Transaction Processing Monitor (TPM) Real-time Benchmark Testbed

MARIA LUISA CATALAN, DENNIS A. LUDENA R., HIDENORI UMENO, MASAYOSHI ARITSUGI
Graduate School of Science and Technology
Kumamoto University
39-1, Kurokami 2-chome, Kumamoto City 860-8555
JAPAN

Abstract: The Transaction Processing Monitor (TPM) is the standard middleware used in different e-commerce systems from large enterprises to medium and small businesses available in the internet. Because of its significantly growing popularity, the necessity for a more efficient TPM performance is now becoming a major concern between developers and researchers. Due to this concern, the need for a high-end benchmark platform for TPM is at present very vital to meet the high performance needs of online transactions. In this paper, we propose an 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

1 Introduction
The TPM (Transaction Processing Monitor) is at present the most used middleware for online transaction processing systems. Due to the greatly increasing usage of online transactions around the world, the need to develop a highly efficient TPM system is becoming an enormous concern among developers and researchers. TPM handles thousands of transactions everyday 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
- Several virtualized users that will perform transaction requests to the TPM

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 of the users.

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 system. The virtualized users can be observed in figure 2.

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

In this paper, we explore some of the fundamental characteristics of NS2 for our research and
development purposes, and how we utilize and integrate it to our project.

Figure 1. Desired architecture

Figure 2. Example of Virtualized Users

2 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].

Figure 3. NS2 Architecture

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, network component object libraries, and network setup. In other words, to use NS, programming should be 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 the NS2, the scheduler became 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 4 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 on the other hand, advanced users 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.
3 Real time applications in NS2
As explained before, one of the most important features of NS2 is the capability to generate traffic in 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

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 embedding 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 [NS2 manual].

In NS2, emulation capabilities are divided into:

1) *Opaque mode*, where the data from the real time network is treated as black data packets.

2) *Protocol mode*, where the real time data may be interpreted/generated by the simulator.

4 Benchmark architecture
We already analyzed 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 to test the TPM.

Figure 5 shows the architecture of the proposed platform.

```
Figure 5. TPMLTest platform
```

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

Some of the characteristics of the following architecture will be, but not limited to:

4.1 Scalability
The system could increase the number of “virtual users” that execute transaction requests to the TPM system.

4.2 Isolation
The system will be completely independent from the university network, which allows unlimited experiments without interference from the normal traffic of the network.

4.3 High simulation capabilities
Using some of the features of NS2, the system will be 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.

4.4 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.

5 Conclusion
In this paper, we introduce the initial configuration of a real-time benchmark testbed. The use of the emulation capabilities of NS2 (under development), 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. Also, other applications based on standard communications protocols.

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

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

References:


