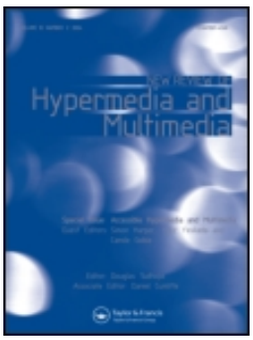

New Review of Hypermedia and Multimedia

ISSN: 1361-4568 (Print) 1740-7842 (Online) Journal homepage: https://www.tandfonline.com/loi/tham20

Guaranteeing QoS of media-based applications in virtualized environment

Like Zhou, Song Wu, Xuanhua Shi, Hai Jin & Jiangfu Zhou

To cite this article: Like Zhou, Song Wu, Xuanhua Shi, Hai Jin & Jiangfu Zhou (2013) Guaranteeing QoS of media-based applications in virtualized environment, New Review of Hypermedia and Multimedia, 19:3-4, 217-233, DOI: 10.1080/13614568.2013.834487

To link to this article: https://doi.org/10.1080/13614568.2013.834487

Published online: 14 Nov 2013.

New Review of Hypermedia and Multimedia, 2013
Vol. 19, Nos. 3–4, 217–233, http://dx.doi.org/10.1080/13614568.2013.834487

Taylor & Francis
Taylor & Francis Group

Guaranteeing QoS of media-based applications in virtualized environment

LIKE ZHOU, SONG WU*, XUANHUA SHI, HAI JIN and JIANGFU ZHOU

Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, PR China

(Received 29 August 2012; final version received 10 August 2013)

With the rapid development of web technology and smart phone, multimedia contents spread all over the Internet. The prevalence of virtualization technology enables multimedia service providers to run media servers in virtualized servers or rented virtual machines (VMs) in a cloud environment. Although server consolidation using virtualization can substantially increase the efficient use of server resources, it introduces resources competition among VMs running different applications. Recently, hypervisors do not make any *Quality of Service* (QoS) guarantee for media-based applications if they are consolidated with other network-intensive applications, which leads to significant performance degradation. For example, Xen only offers a static method to allocate network bandwidth. In this paper, we find that the performance of media-based applications running in VMs degrades seriously when they are consolidated with other VMs running network-intensive applications and argues that dynamic network bandwidth allocation is essential to guarantee the QoS of media-based applications. Then, we present a dynamic network bandwidth allocation system in virtualized environment, which allocates network bandwidth dynamically and effectively, and does not interrupt running services in VMs. The experiments show that our system can not only guarantee the QoS of media-based applications well but also maximize the system's the overall performance while ensuring the QoS of media-based applications.

Keywords: Multimedia; Network management; Virtualization

1. Introduction

In the traditional computing environment, operating systems run on the physical hardware directly and manage all kinds of hardware resources. However, the

*Corresponding author. Email: wusong@hust.edu.cn

appearance of virtualization technology (Menascé 2005) changes many things. Virtualization technology separates the physical hardware and the operating system perfectly, which enables multiple operating systems to run on the same physical machine simultaneously. Virtualization technology has quite a wide usage, especially, in data centers and large-scale cluster environments (Nathuji and Schwan 2007, Hermenier *et al*. 2009, Stillwell *et al*. 2009). With server consolidation in data centers, virtualization technology improves the resource utilization rate, reduces the energy consumption, and saves the cost on server management (Rosenblum and Garfinkel 2005, Padala *et al*. 2007, Wood *et al*. 2007).

With the rapid development of web technology and smart phone, multimedia contents spread all over the Internet. Due to the advantages of virtualization, multimedia service providers may run media servers in virtualized servers or rented virtual machines (VMs) in cloud environment, such as Amazon's Elastic Compute Cloud (EC2). Because media-based applications (such as streaming media server, VoIP) have strong demands on central processing unit (CPU) and network resources (Patnaik *et al*. 2009, Barker and Shenoy 2010), and virtualization introduces resources competition among VMs running different applications, the *virtual machine monitor* (VMM), also called hypervisor, must provide enough resources to the VMs running media-based applications to guarantee their *Quality of Service* (QoS). However, recent studies (Chen *et al*. 2010, Lee *et al*. 2010, Kim *et al*. 2012) only focus on optimizing CPU scheduling or providing enough CPU resources to the VMs to improve the performance of media-based applications. They only address one aspect and do not solve the network bandwidth competition among different VMs.

We find that the performance of media-based applications running in VMs degrades seriously when they are consolidated with other VMs running network-intensive applications. The competition on network resources is the key factor to affect the QoS of media-based applications. If we want to guarantee the QoS of media-based applications, we must provide enough network bandwidth to the corresponding VMs. However, although Xen (Barham *et al*. 2003) allows users or administrators to change memory size or virtual CPU resources when VMs are running, it only offers a static method to allocate network bandwidth, which cannot allocate network bandwidth when VMs are running and results in a waste of network resources.

We argue that dynamic network bandwidth allocation is essential to guarantee the QoS of media-based applications. The reasons are as follows: First, we cannot decide how much network bandwidth should be allocated to a VM before starting it. Second, the network bandwidth demands of media-based applications always change as time goes on. Finally, if users or administrators know the network bandwidth demands of media-based applications, they cannot allocate appropriate network bandwidth to corresponding VMs on the fly in current hypervisors.

The aim of this paper is to guarantee the QoS of media-based applications in virtualized environment from the aspect of network management. We conduct a motivational experiment to demonstrate the importance of providing enough network resources to a media-based application to guarantee its QoS. Then, we design and implement a dynamic network bandwidth allocation system in the Xen

hypervisor, named *DNBA*, which enables administrators to allocate network bandwidth dynamically when VMs are running. All the operations are transparent to services and do not interrupt the running services. Our experiments show that *DNBA* can allocate network bandwidth dynamically and effectively and guarantee the QoS of media-based applications very well. Besides, *DNBA* can maximize the overall performance of the system while ensuring the QoS of media-based applications.

The main contributions of this paper are as follows:

- We identify the network competition problem that network competition affects media-based applications running in VMs seriously, and argue that dynamic network bandwidth allocation is essential to guarantee the QoS of media-based applications.
- We present a dynamic network bandwidth allocation system to address the network competition problem, which is not solved by the current hypervisors, and we conduct various experiments to validate the effectiveness of the system. The experimental results show that allocating enough network bandwidth to media-based applications is the key factor to guarantee the QoS of these applications.

The rest of the paper is organized as follows: Section 2 provides an overview of background information and related work. Then we present a motivational example to illustrate the drawback of current network virtualization in Section 3. Section 4 describes the design and the implementation of the dynamic network bandwidth allocation system. Section 5 presents performance evaluation. In Section 6, conclusions summarize results and future work.

2. Background and related work

2.1. Network virtualization

Xen uses split-driver model to virtualize input/output (I/O) devices. A device driver is split into a front-end driver and a back-end driver. The front-end driver is running in a guest domain or DomainU, and the back-end driver is running in an *Isolated Driver Domain* (IDD) or Domain0. All the requests and replies of the front-end driver are processed by the back-end driver. However, the split-driver model needs to modify the kernel of the guest operating systems (guest OSes), which is adopted by para-virtualization. Xen uses Qemu (Bellard 2005) to emulate devices in full virtualization.

The network architecture of Xen is shown in Figure 1. DomainU uses the netfront (or network front-end driver) as its network device driver, which sends packets to the netback (or network back-end driver) in Domain0. The packets then go through the Linux software bridge and, finally, go to the network via Linux's native network driver. The process of receiving packets is reverse.

System administrators always need to set up VMs' network, including limiting their maximum network bandwidth. However, Xen only provides a static method to limit the maximum network bandwidth, which specifies the bandwidth rate in

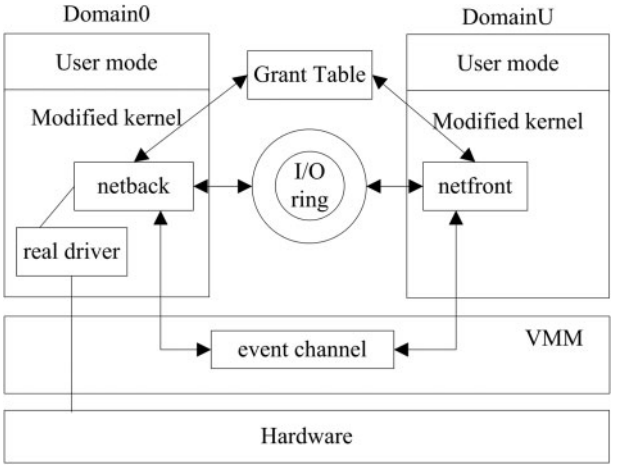

Figure 1. Network architecture of Xen.

the domain-specific configuration file:

$$\text{vif} = [\text{'type} = \text{netfront, bridge} = \text{xenbr0, rate} = \text{xMB/s'}]$$

When users create a VM using the configuration file, the rate parameter is stored in *XenStore*, which is an information storage space shared between domains. The maximum network bandwidth of the VM is limited to xMB/s permanently.

When a VM is created, the network back-end driver reads the bandwidth parameter from *XenStore* and allocates a certain number of *credits* to the VM. When the VM is running, the network back-end driver ensures that in a defined time, T, the *credit* consumption of the VM will not exceed C (the value of C is calculated according to the bandwidth parameter and T).

When the VM sends/receives data via network, it consumes corresponding amount of *credits*. When the VM drains the *credits*, the network back-end driver checks whether the time exceeds T. If the time does not exceed T, the rest of the *credits* cannot support a complete network operation. The virtual network interface of the VM will "sleep" for a while, until the system reallocates credits at the end of time T; otherwise, the network front-end driver still can send/receive data.

From the above discussion, we can come to the conclusion that there is still some work we can do in network bandwidth allocation, which is used to guarantee the QoS of media-based applications. First, various applications have different demands on network resources. Users cannot know the exact network bandwidth demands of applications in advance, and they always change as time goes on. Second, Xen provides tools to adjust memory size and virtual CPU resources of a VM when it is running. However, there is no tool in Xen that can change the network bandwidth on the fly. Finally, dynamic network bandwidth allocation is the base of adaptive network bandwidth allocation, which can adjust network bandwidth automatically according to the demands of applications. Therefore, it is urgent to design and implement a dynamic network bandwidth allocation system.

2.2. Related work

In order to increase the performance of media-based applications, most studies (Chen *et al*. 2010, Lee *et al*. 2010, Kim *et al*. 2012) optimize the CPU scheduler of VMM. This is because of the fact that CPU resource is the most important one in the whole system, and the CPU scheduler can affect I/O performance in virtualized environment. Lee *et al*. (2010) introduce *laxity* to denote the emergency of a VM. The VM with low *laxity* means that it is desirable to run. Kim *et al*. (2012) modify Xen's Credit scheduler to reallocate *credits* for the VM running multimedia applications adaptively. If the quality of multimedia applications does not meet the expectation, the system allocates more *credits* to the target VM; otherwise, it decreases *credits* that the VM can get. Chen *et al*. (2010) introduce real-time priority and dynamic time-slice to credit scheduler. They use feedback control to adjust the real-time priority of the VM running audio applications to meet their quality adaptively. As applications running in VMs are always associated with a *service level agreement* (SLA), Zhong *et al*. (2012) investigate the performance implications of the nonuniform virtual central processing unit (VCPU)—physical central processing unit (PCPU) mappings in virtualized environment.

Media-based applications always have a lot of I/O operations. The performance of I/O virtualization affects the quality of such applications directly. There is a lot of work to improve the performance of network virtualization. VMM-bypass I/O (Liu *et al*. 2006) is a new device virtualization model. VMM-bypass allows time-critical I/O operations to be carried out directly in the guest VMs without involvement of the VMM and/or a privileged VM. Ram *et al*. (2009) use multiqueue network card to eliminate the software overhead of demultiplexing and copying and reuse a grant to reduce memory protection overheads. With the number of cores growing in modern architectures, using a dedicated core to improve the performance of networking becomes feasible (Liu and Abali 2009, Shalev *et al*. 2010). Besides, there is also some work to improve the performance of I/O virtualization from the aspect of CPU scheduling (Govindan *et al*. 2007, Ongaro *et al*. 2008, Kim *et al*. 2009, Hu *et al*. 2010). VM placement may affect the performance of media-based applications. Liao *et al*. (2010) present an optimized resource distribution policy to increase the resource utilization of virtual cluster and shorten the response time. Besides, QoS can be improved from the aspect of workflow (Cao *et al*. 2010).

There are also some studies related with multimedia in traditional environment. Elmisery and Botvich (2011) present a framework for a private internet protocol television (IPTV) recommender service based on *Enhanced Middleware for Collaborative Privacy* (EMCP), which not only protects the user's privacy but can also maintain the accuracy of the recommendations. Vakili and Gregoire (2011) present a method based on packet scheduling to improve the performance of video streaming over noisy channels. Bhattacharya *et al*. (2012) introduce an affect-based methodology of *Quality of Experience* (QoE) evaluation in voice communication. Luo and Shyu (2011) survey the QoS provision in mobile multimedia at different network layers and cross-layer design and provide some thoughts about the challenges and the directions for future research.

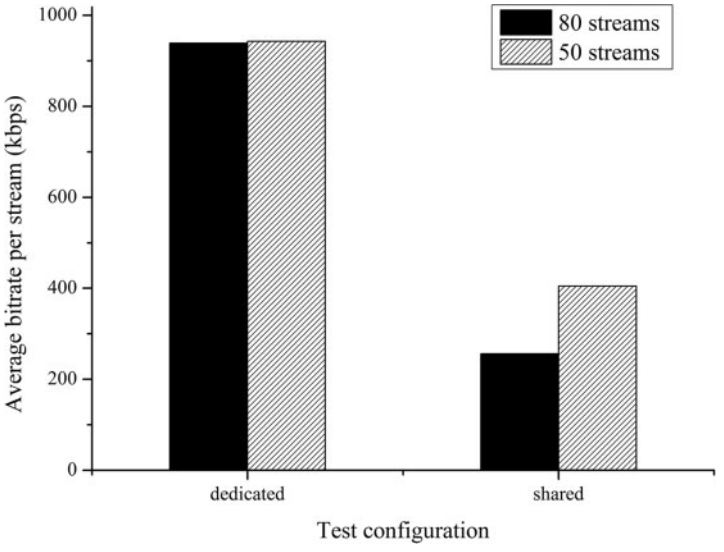

Figure 2. Average bit rate over 80 or 50 concurrent streams in different configurations.

3. Motivational example

In this section, we conduct a test to illustrate the drawbacks of existing network virtualization in Xen and to demonstrate the importance of network bandwidth allocation to guarantee the QoS of media-based applications. The test environment is described in Section 5. We use two VMs to conduct this test. They run *Darwin Streaming Server* (DSS) and a file transfer protocol (FTP) server, respectively. We run 80 and 50 concurrent streams to generate loads to DSS, respectively, and use *wget* to download 1 GB file from the FTP server. We conduct this test in two configurations: dedicated configuration and shared configuration. The dedicated configuration means that there is no background interference. The shared configuration means that both VMs are running on the same physical machine simultaneously, and we generate loads to DSS and the FTP server at the same time. Thus, these VMs compete for network bandwidth with each other. We are interested in two primary metrics: the average bit rate per stream (the metric of DSS) and the total file transfer time (the metric of the FTP server). The test results are shown in Table 1 and Figure 2.

As shown in Figure 2, we can see that the performance of DSS degrades seriously in the shared configuration. Table 1 shows an interesting phenomenon. The file transfer time is 94 s in the dedicated configuration, but it is 106 s and

Table 1. File transfer time in different configurations.

		Shared	
Configurations	Dedicated	80 streams	50 streams
File transfer time (s)	94	106	105

105 s in the shared configuration under the 80 and the 50 concurrent streams tests, respectively. The file transfer time in the shared configuration shows almost no difference despite the different loads to DSS and increases slightly compared to the file transfer time in the dedicated configuration. We observe that the VM running the FTP server can use 80% of network bandwidth of the network card even if we generate different number of concurrent streams. The VM running the FTP server is allocated more network bandwidth than the VM running DSS. Moreover, we run DSS and the FTP server in the same VM and the native operating system (OS; CentOS 5.3), respectively. We observe that DSS and the FTP server share the network bandwidth fairly in both scenarios. Therefore, the network bandwidth distribution between these VMs is unfair in virtualized environment, and the performance of DSS degrades seriously.

We summarize the motivational example as follows. First, Xen does not guarantee adequate network isolation. A VM can use more network bandwidth than the other. Second, the performance of the streaming media server is not guaranteed when network-intensive applications running in other VMs compete for network resources. If we want to allocate more network bandwidth to a VM according to the application's requirement, Xen does not provide tools to do this. Thus, in order to guarantee the QoS of media-based applications, we need to design and implement a system to allocate enough network bandwidth to corresponding VMs.

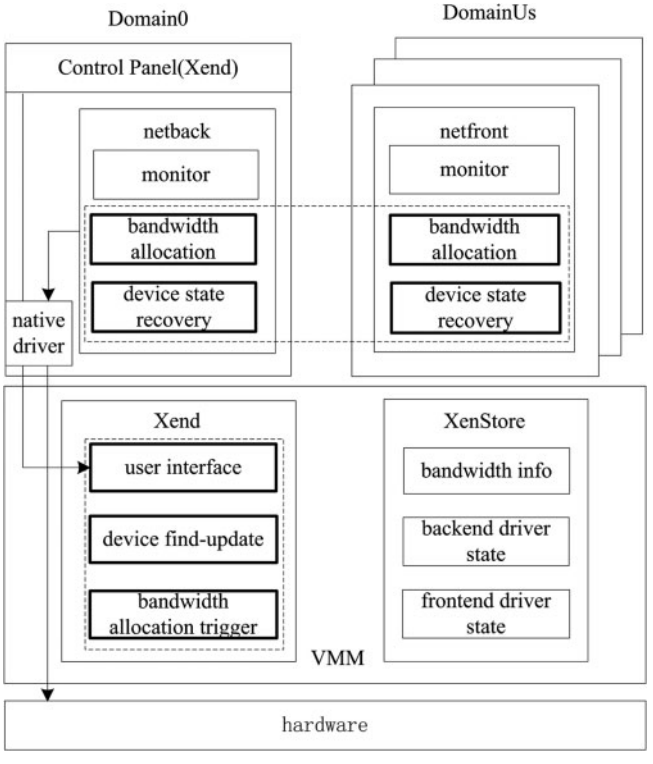

Figure 3. System architecture.

4. Design and implementation

As described in Section 2, Xen uses the configuration file to specify the maximum limit of network bandwidth. With server consolidation, VMs running different applications are running on the same physical machine simultaneously. The network bandwidth demands of these VMs are different and always change as time goes on. Therefore, the static method, which lacks flexibility, is not perfect in virtualized environment and cannot guarantee the QoS of media-based applications properly. Taking into account the above reasons, we design and implement a dynamic network bandwidth allocation system in virtualized environment, named *DNBA*. The goals of *DNBA* are as follows: First, it is compatible with the management of virtual device in Xen, which is accepted by most users, and has high scalability. Second, a friendly interface must be provided to users. Finally, users can adjust VMs' maximum limit of network bandwidth when VMs are running, and the operation is transparent to guest OSes and applications, namely, the dynamic network bandwidth allocation does not interrupt running services.

The system architecture is shown in Figure 3. On one hand, we add a bandwidth allocation command to the user's console and provide application programming interfaces (APIs) to programmers, which enable cloud providers to design and deploy their network bandwidth allocation strategies. On the other hand, we add the device state and the bandwidth allocation method in the network back-end driver. The procedure of dynamic network bandwidth allocation is as follows: The system accepts user's bandwidth allocation command from the console, analyzes parameters, and then generates a trigger to the bandwidth allocation module. The bandwidth allocation module in the back-end driver responds to the trigger and executes transparent bandwidth allocation.

System administrators can use *DNBA* to configure the maximum limit of network bandwidth as follows:

xm config <*domid*> <*vifid*> <*new-rate*>

domid is the ID of a VM, *vifid* is the ID of virtual network interface, and *new-rate* specifies the maximum limit of network bandwidth. If a VM has many virtual network interfaces, *DNBA* can set the network bandwidth limit for each virtual network interface.

We also provide APIs to programmers and implement a Web UI to monitor and allocate network bandwidth based on the APIs. Cloud providers can also use the APIs to design and to implement their own network bandwidth allocation strategies.

DNBA mainly includes two modules: *bandwidth allocation management module* and *driver layer bandwidth allocation module*.

Bandwidth allocation management module is the bridge of DNBA. It plays an important role in connecting users and *driver layer bandwidth allocation module*. It specifically includes *user interface module*, *device find-update module*, and *bandwidth allocation trigger module*. All of them are located in the management layer of Xen.

In *bandwidth allocation management module*, the system accepts users' bandwidth allocation requests from the console and analyzes parameters to extract the device information (includes the VM's ID and the ID of virtual

network interface) and a new bandwidth value. The system searches the network virtual interface in the hypervisor according to the device information. When the interface is found, it uses the new bandwidth value to update the bandwidth parameter of the virtual network interface, which is stored in *XenStore*.

When a VM is created successfully, the state of the front-end device and the back-end device is *XenbusStateConnected*. We add a new status—*XenbusStateUpdating* to denote that network bandwidth limit needs to update. When users use *DNBA* to reconfigure network bandwidth limit, the system changes the state of the back-end device from *XenbusStateConnected* to *XenbusStateUpdating*, which triggers *driver layer bandwidth allocation module* to complete network bandwidth allocation. At last, *bandwidth allocation management module* receives the result of bandwidth allocation from the driver layer and displays it on the console. The result tells users whether the allocation is successful.

Driver layer bandwidth allocation module is the core of *DNBA*, which, indeed, takes charge of the bandwidth allocation operation. It specifically includes *bandwidth allocation module* and *device state recovery module*. It works based on the following principles:

(1) There is a device state monitoring mechanism in the split-device driver model.

Both the front-end driver and the back-end driver have a monitor in the split device driver model. Each monitor probes the state of the other side device (more precisely, it is *XenBus* state of a device; here we generally use device state to take place of it). That is, to say, the front-end driver probes the state of the back-end device, and the back-end driver probes the state of the front-end device.

(1) There is an interactive device parameter updating process in the split device driver model.

The front-end driver is located in DomainU, and the back-end driver is located in the IDD or Domain0. Both of them can only modify their own device parameter and state. The state of change of one of side the device can trigger the other side to execute corresponding operations based on the state of the monitoring mechanism of the split device driver model. So our system contains a complete front-end and back-end device interactive process in order to finish the device parameter update.

In *driver layer bandwidth allocation module*, the front-end driver probes the state of change of the back-end device and invokes the update callback function of the front-end device, which changes the state of the front-end device from *XenbusStateConnected* to *XenbusStateUpdating*. Then, the back-end driver probes this change and invokes the update callback function of the back-end device, which reads the new bandwidth value from *XenStore*, and writes it to the back-end driver to complete the bandwidth allocation operation. After the bandwidth allocation operation is completed, the state of the front-end device and the back-end device needs to be recovered to *XenbusStateConnected* to ensure their normal

state change in their original life cycle. At last, this module returns the result of bandwidth allocation to *bandwidth allocation management module*.

Figure 4 shows the system workflow that describes the whole operations of *DNBA*.

5. Performance evaluation

In this section, we evaluate the feasibility and the effectiveness of *DNBA* and show that *DNBA* guarantees the QoS of media-based applications well. We first describe the experimental environment and then present the experimental results, including the effectiveness of dynamic bandwidth allocation and ensuring the QoS of media-based applications.

In our experiment, the physical machine has a dual-core Intel CPU (2.6 GHz), 2 GB memory, 500 GB SATA disk, and 100 Mbps Ethernet card. We use Xen-3.4.1 as the hypervisor and CentOS 5.3 distribution with the Linux-2.6.18.8 as the OS. All the configurations of VMs are as follows: 2VCPUs, 384 MB memory, and 20 GB virtual disk.

5.1. General test

Since it is a network bandwidth test, we adopt a widely used network performance analysis tool – IxChariot (*IxChariot – Official website*), and use two VMs to conduct this experiment. For simplicity, we call them as VM1 and VM2. Both of them install an IxChariot's endpoint. IxChariot shows many network test results such as real-time bandwidth curves and bandwidth statistics information including maximum, minimum, average bandwidth, etc. In order to get accurate test results, we set the data to be transferred as 1 GB. We chose the real-time network bandwidth curve and the average bandwidth as our test results.

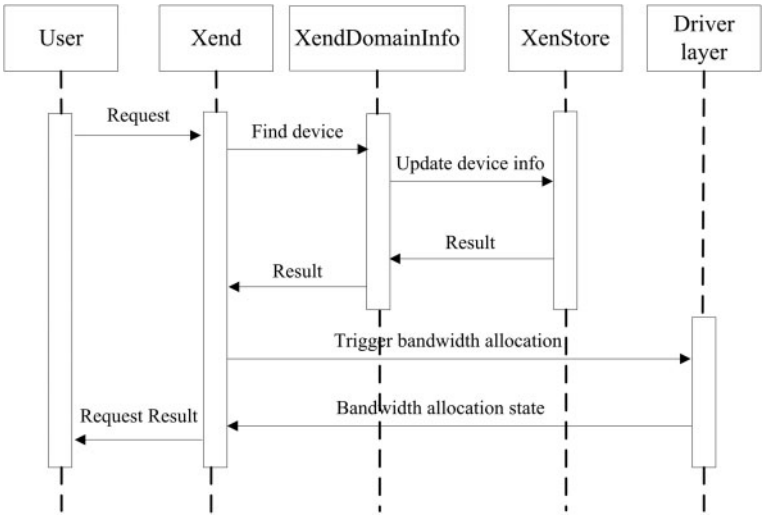

Figure 4. System workflow.

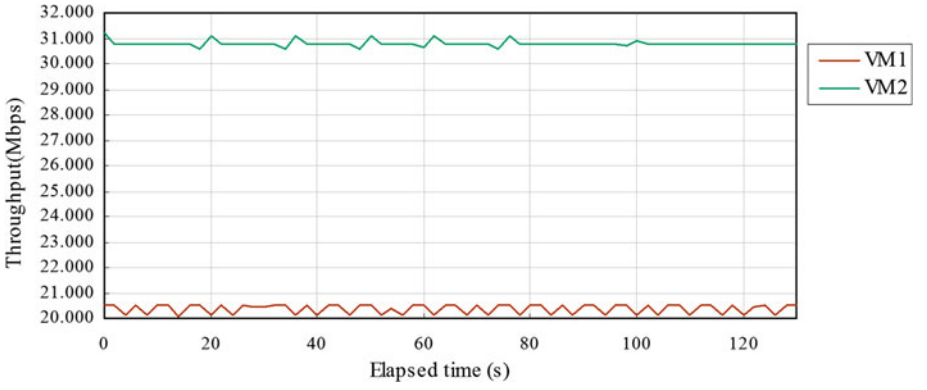

Figure 5. The curve of bandwidth allocation under general test.

5.1.1. Bandwidth allocation

This test shows the comparison of specified and measured network bandwidth. We use *DNBA* to limit the maximum network bandwidth of VM1 and VM2 to 20 and 30 Mbps, respectively.

From the test results, we observe that the average bandwidth of VM1 and VM2 is 20.322 and 30.776 Mbps, respectively. The measured network bandwidth is close to the specified value. The real-time bandwidth curve is shown in Figure 5. The results show that *DNBA* works well.

5.1.2. Dynamic bandwidth allocation

This test shows the effectiveness of *DNBA*. With the original network bandwidth allocation scheme of Xen, when a VM is running, users cannot change the maximum network bandwidth limit of the VM. However, *DNBA* can do this well. When the VM is running, users can adjust the maximum network bandwidth limit of the VM dynamically according to the demands of services.

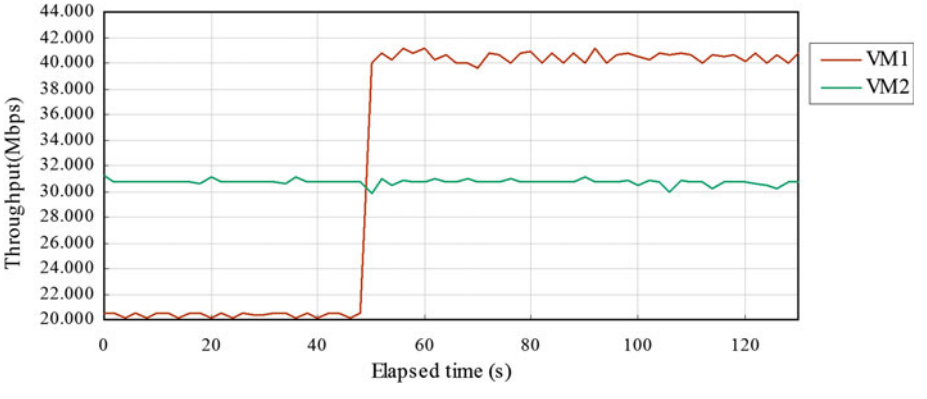

Figure 6. The curve of dynamic bandwidth allocation under general test.

When the test starts, we limit the maximum network bandwidth of VM1 and VM2 to 20 and 30 Mbps, respectively. We adjust the maximum network bandwidth limit of VM1 to 40 Mbps in the process of data transferring.

From the test results, we can see that the average bandwidth of VM2 is 30.744 Mbps and the minimum and maximum bandwidth of VM1 is 20.090 and 42.239 Mbps, respectively.

The real-time bandwidth curve is shown in Figure 6. From the curve, we can see that our bandwidth allocation is transparent to the service, which has no sign of interrupt.

5.2. Performance test

DNBA introduces some additional operations, which brings extra overhead costs. This test tells us how much extra costs are introduced by *DNBA*.

First of all, we launch a VM to test the total bandwidth of the system without any bandwidth allocation. The result is 93.716 Mbps. We use *rate′* to indicate the measured total network bandwidth in the following tests, and *rate* to represent the total bandwidth (i.e. 93.716 Mbps). We choose absolute error Δ and relative error δ as the performance metrics.

$$\Delta = |\text{rate}' - \text{rate}| \tag{1}$$

$$\delta = \frac{|\text{rate}' - \text{rate}|}{\text{rate}} \times 100\% \tag{2}$$

5.2.1. Bandwidth allocation

We use four VMs to conduct this experiment. All of them are running IxChariot's endpoint. For simplicity, we call them VM1, VM2, VM3, and VM4, respectively. We use *DNBA* to limit the maximum network bandwidth of VM1, VM2, and VM3 to 10, 20, and 30 Mbps, respectively. The network bandwidth of VM4 is not limited. It can use the rest of network bandwidth.

From the test results, we can see that the average network bandwidth of VM1, VM2, VM3, and VM4 is 9.850, 19.871, 29.213, and 35.056 Mbps, respectively.

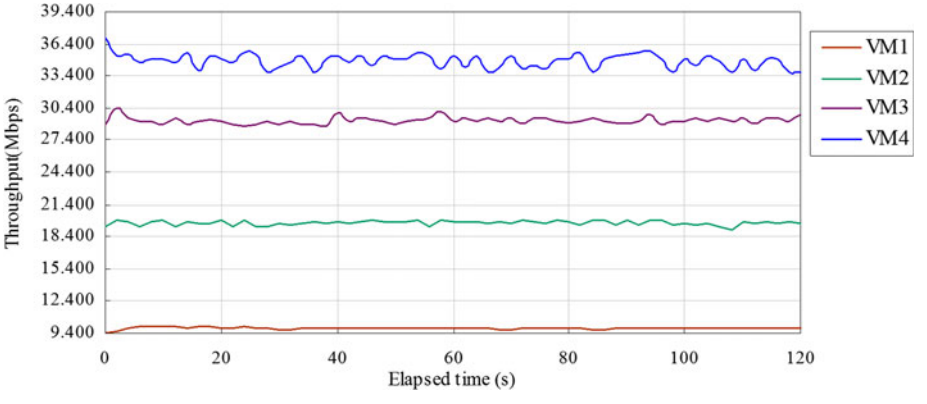

Figure 7. The curve of bandwidth allocation under performance test.

The total network bandwidth of these VMs is 93.596 Mbps (rate′). Therefore, Δ is 0.12 Mbps and δ is 0.128% according to (1) and (2). The performance loss is not obvious. The real-time bandwidth curve is shown in Figure 7.

5.2.2. Dynamic bandwidth allocation

We still choose VM1, VM2, VM3, and VM4 as testing VMs. At the beginning, we limit the maximum network bandwidth of VM1, VM2, and VM3 to 10, 20, and 30 Mbps, respectively. The network bandwidth of VM4 is not limited. It can use the rest of network bandwidth. Then, we start the test. In the process of data transferring, we change the maximum bandwidth of VM1, VM2, and VM3 three times. We first set the maximum network bandwidth of VM1, VM2, and VM3 to 20, 30, and 10 Mbps, respectively. After a while, we set the maximum network bandwidth of VM1, VM2, and VM3 to 30, 10, and 20 Mbps, respectively. Finally, we set the maximum network bandwidth of VM1, VM2, and VM3 to 10, 20, and 30 Mbps, respectively, a moment later.

The real-time bandwidth curve is shown in Figure 8. The total network bandwidth of these VMs is 93.456 Mbps (rate′). Therefore, Δ is 0.26 Mbps and δ is 0.277% according to Equations (1) and (2). This test introduces more operations, which results in more overhead. Although the overhead cost is higher than the previous test, the performance loss is still not obvious.

5.3. Streaming media server test

In order to guarantee the quality of video or audio services, the streaming media server must continuously transmit data across the network to the client. In this section, we test the quality of the streaming media server. We use Apple's open-source DSS as the streaming media server. DSS provides a tool named *StreamingLoadTool* to generate artificial client loads on the server, which greatly simplifies testing.

We use two VMs to conduct this test. One VM, named VM_DSS, runs DSS. The other, named VM_FTP, runs an FTP server. Moreover, we use two other machines as the clients. Both the servers and the clients are in the same LAN. We

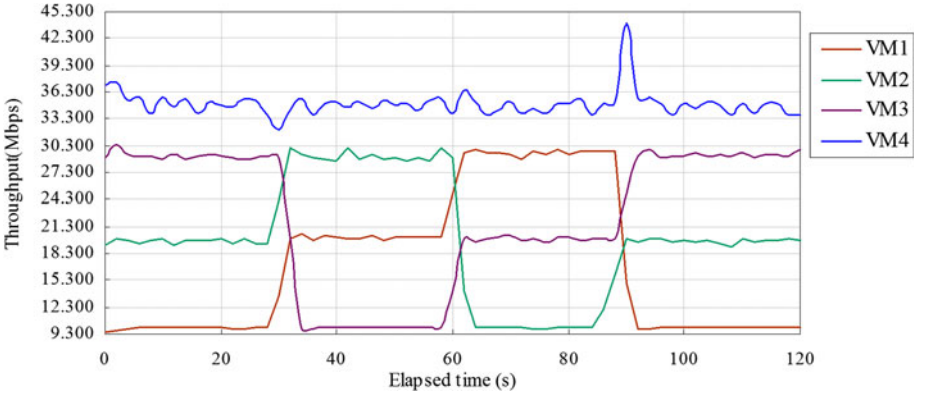

Figure 8. The curve of dynamic bandwidth allocation under performance test.

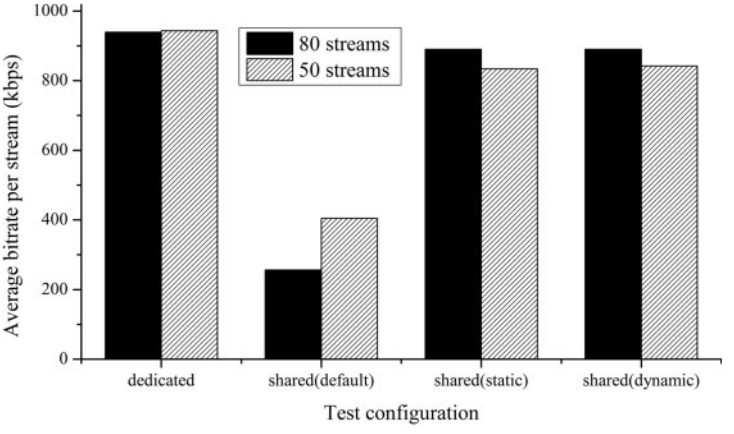

Figure 9. Average bit rate over 80 and 50 concurrent streams in a variety of configurations.

run *StreamingLoadTool* and *wget* on different clients to generate loads to DSS and the FTP server. We are interested in two primary metrics: the average bit rate per stream (the metric of DSS) and the total file transfer time (the metric of the FTP server) of downloading 1 GB file from the FTP server. To evaluate the average bit rate per stream, we start up several 1 Mbps streams of a movie file and run them for 60 seconds. In order to get the file transfer time, we use *wget* to download 1 GB file from the FTP server. We use four different configurations to conduct this test: dedicated, shared (default), shared (static), and shared (dynamic). The dedicated configuration means that there is no background interference. The shared configuration means that both VMs are running on the same server simultaneously, and we generate loads to DSS and the FTP server at the same time. The content in the bracket represents different bandwidth allocation policies

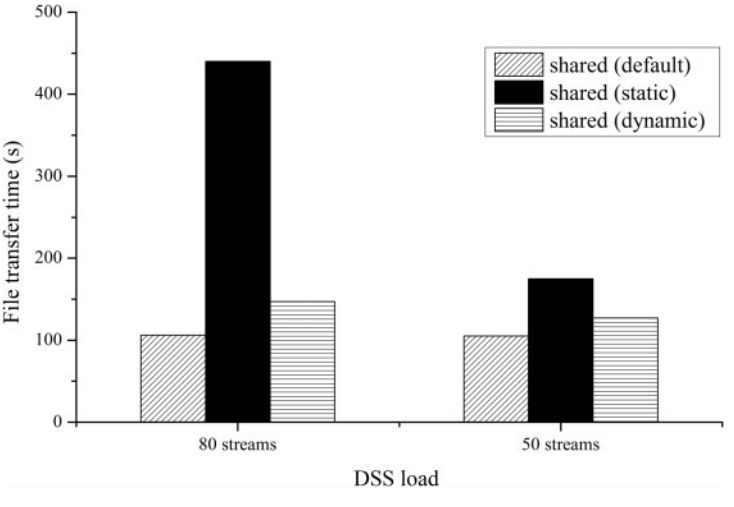

Figure 10. File transfer time under different DSS load with a variety of bandwidth allocation policies.

adopted in the test that are no bandwidth allocation, static bandwidth allocation, and dynamic bandwidth allocation, respectively.

Because the maximum bandwidth of the NIC is 100 Mbps, if two VMs with maximum network loads share the total network bandwidth, each can get 50 Mbps bandwidth ideally. Therefore, we run 50 concurrent streams to fully use the half of NIC's bandwidth. Besides, we run 80 concurrent streams as the high load to test the function of *DNBA*. In the static configuration, if we run 80 concurrent streams, we limit the maximum network bandwidth of VM_DSS and VM_FTP to 80 and 20 Mbps, respectively. Accordingly, if we run 50 concurrent streams, we limit the maximum network bandwidth of VM_DSS and VM_FTP to 50 and 50 Mbps, respectively. The maximum network bandwidth cannot be changed in the static configuration when the VMs are running. In the dynamic configuration, we allocate the same bandwidth before the starting of the test. We set the maximum network bandwidth of VM_FTP to 100 Mbps after the DSS test completion. We conduct each test three times and use the average value as the metrics. The test results are shown in Figures 9 and 10.

From the test results, we can see that both static and dynamic bandwidth allocation guarantee the QoS of the streaming media server well. However, the static method impairs the performance of other applications seriously. These applications cannot use extra network bandwidth even if there is a lot of network bandwidth available in the static configuration. Besides, if the network requirements of media servers are changed, users cannot adjust the maximum limit of network bandwidth on the fly. It means that the static bandwidth allocation cannot guarantee the QoS of media servers anymore. In this test, the file transfer time under the load of 80 concurrent streams in the static and dynamic bandwidth allocation is 440s and 147s, respectively. Therefore, *DNBA* is better and more flexible. It can maximize the system's overall performance while ensuring the QoS of media-based applications.

In summary, the experimental results show that the proposed solution in this paper can guarantee the QoS of media-based applications well and maximize the system's overall performance.

6. Conclusion

In this paper, we identify the network competition problem when VMs running media-based applications are consolidated with other VMs running network-intensive applications. And then, we present a dynamic network bandwidth allocation system in virtualized environment to address the problem, which allows administrators to allocate network bandwidth according to the network demands of applications when VMs are running. The system provides a user interface to administrators and APIs to programmers. Cloud providers can use the APIs to design and implement their own network bandwidth allocation strategies. Besides, the system is transparent to services running in VMs and does not interrupt the running services. This system can not only be used to guarantee the QoS of media-based applications but also be used to meet the quality of any applications that their performance has strong relationship with network bandwidth. The

experiment results show that our system can allocate network bandwidth among VMs flexibly and effectively and guarantee the QoS of streaming media servers well. Besides, our system can maximize overall performance of the system while ensuring the QoS of media-based applications.

In the future, we will design and implement some bandwidth allocation strategies to do dynamic adaptive bandwidth allocation by using our system, which can allocate network bandwidth automatically according to the historic bandwidth usage and some inferred policies.

Acknowledgment

The research is supported by National Science Foundation of China under grant Nos.61073024 and 61232008. It is also supported by National 863 Hi-Tech Research and Development Program under grant Nos.2013AA01A213 and 2013AA01A208, Outstanding Youth Foundation of Hubei Province under grant No.2011CDA086S, and Guangzhou Science and Technology Program under grant No. 2012Y2-00040.

References

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 *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, NY, USA, 19–22 October 2003, New York, NY, USA: ACM, pp. 164–177, 2003.

S.K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud", in *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems (MMSys'10)*, Phoenix, AZ, USA, 22–23 February 2010, New York, NY, USA: ACM, pp. 35–46, 2010.

F. Bellard, "QEMU, a fast and portable dynamic translator", in *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATC'05)*, Anaheim, CA, USA, 10–15 April, 2005, Berkeley, CA, USA: USENIX Association, pp. 41–46, 2005.

A. Bhattacharya, W. Wu and Z. Yang, "Quality of experience evaluation of voice communication: An affect-based approach", *Human-centric Computing and Information Sciences*, 2(7), pp. 1–18, 2012.

H. Cao, H. Jin, X. Wu and S. Wu, "Service flow: QoS-based hybrid service-oriented grid workflow system", *Journal of Supercomputing*, 53(3), pp. 371–393, 2010.

H. Chen, H. Jin, K. Hu and M. Yuan, "Adaptive audio-aware scheduling in Xen virtual environment", in *Proceedings of 2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA'10)*, Hammamet, Tunisia, 16–19 May 2010, Washington, DC, USA:IEEE Computer Society, pp. 1–8, 2010.

Credit Scheduler [Online]. Available online at http://wiki.xensource.com/xenwiki/CreditScheduler (Accessed 15 July 2012).

A. M. Elmisery and D. Botvich, "Enhanced middleware for collaborative privacy in IPTV recommender services", *Journal of Convergence*, 2(2), pp. 33–42, 2011.

S. Govindan, A. Nath, A. Das and B. Urgaonkar, "Xen and Co.: Communication-aware CPU scheduling for consolidated Xen-based hosting platforms", in *Proceedings of the 3rd International Conference on Virtual Execution Environments (VEE'07)*, San Diego, CA, USA, 13–15 June 2007, New York, NY, USA: ACM, pp. 126–136, 2007.

F. Hermenier, X. Lorca, J. M. Menaud, G. Muller and J. Lawall, "Entropy: A consolidation manager for clusters", in *Proceedings of the ACM/Usenix International Conference on Virtual Execution Environments (VEE'09)*, Washington, DC, USA, 11–13 March 2009, New York, NY, USA: ACM, pp. 41–50, 2009.

Y. Hu, X. Long, J. Zhang, J. He and L. Xia, "I/O scheduling model of virtual machine based on multi-core dynamic partitioning", in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC'10)*, Chicago, IL, USA, 21–25 June 2010, New York, NY, USA: ACM, pp. 142–154, 2010.

IxChariot – Official Website [Online]. Available online at http://www.ixchariot.com/ (accessed 15 July 2012).

H. Kim, J. Jeong, J. Hwang, J. Lee and S. Maeng, "Scheduler support for video-oriented multimedia on client-side virtualization", in *Proceedings of the 3rd Multimedia Systems Conference (MMsys'12)*, Chapel Hill, NC, USA, 22–24 February 2012, New York, NY, USA: ACM, pp. 65–76, 2012.

H. Kim, H. Lim, J. Jeong, H. Jo and J. Lee, "Task-aware virtual machine scheduling for I/O performance", in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'09)*, Washington, DC, USA, 11–13 March 2009, New York, NY, USA: ACM, pp. 101–110, 2009.

M. Lee, A. Krishnakumar and P. Krishnan, "Supporting soft real-time tasks in the Xen hypervisor", in *Proceedings of the 6th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'10)*, Pittsburgh, PA, USA, 17–19 March 2010, New York, NY, USA: ACM, pp. 97–108, 2010.

X. Liao, H. Jin and X. Yuan, "Espm: An optimized resource distribution policy in virtual user environment", *Future Generation Computer Systems*, 26(8), pp. 1393–1402, 2010.

J. Liu and B. Abali, "Virtualization polling engine (VPE): Using dedicated CPU cores to accelerate I/O virtualization", in *Proceedings of the 23rd International Conference on Supercomputing (ICS'09)*, Yorktown Heights, NY, USA, 8–12 June 2009, New York, NY, USA: ACM, pp. 225–234, 2009.

J. Liu, W. Huang, B. Abali and D.K. Panda, "High performance VMM-bypass I/O in virtual machines", in *Proceedings of the Annual Conference on USENIX'06 Annual Technical Conference (ATC'06)*, Boston, MA, USA, 30 May – 3 June 2006, Berkeley, CA, USA: USENIX Association, pp. 29–42, 2006.

H. Luo and M. Shyu, "Quality of service provision in mobile multimedia – a survey", *Human-centric Computing and Information Sciences*, 1(5), pp. 1–15, 2011.

D. Menascé, "Virtualization: Concepts, applications, and performance", in *Proceedings of the Computer Measurement Group's 2005 International Conference (CMG'05)*, Orlando, FL, USA, pp. 407–414, 2005.

R. Nathuji and K. Schwan, "Virtual power: Coordinated power management in virtualized enterprise systems", in *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, Stevenson, WA, USA, 14–17 October 2007, New York, NY, USA: ACM, pp. 265–278, 2007.

D. Ongaro, A.L. Cox and S. Rixner, "Scheduling I/O in virtual machine monitors", in *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'08)*, Seattle, WA, USA, 5–7 March 2008, New York, NY, USA: ACM, pp. 1–10, 2008.

P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant and K. Salem, "Adaptive control of virtualized resources in utility computing environments", in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'07)*, Lisbon, Portugal, 21–23 March 2007, New York, NY, USA: ACM, pp. 289–302, 2007.

D. Patnaik, A.S. Krishnakumar, P. Krishnan, N. Singh and S. Yajnik, "Performance implications of hosting enterprise telephony applications on virtualized multi-core platforms", in *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'09)*, Atlanta, GA, USA 7–8 July 2009, New York, NY, USA: ACM, 2009.

K.K. Ram, J.R. Santos, Y. Turner, A.L. Cox and S. Rixner, "Achieving 10 Gb/s using safe and transparent network interface virtualization", in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE'09)*, Washington, DC, USA, 11–13 March 2009, New York, NY, USA: ACM, pp. 61–70, 2009.

M. Rosenblum and T. Garfinkel, "Virtual machine monitors: Current technology and future trends", *IEEE Computer*, pp. 39–47, 38(5), 2005.

L. Shalev, J. Satran and E. Borovik, "IsoStack: Highly efficient network processing on dedicated cores", in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference (ATC'10)*, Boston, MA, USA, 23–25 June 2010, Berkeley, CA, USA: USENIX Association, 2010.

M. Stillwell, D. Schanzenbach, F. Vivien and H. Casanova, "Resource allocation using virtual clusters", in *Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'09)*, Shanghai, China, 18–21 May 2009, Washington, DC, USA: IEEE Computer Society, pp. 260–267, 2009.

A. Vakili and J. Gregoire, "Modelling the impact of the position of frame loss on transmitted video quality", *Journal of Convergence*, 2(2), 43–48, 2011.

T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Black-box and gray-box strategies for virtual machine migration", in *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI'07)*, Cambridge, MA, USA, 11–13 April 2007, Berkeley, CA, USA: USENIX Association, pp. 229–242, 2007.

A. Zhong, H. Jin, S. Wu, X. Shi and W. Gao, "Performance implications of non-uniform VCPU-PCPU mapping in virtualization environment", *Cluster Computing*, 16(3), pp. 347–358, 2013.