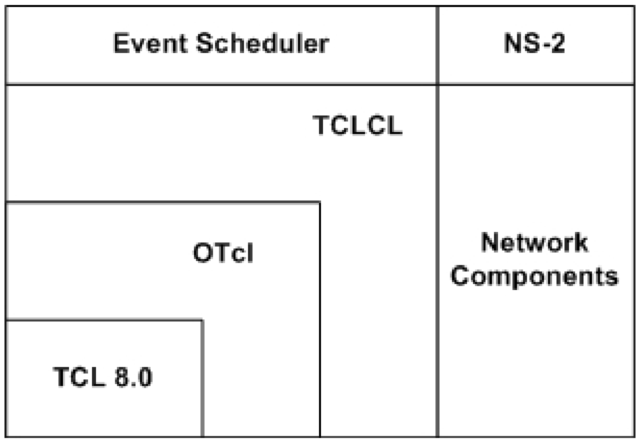

Figure 5. NS2 Architecture [6]

6 Real time applications in NS2

As explained before, one of the most important features of NS2 is the capability to generate traffic through the scheduler, which we can define in the OTcl script. In the case of real time applications we should use the "real time scheduler" in the first line of our OTcl script, chart 1.

```
set ns [new Simulator]
$ns use-scheduler RealTime
```

Chart 1. Declaration of the real time scheduler in NS2 [6, 23, 24]

Objects including tap agents and network objects are the interfaces between the simulator and the real network traffic [5]. Tap agents are in charge of embedded real-time data into simulated packets and vice-versa. The Sending and Reception in the real time data are the ones in charge of the Network objects, which are installed in the Tap Agents [6].

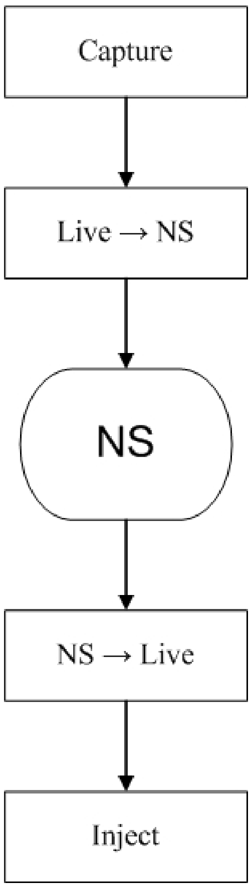

Figure 6. Interaction between the different parts of the real-time emulation in NS2 [6, 23, 24]

In NS2 emulation capabilities are divided into:

6.1 Opaque mode

In Opaque mode, the data from the real time network is treated as black data packets. And, the simulator treats network data as uninterpreted data.

In particular, real-world protocol fields are not directly manipulated by the simulator.

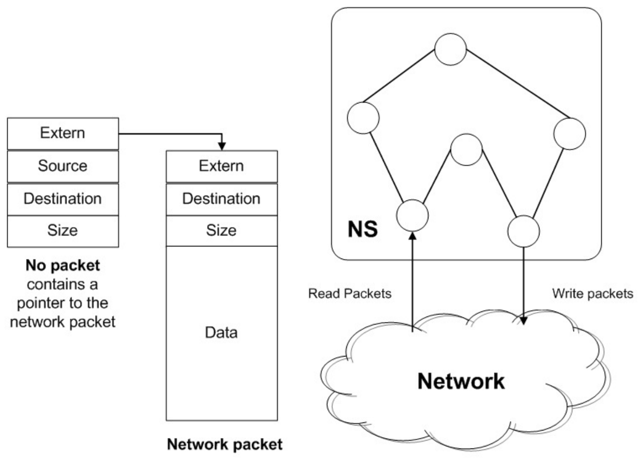

Figure 7. Opaque mode Emulation [6, 23, 24]

In opaque mode, live data packets may be dropped, delayed, re-ordered, or duplicated. But because no protocol processing is performed, protocol-specific traffic manipulation scenarios (e.g. ``drop the TCP segment containing a retransmission of sequence number 2045'') may not be performed.

In protocol mode, the simulator is able to interpret and/or generate live network traffic containing arbitrary field assignments.

6.2 Protocol mode

In Protocol Mode, the real time data may be interpreted/generated by the simulator. In order to connect the real time network with the simulator we use a series of objects called tap agents and Network objects. Figure 9 shows how they interact.

We mentioned that NS2 is an event scheduler based emulation package, which means that we have to declare the sequence events in the time. In order to inject real time data to the emulation, we have to use a modified or parallel version of this scheduler, called Real Time scheduler.

Real Time Scheduler

The Real-time scheduler ties event execution within the simulator to real time. It is necessary to have PC resources available to keep up with arriving packets, and the simulator virtual time should closely track real time.

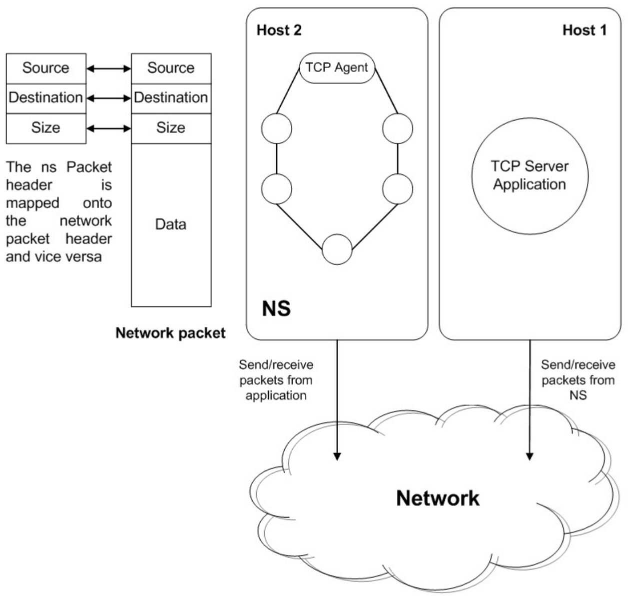
Source
Destination
Size
Source
Destination
Size
Data
The ns Packet header is mapped onto the network packet header and vice versa
Network packet
Host 2
TCP Agent
NS
Host 1
TCP Server Application
Send/receive packets from application
Send/receive packets from NS
Network

Figure 8. Protocol mode Emulation [6, 23, 24]

If the simulator becomes too slow to keep up with elapsing real time, a warning is continually produced if the skew exceeds a pre-specified constant ``slop factor'' (currently 10ms).

Tap Agents

The class TapAgent is a simple class derived from the base Agent class. As such, it is able to generate simulator packets containing arbitrarily-assigned values within the ns common header.

The tap agent handles the setting of the common header packet size field and the type field. It uses the packet type PT_LIVE for packets injected into the simulator. Each tap agent can have at most one associated network object, although more than one tap agent may be instantiated on a single simulator node [23, 24].

Network Objects

The Network objects are in charge of giving access to a live network (or to trace file captured network packages). There are several forms of network objects depending on the protocol layer specified for access to the underlying network, in addition to the facilities provided by the host operating system. Generally, network objects provide an entrypoint into the live network at a particular protocol layer (e.g. link, raw IP, UDP, etc) and with a particular access mode (read-only, write-only, or read-write). Some network objects provide specialized facilities, such as filtering or promiscuous access (i.e. the pcap/bpf network object) or group membership (i.e. UDP/IP multicast).

The C++ class Network is provided as a base class from which specific network objects are derived. Three network objects are currently supported: pcap/bpf, raw IP, and UDP/IP.

7 Benchmark architecture

We already analyze the capabilities of NS2 and Xen. The proposed architecture is to combine both of the above mentioned architectures in order to present a dynamic platform for testing TPM.

Figure 9 shows the architecture of the proposed platform.

As shown in the figure, there are several “virtual users”. We are going to follow the original architecture (Figure 1) in the next implementation step.

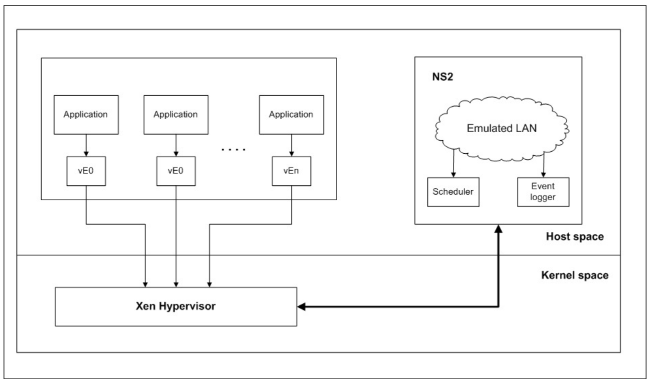

Figure 9. Proposed TPM Benchmark Testbed

Some of the characteristics of the following architecture will be:

7.1 Scalability

The system could increase the number of “virtual users” that can send transaction requests to the TPM system.

7.2 Isolation

The system is completely independent from the university network, which allows several experiments without interference from the normal traffic of the network.

7.3 High simulation capabilities

Using some of the features of NS2, the system is capable of generating the following: errors in the network, package drops, several channel conditions, etc. which allow the system to deal with a more realistic environment.

7.4 Configurability

Exploiting the combined high capabilities of Xen virtualization and the NS2 Emulation, the system will have a high configurability grade. The system could generate numerous scenarios where the TPM can be tested, therefore, providing the researchers with a wide experimentation space to test research studies and developments.

Also, this system can be integrated into an educative platform, like Learning by Development (LbD) [20]. Regardless of the specific application shown, the system can be configured to test another network based e-Learning systems [21. 25].

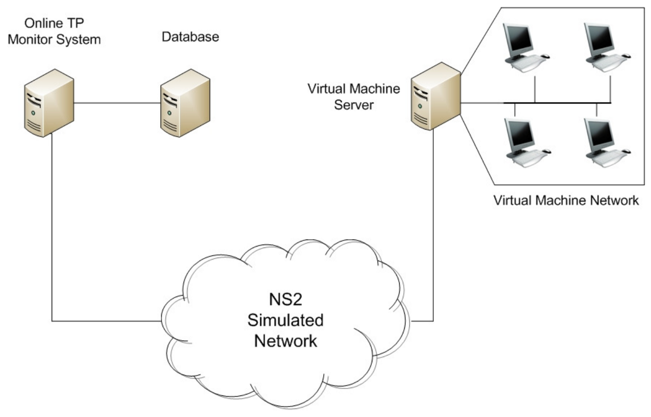

Figure 10. General view of the complete system.

8 Conclusion

In this paper, we introduce the initial configuration of a real-time benchmark testbed. The use of the emulation capabilities of NSE, will allow us to inject real traffics from the "virtual users" or real users (with the use of a tunneling software).

The proposed architecture will allow us to test not only the TPM system, but also specific client server applications based on the TCP/IP protocols, and also other applications based on standard communications protocols.

Our future work will be focused on the design details, performance tests and analysis of the simulations.

Acknowledgment

We dedicate this paper in the memory of Professor Hidenori Umeno, for his teachings, unconditional support and kind advice.

References:

[1] Catalan, Maria Luisa, Ludena, Dennis, Umeno, H., A dynamic network simulator for testing TP monitor system performance and behavior, *Proceedings of the International Conference on Control, Automation and Systems, ICCAS*, 2007, pp. 1851 – 1854.

[2] C. Edwards, A. Harwood and E. Tanin, Network Virtualisation for Transparent Testing and Experimentation of Distributed Applications, *Proceedings of the 2005 13th IEEE International Conference on Networks*, Volume 2, 2005, pages 1089- 1094

[3] Breslau, L.; Estrin, D.; Fall, K.; Floyd, S.; Heidemann, J.; Helmy, A.; Huang, P.; McCanne, S.; Varadhan, K.; Ya Xu; Haobo Yu; Advances in Network Simulation, *Computer*, Volume 33, Pages 59-67, 2000

[4] Jansen, S. and McGregor, A., Performance, Validation and Testing with the Network Simulation Cradle, *Proceedings of 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2006*, pages 355-362, 2006

[5] Kevin Fall, Kannan Varadhan, The ns Manual, The VINT Project, 2008

[6] http://www.isi.edu/nsnam/ns/

[7] W. Huang, J. Liu, B. Abali and Dhabaleswar K. Panda, *A case for High Performance Computing with Virtual Machines*, International Conference on Supercomputing, Proc. of the 20th Annual International Conference on Supercomputing, Cairns, Queensland, Australia, pp: 125-134, 2006

[8] R. Haukioja and N. Dunbar, *Introduction to Linux Virtualization Solutions*, Hewlett-Packard Development Company, L. P. 2007 available at http://opensource.hp.com/techbriefs/haukioja_dunbar.html

[9] David Chisnall, *The Definitive Guide to the Xen Hypervisor*, Prentice Hall, 2008

[10] David E. Williams, Juan Garcia, V*irtualization with Xen*, Syngress, 2007

[11] C. Browne, *Transaction Processing Monitors*, available at http://cbbrowne.com/info/tpmonitor.html

[12] Ji Hu, Dirk Cordel and Christoph Meinel, *A Virtual Laboratory for IT Security Education*, presented at the Int. Conference on Information Systems in E-Business and EGovernment (EMISA), Luxembourg, Oct. 2004, pp: 60–71.

[13] J. Nieh and C. Vaill, *Experiences Teaching Operative Systems Using Virtual Platforms and Linux*, presented at ACM SIGOPS Operating Systems Review, Volume 40, Issue 2, April 2006, pp: 100-104.

[14] K. Begnum, J. Sechrest and S. Jenkins, *Getting more from your Virtual Machine*, presented at Journal of Computing Sciences in Colleges, Volume 22 , Issue 2, December 2006, pp: 66 – 73.

[15] Open Source and Linux from HP:HP TechBrief- Introduction to Linux Virtualization. http//www.hp.com/cgi-bin/pfnew.cgi?IN=http://opensource.hp.com/techbriefs/haukio…

[16] W. Huang, J. Liu, B. Abali and Dhabaleswar K. Panda, *A case for High Performance Computing*

with Virtual Machines, presented at International Conference on Supercomputing, Proc. of the 20th Annual International Conference on Supercomputing, Cairns, Queensland, Australia 2006, pp: 125-134.

[17] W. I. Bullers Jr, S. Burd and A. F. Seazzu, *Virtual Machines-An Idea Whose Time Has Returned: Application to Network, Security, and Database Courses*, presented at Technical Symposium on Computer Science Education, Proc. of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, Texas, USA 2006, pp: 102-106.

[18] Comparison of virtual machines, available: http://en.wikipedia.org/wiki/Comparison_of_virtual_machines

[19] Stephen Soltesz, Herbert Pötzl, Marc E. Fiuczynski, Andy Bavier, and Larry Peterson, *Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors*. ACM SIGOPS Operating Systems Review archive Volume 41 , Issue 3 (June 2007), pp: 275 – 287. 2007

[20] Jyri Rajamaki and Rauno Pirinen. *Linkage of Learning by Developing and Virtual Learning Case: Network Design Specialisation Studies*. 2nd WSEAS European Computing Conference (ECC'08), Malta, pp: 397 – 402. 2008

[21] F. de Arriaga, C. Gingell, A. Aarriaga, J. Arriaga, and F. Arriaga Jr. *A General Student's Model Suitable for Intelligent E-Learning Systems*. 2nd WSEAS European Computing Conference (ECC'08), Malta, pp: 167 – 172. 2008.

[22] PDNS - Parallel/Distributed NS, available on: http://www.cc.gatech.edu/computing/compass/pdns/index.html

[23] Kevin Fall and Kannan Varadhan. The ns Manual. The VINT Project, 2008

[24] Richard M. Fujimoto, Kalyan S. Perumalla, and George F. Riley. *Network Simulation*. Published by Morgan & Claypool Publishers, 2007

[25] Krzysztof Tokarz and Piotr Jedrychowski. *Control Application for eLearning system*. 2nd WSEAS European Computing Conference (ECC'08), Malta, pp: 234 – 239. 2008.