Provisioning Service Differentiation for Virtualized Network Devices

Suk Kyu Lee, Hwangnam Kim, Jun-gyu Ahn, Kwang Jae Sung, and Jinwoo Park
School of Electrical Engineering
Korea University, Seoul, Republic of Korea
Email: {sklee25, hnkim, dubhe, kjsung80, jwpark}@korea.ac.kr

***Abstract*— In order to efficiently utilize the network bandwidth and flexibly enable one or more networks to be combined or subdivided into virtual networks, it is essential to virtualize network devices and then to provide service differentiation for the virtualized network devices. In this paper, we propose a virtualizing method for network devices based on the virtual machine and offers a differentiated scheduling scheme to satisfy QoS requirements that are imposed on virtualized devices. We have built the network virtualization framework combining the Virtual Box, time-slot-based time-sharing scheme, and leaky-bucket controller, and then we have conducted a performance evaluation study with real testbed. The empirical study indicates that the service differentiation for virtualized network devices is successfully supported by the proposed framework.**

Keywords **-** Network Virtualization, Scheduling Policy, Virtual Box, Virtual Machine

I. INTRODUCTION

There has been a large improvement in the field of virtualization in the past decade. As noted by Goldberg [7], the idea of the virtual machine emerged around 1970s, but, due to the lack of computing power, the field of virtualization has arisen in the early 2000s. The hardware virtualization allows many users and corporations to reduce the expenditure of buying multiple physical machines to support various applications since it runs those applications with multiple virtual machines in a physical machine. Additionally, the virtualization motivates us to provide an efficient way to run multiple networks, each combined with many networks and/or parts of networks into a virtual network or each isolated with a suite of applications in an independent execution environment with a pseudo network interface. The network virtualization gives the network service providers economic benefits since it decouples network infrastructure installment from network service deployment by running multiple virtual networks over a physical network. Also, the network virtualization benefits consumers with customized programmable network services by encapsulating one or more services into a single virtual machine and activating one or more of them according to customer's demand. One of key components for realizing the network virtualization is to isolate one set of network services from another and to control and manage their access to network resources according to QoS specifications. Therefore, the scheduler among virtualized network devices should be implemented with priority. The works such as the Xen [1] and VMWare [16] have mainly dealt with how to distribute the CPU usage fairly amongst the virtual machines (VMs). Moreover, the work such as the Denali [19] has been focused on scheduling I/O fairly amongst the VMs. MultiNet [3] has devised a framework that virtualizes the IEEE 802.11 wireless LAN card, and has proposed a fair scheduling algorithm among virtualized network interface cards. As we can see with the existing works (that are described in Section II), many of the virtualization techniques have been focused on the fairness among virtual machines' CPU and I/O.

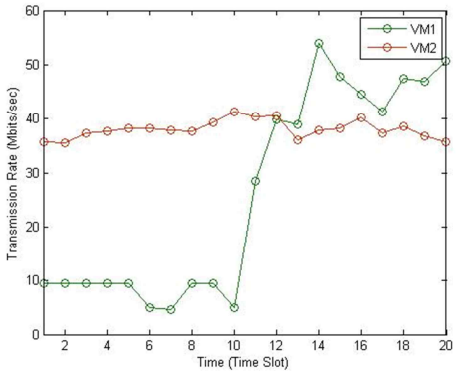

Figure 1. Comparison of network resource usage between two virtual machines without any scheduling scheme.

However, there has not been much research conducting on how to provide service differentiation for the network resources. Specifically, the network service provider or customer may want to allocate a different amount of network resources, e.g. network bandwidth, to each virtualized network device according to QoS specification. Thus, depending on the QoS specification, the network resource should be distributed differently to each VM. On the other hand, with current existing technology, if n VMs exist, then each VM should have $1/n$ rate of the work. However, such the fair allocation cannot be always guaranteed. A unfair resource allocation is presented in Figure 1, where the network bandwidth usage is compared when two virtual machines compete for the network device. We can observe from the figure that the result of current scheduling scheme for virtual machines is not effective in perspective of service differentiation over the virtualized network devices.

Based on this motivation, we propose an internal network virtualization framework to virtualize network devices, which is based on virtual machine, and also, we present a differentiated scheduling scheme to support service differentiation that is imposed on the virtualized devices. We implemented the service differentiation by juxtaposing the leaky-bucket controller [11] and the time slot-based resource allocator [12]. Conclusively, the major contributions are three-fold:

1. To provide service differentiation for internal network virtualization for the first time;
2. To build up the leaky bucket controller and time slot-based resource allocator for virtualized network device;
3. To carry out performance evaluation study in real testbed in terms of (*a*) network performance and (*b*) inter-packet delays to evaluate the service differentiation for

 ISBN: 978-1-61208-133-5

virtualized network devices.

To the best of knowledge, this is the first trial of provisioning the service differentiation for the virtualized network devices.

The rest of the paper is organized as follows. In Section II, we summarize related work in the area of scheduling methods for virtual machines. Then we describe both the specific design and the architecture for implementing the internal network virtualization framework for service differentiation in Section III. With a real testbed, we discuss about the performance and feasibility of the proposed service differentiation framework in Section IV. Finally, we conclude the paper with Section V.

II. PRELIMINARY

In this section, we briefly summarize previous work focused on scheduling methods in virtual machines, and then we explain the specific motivations.

A. Related Work

Many approaches are available to address the scheduling problem within the virtualization. However, there has not been any paper related with provisioning service differentiation for the virtualized network resources. We thus simply explain existing scheduling methods for virtualization according to two categories: I/O based and CPU based.

The CPU based fair scheduling approach focuses entirely on the virtual CPU in order to distribute the host machine's CPU fairly amongst the virtual machines. Govindan *et al.* [8] proposed to use credit-based scheduling algorithm to distribute the CPU resource fairly amongst the VMs by devising a credit scheduler to assign and monitor the credits for each VM. Gulati *et al.* [9] studied on how to proportionally schedule the virtual CPU amongst the VMs in order to improve the CPU fairness by using the Adaptive DRR. Scheduling I/O based fair scheduling approaches extended the mechanism on top of the credit scheduler [8] by adding the BOOST state on the credit scheduler. Instead of only using the Under and Over state, Ongaro *et al.* [14] implemented the BOOST state where it prioritizes the I/O scheduled VM. With this implementation, it provides better chance for I/O-bounded work to control the CPU of the host machine.

Note that all the previous approaches aim at encouraging fair scheduling for CPU or I/O based, extensively relied on the Xen [1] hypervisor. On the contrary, our proposed work aims at how to support service differentiation among multiple VMs without modifying the guest OS. By providing service differentiation method for virtualized network devices, we can dynamically control the network usage for each VM, based on the type of work or a given QoS specification.

B. Motivation: Limitation in Existing Scheduling Schemes

The scheduler in a virtual machine is responsible for assigning computing resources to each virtual machine. It usually exists at the virtual machine monitor (VMM), which is a software layer where it virtualizes many of the resources of the physical host machine. In order for the VMM to handle the task, the resources such as the CPU, network device, I/O devices, and physical memory need to be virtualized. Additionally, there are still many challenges in order to fairly schedule these devices. For example, Virtual Box [18]'s VM scheduling algorithm basically depends on the host machine's thread scheduling mechanism, where it gives the impression of distributing the resource fairly to the VM until the VM is dead. However, with some portion of accuracy, it is not quite true. On the other hand, the scheduling algorithms, such as Xen's Credit Scheduler [8] and Ongaro *et al.* [14], have been taken to distribute the resources fairly to the CPU and the I/O devices. These works focused on how to schedule the resource in order to distribute the resource into *n* number of VMs. The advantage of these algorithms is that it can almost distribute and share the physical resource of the host machine *almost equally* amongst the virtual machines. However, the question of how to provide a service differentiation according to a QoS specification is still unclear.

The main objective of this work is to implement a scheduling algorithm in order to realize service differentiation by coordinating the progress of multiple VMs according to a given QoS specification. For example, if VM1 has been assigned to work for 30% of CPU usage then it is mandatorily use 30% CPU usage as well as the other VM use the rest of the CPU usage, 70%.

III. DESIGN AND IMPLEMENTATION

In this section, we describe the internal network virtualization framework for virtualizing network devices, and the scheduling scheme of providing the service differentiation.

A. Architecture

In order to virtualize network devices, we chose a virtual machine (VM) approach based on the Virtual Box OSE 3.16 SDK[17]. The Virtual Box OSE is a open source software developed by Oracle. Each VM is considered as an EMT, Emulation thread, when the host machine schedules the threads. Unlike the Xen's Credit Scheduler [8], Virtual Box does not have a customary scheduler where it schedules effectively amongst the VMs. However, since the Virtual Box SDK 3.16 [17] provides interfaces to interact with the VMM and virtual device to control the VMs that run concurrently, we implement a scheduling scheme for implementing service differentiation. Figure 2 presents the architecture for the proposed network virtualization framework with the differentiated scheduling scheme.

B. Scheduling Scheme for Service Differentiation

In order to realize the service differentiation in scheduling scheme for the virtualized network devices, we chose two basic building blocks. The first one is the leaky bucket controller, which has been used in packet switched networks and the telecommunications networks in order to regulate the data transmission with the credit-generating rate [1] and the burstiness[11]. In our proposed scheduling scheme, the controller generates the credits according to the QoS specification. The other one is time-slot based resource allocator, with which we can decide the basic time allocation unit instead of using infinitesimal time unit. With these two schemes, we designed a scheduling scheme for providing

[1] We adopted the concept of credit to assign CPU resource to each VM.

 ISBN: 978-1-61208-133-5

service differentiation for virtualized network devices. The scheduler is presented in Figure 3.

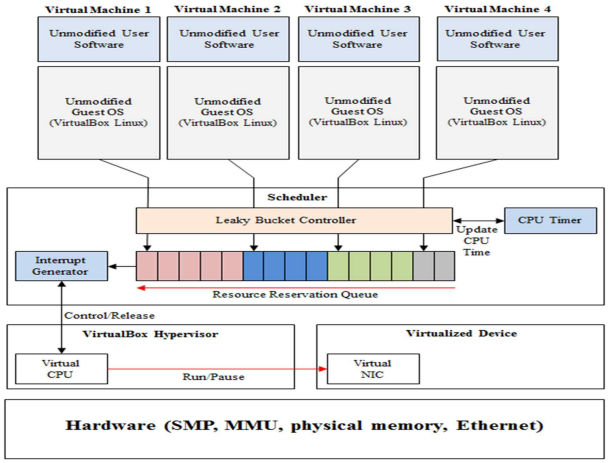

Figure 2.The architecture for the proposed network virtualization.

Once the system starts, the scheduler is periodically executed to produce credits via the leaky-bucket controller and assign the credits to each VM. For example, if there are two VMs and the ratio of CPU usage is given with 65:35 as the QoS specification for two VMs, VM1 should be assigned with use the resource usage of 65% and VM2 should acquire 35% of resource usage. Based on this ratio, the scheduler assigns the credits to each designated VM. Once every VM acquires its own credits, the scheduler assigns the time slot to it. In order to distribute the time slot to the VMs, we used the following equation:

$$VM_i TimeSlot = \frac{VM_i Credit}{\left(\frac{\sum_i^n VM_i Credit}{\#ofTimeSlot}\right)} \quad (1)$$

Each VM is basically inserted into a scheduling queue, and then it is scheduled in a round robin way. If the credits allocated to a VM are used up, then the VM should wait till the scheduler assigns additional credits to it. Otherwise, the VM runs during the time slot, and then, it is reinserted to the queue after the time slot is expired.

```
procedure scheduler()

assign workload (VMi) according to a QoS;
struct cpu_reservation_schedule q;

while(system is running) {
  compute all of the credits for all VMi;
  for i=0 to all VM
       compute the schedule for VMi with (1);
       insert VMi to q(ω);
  update time();
  for ω =0 to sizeof(q)
        if q (ω)=VMi.D
            run VMi for one time slot
            pause rest of the VM within the q(ω)
        else
            decrement credit of the rest of
                  the VM within the q(ω);
   update time();
}
```

Figure 3. The deterministic scheduling algorithm.

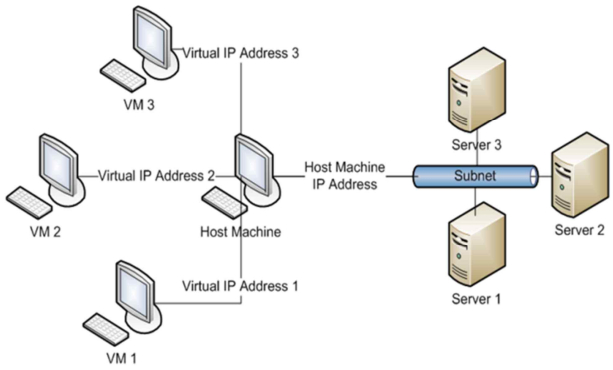

Figure 4. The testbed for performance evaluation study.

IV. PERFORMANCE EVALUATION

In this section, we present the results of performance evaluation, which has been done with some application level benchmarks in the *real testbed*, in order to investigate the performance of the proposed network virtualization framework for providing service differentiation for virtualized network devices

All the empirical experiment have been conducted on a 3.06GHz Intel Core2 Duo with 3MB of L2 cache, 4GB of RAM, and 10/100/1000BASE-T Gigabit Ethernet card. The operating system is Ubuntu 10.04 and the guest OS for the VMs are Ubuntu 9.10. As aforementioned, the Virtual Box OSE 3.16 is employed as the VMM. To generate the UDP and the TCP traffic, we employed the *iperf* [10] utility, which is supposed to constantly generate packets from the VM to the server, and we also used *ping* flood to generate ICMP traffic. The testbed for this study is in Figure 4, where three VMs are resident at one physical machine and each VM communicates with its corresponding real server over the network.

As for the schedulable resource, we used the CPU usage under the assumption that the time amount of using network devices is proportional to that amount of using CPU. As for the performance metrics, we use three metrics, the network bandwidth (transmission rate), inter-packet delay and CPU usage to verify if the proposed scheduling scheme can achieve the service differentiation according to the QoS specification. Note that we selectively present the empirical results in terms of network bandwidth and inter-packet delay due to the space limit. Finally, we have conducted two empirical evaluation study sets: the one is when we activated two virtual machines, and the other is when we used three virtual machines.

A. With Two Virtual Machines

Firstly, we conducted an empirical study with two virtual machines, and we employed UDP, TCP, ICMP traffic to verify if the differentiation is achieved.

In the case of UDP traffic: We have tested in two scenarios: the ratio of CPU usage between VM1 and VM2 is 50:50 (%), and the ratio is 60:40. Figure 5 shows the result of the first case, whereas Figure 6 shows the other case.

As for the results in Figure 5, the average bandwidth of each VM was very similar to each other and it is consistent over time. In specific, the average bandwidth of VM1 is 46.90 Mbits/sec whereas the average of VM2 is 46.95 Mbits/sec. As

 ISBN: 978-1-61208-133-5

for the results in Figure 6, we can observe that the average bandwidth of VM1 is 55.41 Mbits/sec and average bandwidth of VM2 is 38.29 Mbits/sec, which indicates that VM1 used about 60% of total network bandwidth whereas the VM2 used nearly 40% of the bandwidth.

Additionally, we investigate the impact of the differentiated scheduling on the inter-packet delay. Figure 7 presents the fluctuation of inter-packet delays that we observed from the original Virtual Box system (without any change). Specifically, the average delay is 0.303*ms* and its standard deviation is 0.0539*ms* for VM1, and those values for VM2 are 0.327*ms* and 0.0593*ms*, respectively. However, when we used the proposed differentiated scheduling scheme with ratio of 50:50, we could observe *stable and fair dynamics of inter-packet delays* which is presented in Figure 8. In particular, as for VM1, the average delay and standard deviation are 0.132*ms* and 0.0159*ms,* respectively, and, as for VM2, those values are 0.131*ms* and 0.0167*ms*.

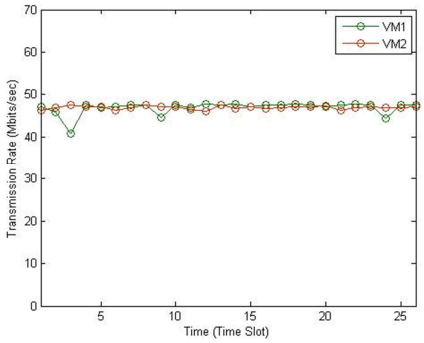

Figure 5. Comparison of network bandwidth when the ratio of using network bandwidth between VM1 and VM2 is 50: 50 and UDP traffic is employed.

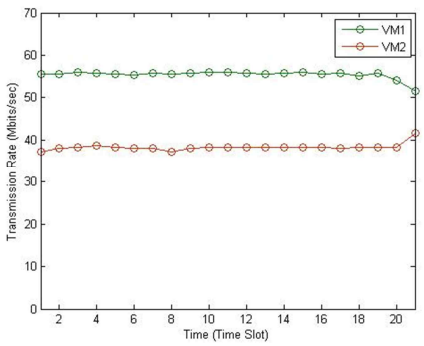

Figure 6. Comparison of network bandwidth when the ratio of using network bandwidth between VM1 and VM2 is 60: 40 and UDP traffic is employed.

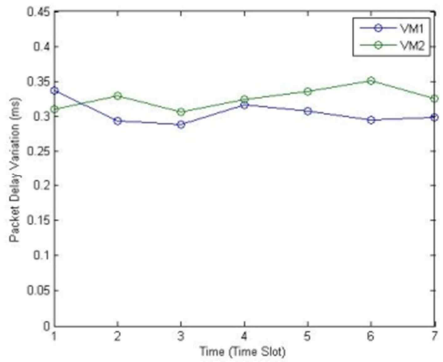

Figure 7. The fluctuation of inter-packet delays that are measured with the original Virtual Box when two VM are activated and UDP traffic is used.

In the case of TCP traffic: When we used TCP traffic, we made a similar observation. Figure 9 compares two network bandwidths when the ratio of network bandwidth usage between two VMs is 60:40. From the figure, we observed that the required differentiation is successfully achieved; specifically, the average of VM1 is 52.85 Mbits/sec and VM2 is 33.72 Mbits/sec.

***In the case of ping (ICMP) traffic*:** We have used the ping flood to verify if the proposed service differentiation is still effective in ICMP traffic. The ratio between VM1 and VM2 for the QoS specification is set to 50:50. Figure 10 compares two network throughputs each of which is used by VM1 and VM2, respectively. The reported average bandwidth of VM1 is 20.05 Mbits/sec, and that of VM2 is 20.26 Mbits/sec.

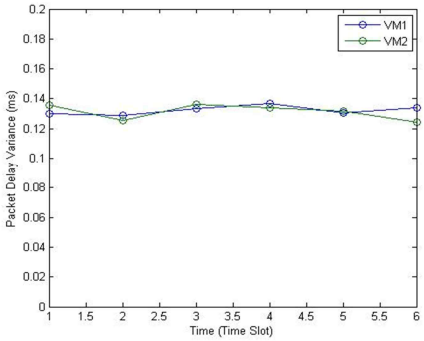

Figure 8. The stable fluctuation of inter-packet delays that are measured under the proposed differentiated scheduling scheme when the ratio of network bandwidth usage between VM1 and VM2 is 50:50 and UDP traffic is used.

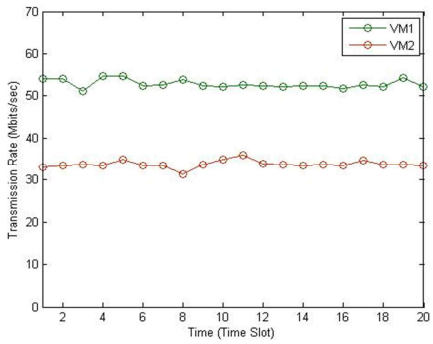

Figure 9. Comparison of network bandwidth when the ratio of using network resources between VM1 and VM2 is 60:40 and TCP traffic is employed.

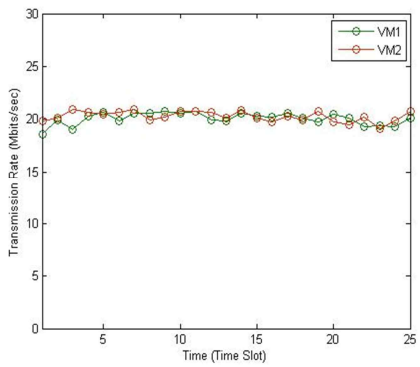

Figure 10. Comparison of network bandwidth when the ratio of using network resources between VM1 and VM2 is 50: 50 and ICMP traffic (generated by ping traffic) is employed.

 ISBN: 978-1-61208-133-5

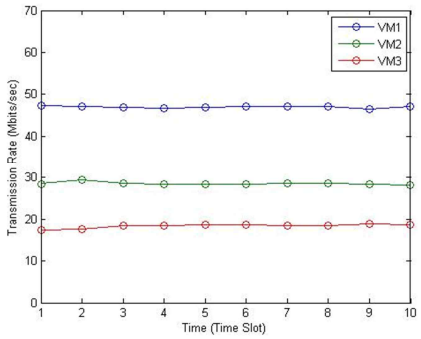

Figure 11. Comparison of three network bandwidths when the ratio among VM1, VM2 and VM3 is 50:30:20 and UDP traffic is employed.

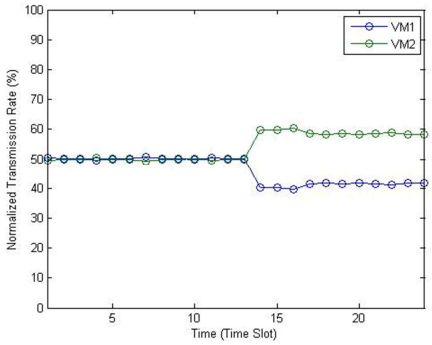

Figure 12. Comparison of two network bandwidth when the ratio between VM1 and VM2 is changed from 50:50 to 60:40 and TCP traffic is employed.

B. With Three Virtual Machines

As the second empirical study, we used three VMs in order to check whether or not the differentiated scheduling scheme is immune to the number of VMs. Figure 11 compares three network bandwidth usages when we use UDP traffic and the ratio for the VMs is set to 50:30:20. We made similar observations to previous empirical studies: the service differentiation is successfully achieved among VMs.

C. Dynamic Service Differentiation

As the last empirical study, we used a dynamic QoS scenario where the ratio betwen two VMs is changed from 50:50 to 40:60. The results are presented in Figure 12. Even though the ratio is changed, the service differentiation that is supported by the proposed scheduling scheme for the virtualized network devices is not affected by the change.

V. CONCLUSION

In this paper, we proposed an internal network virtualization framework based on Virtual Box, and also we built up a scheduling scheme for providing service differentiation among VMs. We specifically presented the proposed architecture for virtualizing network devices and scheduling those devices according to QoS specifications. Then we have demonstrated that the service differentiation can be successfully achieved through both the proposed virtualization framework and the differentiated scheduling scheme, regardless of network traffic, the number of VMs, or dynamic change of QoS specification. Note that the proposed scheduling scheme can cooperate with any framework that supports Virtual Box without the modification.

In the future work, we would like to devise various scheduling resources that can be used elaborately to schedule the virtualized network devices in the proposed framework. We also plan to examine the effect of the proposed scheduling scheme on real multimedia traffic. The study corroborates the effectiveness of the proposed virtualization framework for network devices.

ACKNOWLEDGEMENT

This work was supported in part by the IT R&D program of MKE/KEIT [KI001822, Research on Ubiquitous Mobility Management Methods for Higher Service Availability], and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0014060)

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proc. 19th SOSP,* Lake George, NY, Oct 2003.

[2] E. Bugnion, S. Devine, and M. Rosenblum. Disco: Running Commodity Operating Systems on Scalable Multiprocessors. In *Sixteenth ACM Symposium on Operating System Principles*, October 1997.

[3] R. Chandra, V. Bahl, and P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *INFOCOM*, 2004.

[4] G. W. Dunlap, S. T. King, S. Cinar, M. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, ACM Operating Systems Review, Winter 2002 Special Issue, pages 211.224, Boston, MA, USA, Dec. 2002

[5] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, and M. Williamson. Safe hardware access with the Xen virtual machine monitor. In *Proceedings of the Workshop on Operating System and Architectural Support for the On Demand IT InfraStructure (OASIS)*, Oct. 2004.

[6] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A Virtual Machine-Based Platform for Trusted Computing," in *Proc. 9th ACM Symposium on Operating Systems Principles*, 2003, pp. 193–206.

[7] R. Goldberg. *Architectural Principles for Virtual Computer Systems*. PhD thesis, Harvard University, 1972.

[8] S. Govindan, A. R. Nath, A. Das, B. Urgaonkar, and A. Sivasubramaniam. Xen and co.: communication-aware cpu scheduling for consolidated xen-based hosting platforms. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 126–136, New York, NY, USA, 2007. ACM Press.

[9] A. Gulati, A. Merchant, M. Uysal, and P. Varman, Efficient and adaptive proportional share I/O scheduling, *Technical Report HPL-2007-186, HP Labs*, November, 2007.

[10] Iperf, http://sourceforge.net/projects/iperf

[11] J. Kurose and K. Ross, Computer Networking: A Top-Down Approach 4th Edition, page 675-678, Boston, MA, 2008, Pearson Education International

[12] J. Kurose and K. Ross, Computer Networking: A Top-Down Approach 4th Edition, page 477, Boston, MA, 2008, Pearson Education International

[13] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal 10*, 3 (2006).

[14] D. Ongaro, A. Cox, and S. Rixner. Scheduling I/O in virtual machine monitors. In Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pages 1-10, ACM, 2008

[15] P. Pancha and M. El Zarki. Leaky bucket-access control for VBR MPEG video. In *Proceedings of the IEEE INFOCOM'95,* Boston, April 1995

[16] J. Sugerman, G. Venkitachalam, and B. Lim. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In *Proc. 2001 Ann. USENIX Tech. Conf.*, Boston, MA, USA, June 2001.

[17] Sun Virtual Box Programming Guide and Reference 3.1.6, http://www.virtualbox.org

[18] Virtual Box, http://www.virtualbox.org

[19] A. Whitaker, M. Shaw, and S. D. Gribble. Denali: Lightweight Virtual Machines for Distributed and Networked Applications. Technical Report 02-02-01, University of Washington, 2002.

 ISBN: 978-1-61208-133-5