Isolation Schemes of Virtual Network Platform for Cloud Computing

SungWon Ahn, ShinHyoung Lee, SeeHwan Yoo, DaeYoung Park, Dojung Kim and Chuck Yoo
Dept. of Computer Science and Engineering, Korea University,
Anam-dong, Seongbuk-gu Seoul 136-713, South Korea
[e-mail: {swahn, shlee, shyoo, dypark, djkim, chuckyoo}@os.korea.ac.kr]
*Corresponding author: Chuck Yoo

Received April 9, 2012; revised July 7, 2012; revised September 6, 2012; accepted October 8, 2012; published November 30, 2012

Abstract

Network virtualization supports future Internet environments and cloud computing. Virtualization can mitigate many hardware restrictions and provide variable network topologies to support variable cloud services. Owing to several advantages such as low cost, high flexibility, and better manageability, virtualization has been widely adopted for use in network virtualization platforms. Among the many issues related to cloud computing, to achieve a suitable cloud service quality we specifically focus on network and performance isolation schemes, which ensure the integrity and QoS of each virtual cloud network. In this study, we suggest a virtual network platform that uses Xen-based virtualization, and implement multiple virtualized networks to provide variable cloud services on a physical network. In addition, we describe the isolation of virtual networks by assigning a different virtualized network ID (VLAN ID) to each network to ensure the integrity of the service contents. We also provide a method for efficiently isolating the performance of each virtual network in terms of network bandwidth. Our performance isolation method supports multiple virtual networks with different levels of service quality.

Keywords: Network virtualization, Performance isolation, Cloud computing, Xen

A preliminary version of this paper appeared in ICONI 2011, December 15-19, Sepang, Malaysia. This version includes a concrete analysis and supporting implementation results on the performance isolation scheme of virtualized network. This study was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0029848).

http://dx.doi.org/10.3837/tiis.2012.10.001

1. Introduction

The importance of cloud computing is increasing with the rising popularity of cloud services. To create a cloud computing service, virtual servers, virtual storage, and virtual networks are required [1]. Therefore, virtualization is the core technology used for cloud computing and has been studied in many fields. Virtualization can ensure diversity, which is a major advantage for virtualization of cloud computing, and allows various types of cloud services to coexist on a physical network. To make such advantages possible, among the different types of virtualization technologies available, network virtualization used for creating a virtual network is a particularly important technology for supporting cloud computing in future Internet environments. The design of a virtual network for cloud computing requires the consideration of many criteria, such as scalability, availability, reliability, flexibility, and utilization. In terms of diversity, network virtualization can help ensure that these criteria are selected suitably for cloud computing by supporting variable network topologies that facilitate various cloud services [2].

Network virtualization technology is used to create a virtualized network through both router and link virtualization. Router virtualization provides multiple independent logical routers on one physical router, whereas link virtualization provides multiple virtual links on one physical link. To create a virtualized network, virtualized routers are required, and the use of a software router has been suggested as one possible option. Recently proposed software routers that can overcome existing hardware limitations can also support many services such as content-centric network (CCN) [3] and content delivery network (CDN) [4]. However, the low performance of these routers remains an arguable problem. On the other hand, many studies on network virtualization have focused on large servers and storage systems [5]. There have also been a few analyses on network variety. However, there has been insufficient research on the coexistence of various virtualized networks for various cloud services with relatively low cost.

For the coexistence of various virtual networks to ensure the diversity of various cloud service types, two issues must be considered. The first issue is the isolation between virtualized networks. The second issue is a flexible performance guarantee. Isolation is required because the characteristics of each virtual network differ for different cloud services. Virtualized networks share physical resources. Thus, if each cloud service uses a different protocol, then isolation is required to ensure the integrity of each cloud service. Moreover, an adaptive bandwidth allocation scheme is also necessary to ensure a proper quality of service (QoS) for different content in a cloud-computing environment because each cloud service requires a different bandwidth.

In this paper, we suggest a virtual network platform for cloud computing and its schemes to support various cloud services independently on a physical network. We also suggest communication methods with effective bandwidth management to ensure better quality for each cloud service, and two isolation schemes for a virtualized network in a cloud environment: network isolation and performance isolation. Network isolation ensures the integrity of each virtual network isolated from other networks using a virtual LAN (VLAN) ID. The performance isolation scheme applied to an isolated virtual network then ensures the use of effective bandwidth resource management such as dynamic bandwidth allocation, weight-based bandwidth allocation, and channel bonding. We implemented these techniques for a Xen [6][7] environment, which provides flexible virtualization, and used an Intel 82599

10 G Ethernet card [8], which supports SR-IOV [9] as a hardware platform. We used SR-IOV to improve the throughput performance of the virtual router.

The rest of this paper is organized as follows. Section 2 reviews the background research conducted on this topic, such as the virtualization concept, testbeds with network virtualization, software routers, and performance-improving virtualization technology. Next, in Section 3, we describe the isolation concept for a virtualized cloud network used in our system, and Section 4 illustrates the isolation performance of this virtualized cloud network. In Section 5, we describe the evaluation results of our suggested schemes. Finally, we offer some concluding remarks and future directions for this research in Section 6.

2. Background Research

2.1 Virtualization

Virtualization is a technique used for efficiently controlling the interactions between users, applications, systems, and computing resources with an abstract physical specification. Virtualization can be applied to almost all items related to computers such as the CPU, memory, I/O devices, other hardware resources, and computer applications. Taking advantage of virtualization technology, the utilization of available resources can be maximized. For instance, a single piece of hardware or equipment can be used for several tasks. On the other hand, several pieces of hardware or equipment can perform a single task together. As a result, virtualization can guarantee a reduction of administrative costs. Such properties increase the flexibility and availability of the system. System virtualization can be divided into two parts: full-virtualization (hardware virtual machine (HVM)) and para-virtualization (PV). VM-ware [10] is an example of full-virtualization, and it does not require a modification of the guest OS. In addition, most works operate as software. However, full-virtualization has a low performance. Para-virtualization, on the other hand, overcomes the disadvantages of full-virtualization. A typical example of para-virtualization is Xen [6][7], which improves the performance of virtualization by modifying the guest OS code. Xen manages instructions using a hyper-call, which is similar to a system call. Therefore, there are no additional conversing operations during a guest OS operation. In this study, we use Xen for a virtualization environment to support greater flexibility and ensure a suitable performance.

2.2 Network Virtualization

Network virtualization allows the coexistence of multiple virtual networks on one physical network. Network virtualization can divide the practicalities of Internet service providers (ISPs) into infrastructure providers (InPs) and service providers (SPs). An InP manages the physical infrastructure, whereas an SP creates the virtual networks. Therefore, network virtualization also allows the coexistence of multiple cloud service providers on several heterogeneous virtual networks with isolation capability. Thus, these service providers can serve and manage customized cloud services for end users on virtual networks with an effective sharing and utilization of network resources leased from several infrastructure providers [2].

Current virtualization technologies for network use can be broadly divided into two categories: router virtualization and link virtualization [11][12]. The router virtualization technique provides multiple independent logical routers on a physical router platform using system virtualization. It creates multiple virtual routers from one physical router through a separation of resources. Link virtualization provides multiple virtual links on a physical link,

or integrated virtual links on multiple physical links. It provides a virtual network interface card (VNIC), which has several virtualized network interfaces in one physical network device. We implemented both types of virtualizations in our system.

2.3 Testbeds

Typical testbed studies include PlanetLab [13], NetServ [14], and GENI [15]. PlanetLab was developed to deploy geographically distributed network services with support for researchers and users. Its goal is to create a service-oriented network architecture that combines both a distributed system community and a network community. However, it does not support wireless or non-IP environments. In addition, it is difficult to change the network environment owing to the use of the virtualization of layer 3, which is the network layer. NetServ can be divided into two parts: a click module router and an OSGi framework. This architecture has the advantage of providing independent services and flexible changes in policy. However, a low performance is a specific problem inherent in this system. Finally, GENI is a project of the US NSF, the major goal of which is to build an open, large-scale, realistic experimental facility for evaluating new network architectures. It uses a programmable module block to implement the functions required for the virtualization of physical network resources. Testbeds are used to build a flexible virtual network environment, the foundation of which is network virtualization.

2.4 Software Routers

The benefit of virtualization is the elimination of hardware dependency through devices such as a software router. Software routers have been studied as a cost-saving solution, and can be applied to various platforms as an alternative to a hardware router. They have many advantages such as flexibility, manageability, and scalability. The software router architecture is configured using a software routing application on a hardware operating system. There have been several studies conducted on software routers. For instance, Click [16] ensures a freely configurable routing scheme by making it possible to change modules dynamically. OpenFlow [17] ensures the management of a dynamic routing flow table by separating the data and routing control. Moreover, XORP [18] can support various routing protocols such as unicast and multicast.

However, the long-standing problem of low-performance problem of software routers remains unresolved. Therefore, performance enhancement studies have helped in the development of devices such as PacketShader [19], which use a graphics processing unit (GPU) with hundreds of multi-cores. However, software routers do not ensure the complete isolation of virtualized networks in a shared resource environment where various protocols coexist. As a result, studies on isolation are necessary to support multiple isolated virtual networks for future Internet schemes. Such studies are also required to overcome the limitations of software routers. In this paper, we describe the importance of isolation and its implementation. Additionally, we suggest a method that can ensure the isolation of a network and stably manage various virtualized networks.

2.5 Network Isolation Methods

Isolation between coexisting virtual networks is essential for ensuring the integrity of each network. Several network isolation methods have been utilized in different projects. Among many isolation schemes, we focused on isolation schemes of link virtualization which is focused in this paper. Typical examples of encapsulation protocols include Ethernet frames to

IP packets (EtherIP) [20] and generic routing encapsulation (GRE) [21]. These protocols allow the use of a virtual private network with network isolation. However, they do not ensure performance isolation with adaptive bandwidth allocation.

Another method considered is multiprotocol label switching (MPLS) [22], which involves assigning a short fixed-length label to packets joining an MPLS cloud. Such a label is used to generate forwarding decisions. MPLS allows multiple networks to be separated quickly and effectively. Moreover, it provides various levels of QoS based on the class-of-service label. However, it has a restriction in that all network routers and switches must support MPLS. Although MPLS has several advantages as a candidate Internet backbone network, it also has certain disadvantages. The label distribution protocol (LDP) of MPLS, which is a core technique, is very complex and incomplete. This complex protocol can easily suffer from a lack of interoperability. In addition, as an incomplete protocol lacking explicit routing support, it requires additional protocol extensions.

In this paper, we used the IEEE 802.1Q standard [23] based on its suitability to the virtualization concept of our project. This mechanism uses a VLAN tag for dividing a virtual network. Ensuring isolation through a layer 3 routing protocol in multiple virtual network environments with adaptive bandwidth allocation is one of the differences with previous studies, as mentioned earlier.

2.6 SR-IOV

Single-root I/O virtualization (SR-IOV) is a new I/O virtualization technique using an Intel-VT [24]. This type of method utilizes an HVM, which solves the performance degradation problem through virtualization. SR-IOV is also supported by Xen. It uses an I/O memory management unit (I/O MMU) for an address-translation service. As background, SR-IOV assumes a driver domain that runs the device drivers, and a guest domain that runs the applications. Using SR-IOV, any guest domain can issue an I/O without additional performance overhead. SR-IOV allows guest domains to access and use a single physical PCIe I/O resource directly. To make this possible, SR-IOV supports both a virtual function (VF) and a physical function (PF). A VF has basic PCIe features and can serve PCIe data communication. A PF has full PCIe features and can control all of the PCIe I/O device functions. A guest domain can access I/O devices and receive data through a VF to minimize the virtualization overhead. A PF is generally used in the driver domain for device initialization, configuration, and control.

3. Virtual Network Platform for Cloud Services

3.1 Virtualized Networks

To provide diversity and coexistence to a cloud service on a single physical network, a virtualization technology is required, as mentioned previously. In addition, an isolated communication environment for each virtual cloud network is needed to ensure the QoS of each different cloud service. In this section, we suggest an isolated virtual cloud network using a Xen-based virtual router. The configuration of a virtual network for supporting various protocols is shown in **Fig. 1**. Each virtualized router of a virtual network platform can connect with another virtual network based on the required service, and can create multiple virtual networks on a single physical network. In **Fig. 1**, virtual cloud networks 1 and 2 share one physical network. In other words, it is possible to configure virtual cloud networks that support different protocols such as IPv4, IPv6, and CCN. These protocols coexist on a single router.

For example, virtual cloud network 1 provides a video streaming service using the CCN protocol, and a user terminal that wants to receive a video service can connect with this network. At the same time, another service using the IPv4 or IPV6 protocol can be provided to the user terminals through virtual cloud network 2 on the same physical network.

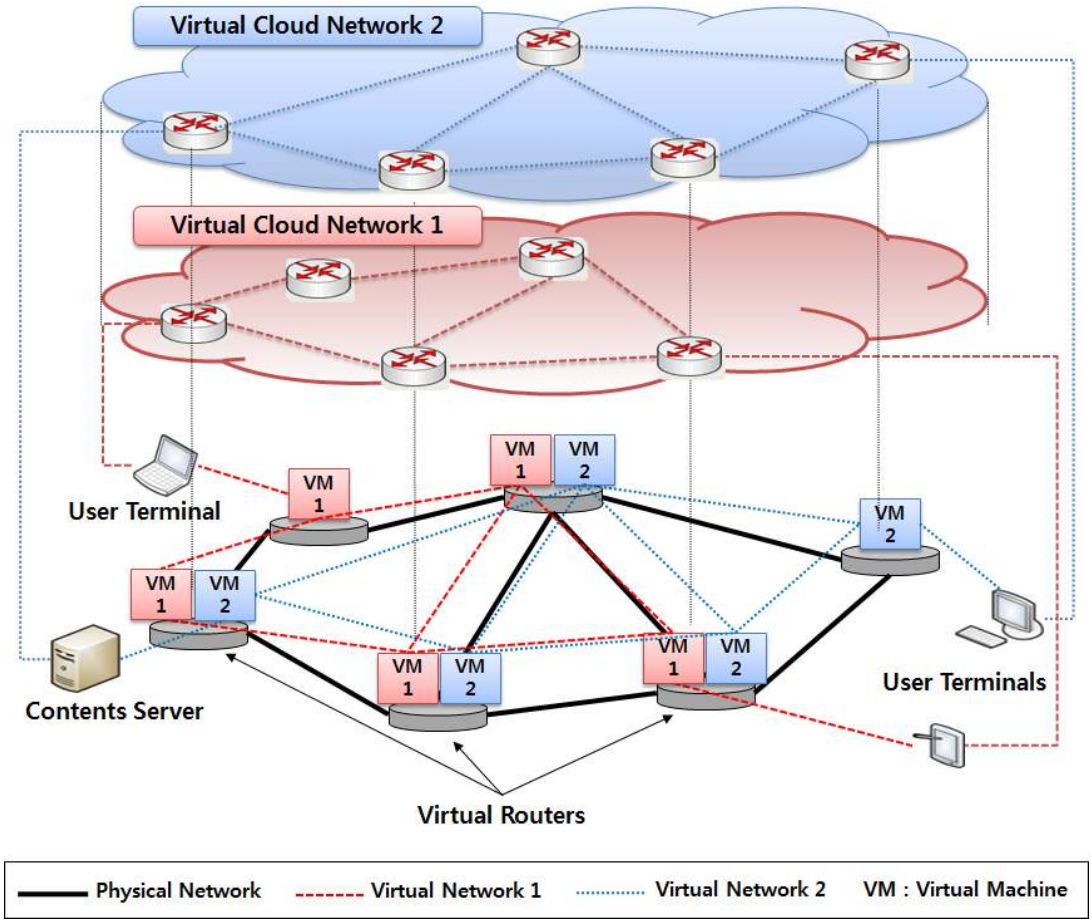

Fig. 1. Virtualized cloud networks

3.2 Virtual Network Platform

Fig. 2 shows the system configuration of a virtual network platform. We implement our system using Xen as a hypervisor. The flexible software router in **Fig. 2** is a virtual router. Each virtual router has a control plane and operates as a router with a specific routing protocol such as RIP or OSPF. These routers have an independent routing table and coexist on a physical router. Each virtual router (the flexible software router shown in **Fig. 2**) has its own virtualized network interface. The virtual network management module in the Xen hypervisor manages these virtual routers. It creates a new virtual router on demand and assigns the required bandwidth. In addition, it controls whether a received packet is sent, and if so, which virtual router it should be sent to. As shown in **Fig. 2**, the data flow of each virtual cloud network is controlled through a different VNIC within the system.

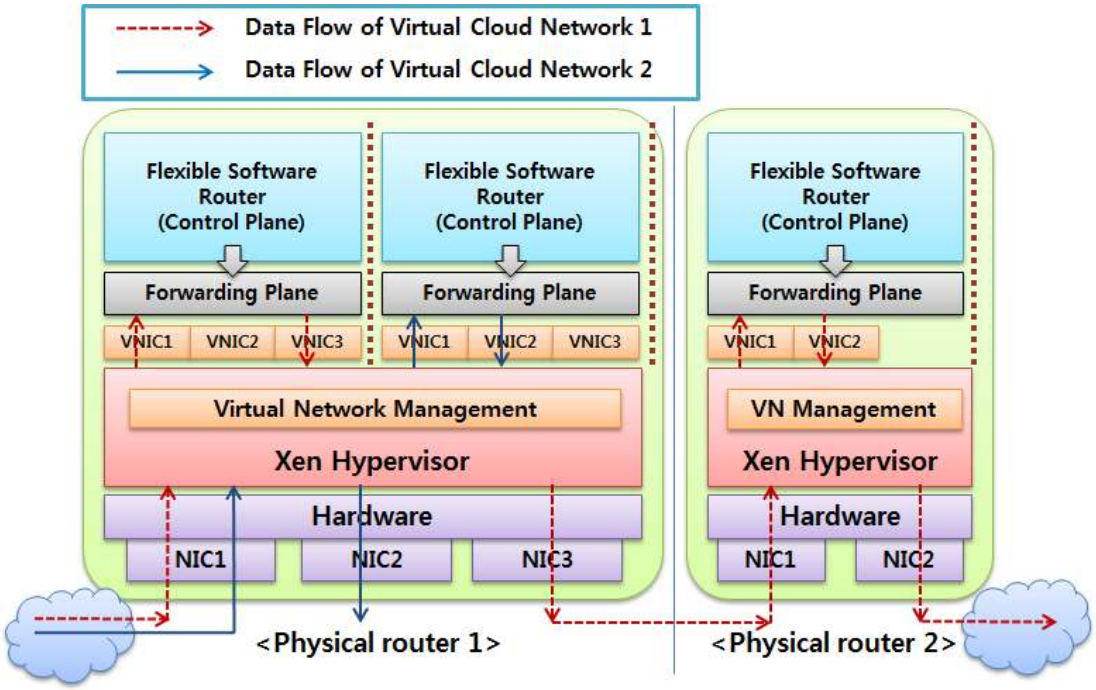

Fig. 2. System configuration of a virtual network platform

3.3 Network Isolation using VLAN ID

As we mentioned in Figs. 1 and 2, each virtual network shares one physical network. Although they use the same physical hardware, they must be operated independently. Fig. 3 shows the isolation of various virtual cloud networks, which is called network isolation. Network isolation indicates a perfectly isolated network that does not affect other networks. For instance, network isolation does not allow the inflow of a secure packet of VLAN 1, or the inflow of a worm virus packet of VLAN 2, to arrive at other virtual networks. In addition, each network should avoid infections, such as from a distributed denial of service (DDOS) attack. Each cloud service can be ensured a level of integrity and service quality through network isolation.

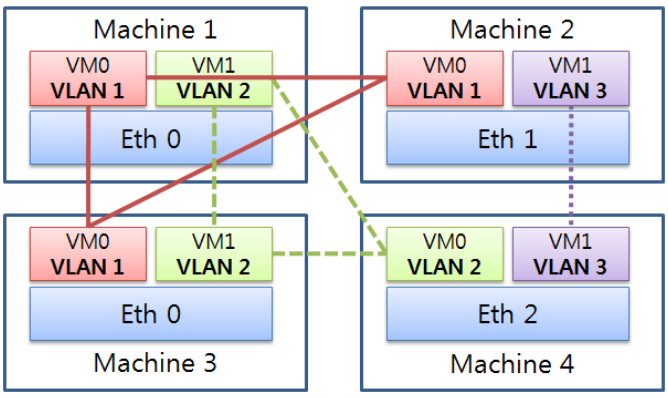

Fig. 3. Isolation of virtual networks

To secure the isolation of virtual networks, we gave a VLAN ID to each packet of each virtual network according to the service or protocol. Each virtual network has an inherent VLAN ID used to construct the network. Only virtual networks that have the same VLAN ID can communicate with each other. Therefore, if the VLAN ID of a virtual network differs from the common ID, the network cannot connect to the infrastructure of another virtual network.

In **Fig. 3,** each machine creates various virtual networks using the VMs in each machine connected to other VMs that have the same VLAN ID. VM0 in machine 1 has a VLAN 1 ID and connects to VM0 in machine 2 and VM0 in machine 3, which also have a VLAN 1 ID. In this same way, VLAN 2 is constructed along with VM1 in machine 1, VM1 in machine 3, and VM0 in machine 4. Each virtualized router receives several different VLAN IDs depending on each service and protocol. Virtualized router machine 1 shown in **Fig. 3** has two different VLAN IDs depending on two cloud services. It belongs to two different topologies at the same time. It also has two different routing tables.

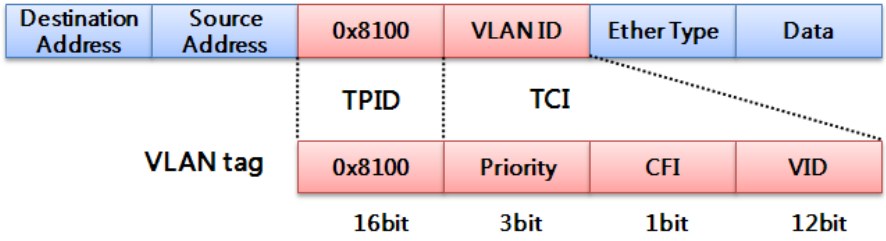

Fig. 4. VLAN ID packet format

To configure a VLAN ID, the packet format used is IEEE 802.1q, as shown in **Fig. 4**. The VLAN tag consists of TPID and TCI. TPID represents the existence of a VLAN tag. VID classifies VLANs using 12 bits other than 0 and 0 × FF. This structure inserts a VLAN ID when transmitting a packet. We configured this mechanism to insert a VLAN ID into an existing packet through the hardware.

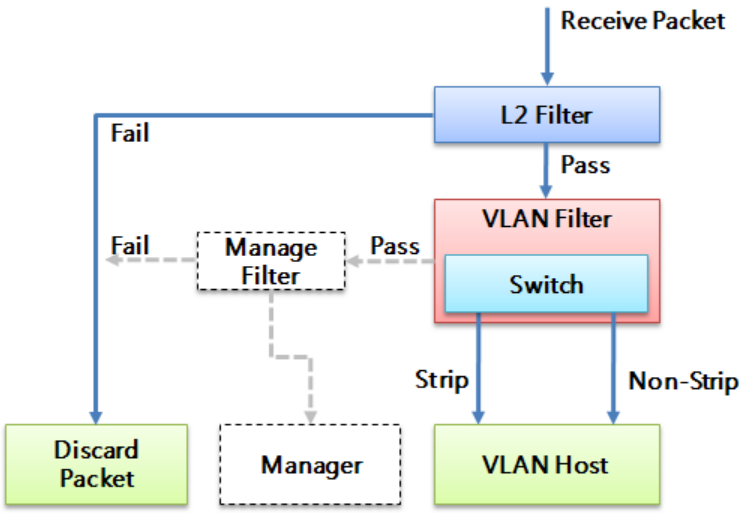

Fig. 5. VLAN ID packet filtering

The filtering process of a VLAN ID packet is shown in **Fig. 5**, which is an Rx flow of an Intel 82599 hardware platform chipset. When VLAN packets are received, the VLAN filter process classifies each one. First, the packet goes through the L2 filter. The VLAN filter in the hypervisor then checks the VLAN ID of the packet to determine which VM is the packet destination. Thus, a virtual router can know the virtual cloud network of the received packets. A packet of a particular virtual cloud network must be processed in that virtual router. The VLAN filter supports packet switching, which determines whether to strip the MAC header. A non-stripped packet will be delivered to the host in a VLAN, and a stripped packet will be delivered to a physical host. We configured this work to utilize hardware processing to improve the virtual network performance.

3.4 Improving the Throughput of a Virtualized Network with SR-IOV

Fig. 6 shows the router virtualization architecture of our system. A virtualized network is created using this architecture, which is configured for network and performance isolation. The hardware platform shown in **Fig. 6** is an Intel 82599 10 Gigabit Ethernet network interface card, and we use a Xen hypervisor as a virtual machine manager (VMM). There are several domains in the Xen hypervisor, Domain0 (Dom0), and Guest Domains (VM1, VM2, (VM: Virtual Machine)).

The NIC is separated into several virtual interfaces such as a PF and VF by SR-IOV, which supports virtualization. The PF is assigned to Dom0 directly in a PCI express (PCIe) supported environment. Therefore, the PF has an independent network interface with no influence from other domains. SR-IOV can create several VFs that have a relatively light PCIe function. VFs are assigned to each virtual machine, and they provide isolated communication, which is controlled by the VM. The PF and VF can work as a single dedicated NIC, which is controlled by each domain directly. Each VF sets the initial configuration through an I/O virtual machine (IOVM), which is the SR-IOV manager of Dom0. The VF then uses direct memory access (DMA) for communication after the initial configuration.

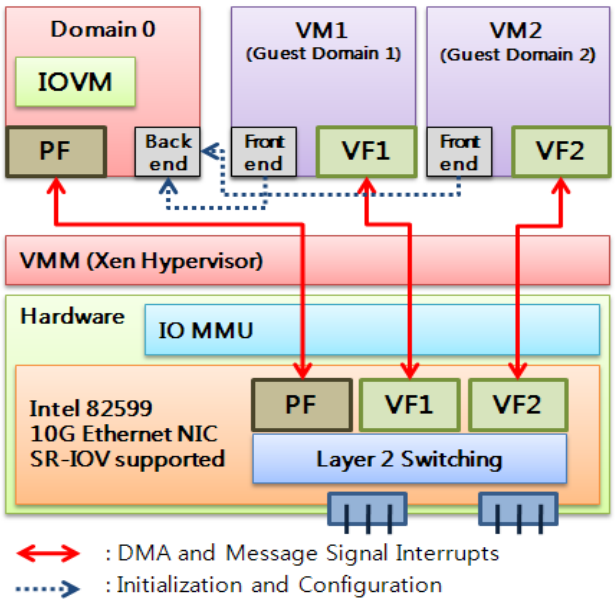

Fig. 6. Virtualized network interface with SR-IOV

4. Performance Isolation of Virtualized Cloud Network

4.1 Traffic Control Using BCN

A traffic controller for a virtual network platform is implemented by managing the bandwidth of each virtual cloud network. As previously mentioned, several virtualized networks coexist on one physical network. This means that the maximum bandwidth offered for a virtual network is the maximum value of the physical network. Therefore, several different bandwidths within the range of the maximum bandwidth are needed to accommodate each virtual cloud network. Some cloud services require a greater amount of bandwidth than other cloud services. For instance, the real-time video streaming service of a cloud network requires a large bandwidth. In this case, it requires bandwidth control and re-assignment. This technique is called performance isolation, which is the management and distribution of the available network bandwidth.

In a virtualized system architecture, as shown **Fig. 6**, the offered bandwidth of each domain is determined by how often the domain occupies the NIC. In this study, we implemented performance isolation by controlling the NIC occupancy rate of each domain. To make this technique possible, we use backward congestion notification (BCN) [8], which is supported in SR-IOV.

Fig. 7 shows the BCN rate scheduler used to apply performance isolation. The descriptor fetch arbiter allocates a packet to send to the assigned pool from the memory ring buffer. Pool0 and Pool1 in the VFs existing on the NIC are assigned to VM1 and VM2, respectively. BCN registers in the descriptor queue (DQ) are present inside each pool. The throughput and packet-buffer sharing values differ based on how the value of the register allocation is set. The maximum rate scheduler allocates less bandwidth assigned to the register. If the bandwidth is increased over the assigned value, the BCN bridge should be disconnected.

The order of packet transmission is essentially based on a round-robin method. A pool satisfying the register value receives a connection to the DQ and occupies the packet buffer and communication. This means that the packet of the virtual domain occupying the corresponding pool is transmitted through the NIC.

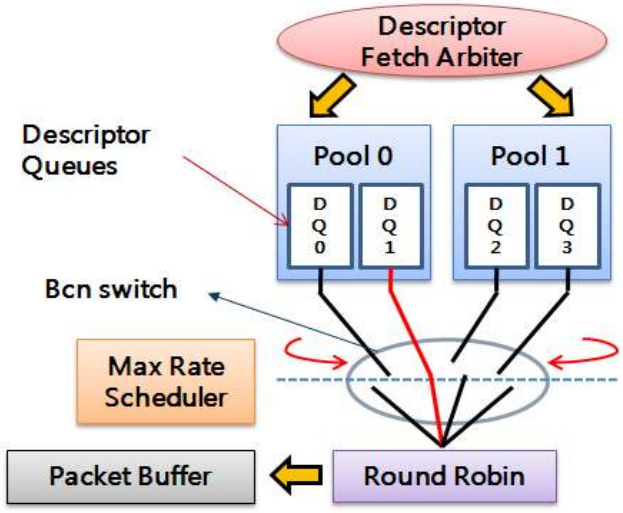

Fig. 7. Scheduling used for dynamic bandwidth allocation

4.2 Weight-based Bandwidth Control

To support flexible bandwidth allocation based on the demand of each cloud service, we suggest a weight-based bandwidth control (WBBC) algorithm on a virtual network platform. A virtual network platform controls the bandwidth to ensure fairness for each virtual network. **Fig. 8** shows the architecture of the WBBC algorithm on a virtual network platform. The bandwidth control module (BCM) checks the bandwidth use of a virtual network through the bandwidth statistic register (BSR) of the NIC. It wakes up periodically according to a timer. Next, it decides whether descriptor fetch to pool depending on the bandwidth usage information of the virtual network obtained from the BSR. If the descriptor does not fetch to pool, it cannot transmit its packet in the NIC. Therefore, a flexible bandwidth control is possible.

Determining the bandwidth allocation for each virtual machine is conducted based on the fetching of the descriptor from the BCM in the virtual machines as follows. The bandwidth use of virtual machine i (VMi) is Ti, and the weight of VMi is Wi. In addition, the bandwidth use of all virtual machines is Tsum, whereas the weight of all virtual machines is Wsum, which are shown in formula (1).

$$\frac{Ti}{Tsum} \leq \frac{Wi}{Wsum} \tag{1}$$

The left-hand side of formula (1) represents the rate of bandwidth use of VMi, whereas the right-hand side indicates the VMi weight rate for the sum of all weights. Thus, if the rate of bandwidth use of VMi is less than the allocated weight rate, the bandwidth is allocated by fetching packets to pool of the NIC. On the contrary, if the rate of VMi bandwidth use is larger than the allocated weight rate, a transmission is not allowed.

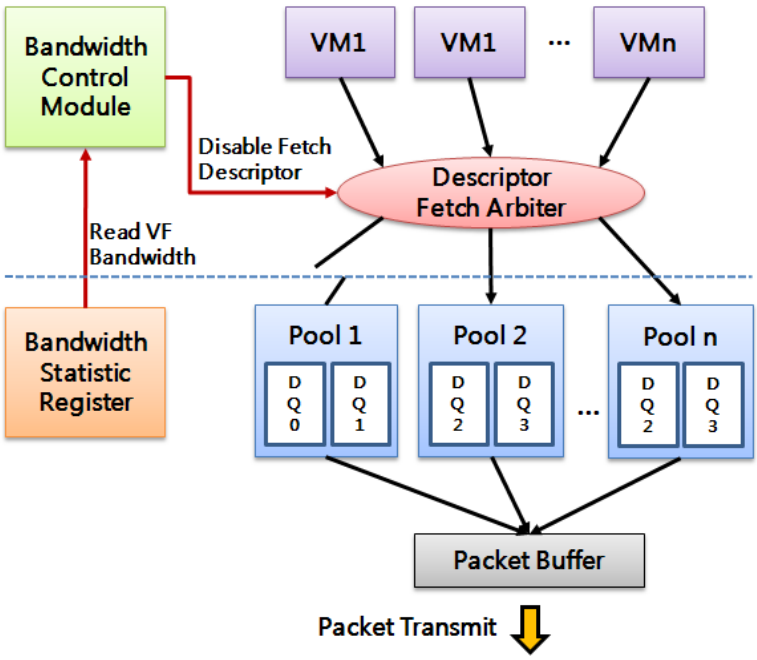

Fig. 8. Architecture of the WBBC algorithm

4.3 Virtual-Channel Bonding

The virtual network platform can support flexible bandwidth allocation through virtual-channel bonding. The virtual-channel bonding architecture is shown in **Fig. 9**. Each virtual network requires a different bandwidth allocation based on the service requirements. Even when multiple virtual networks are connected to the same NIC, different bandwidth allocations may be needed. We therefore changed the existing Linux bonding driver to support different types of bandwidth bonding in virtual machines.

As shown in **Fig. 9**, VM1 requires 12 Gbps of bandwidth. In this case, the V-channel module divides the bandwidth into 10 and 2 Gbps bandwidths. It then offers the required 2 Gbps bandwidth to create more VF links using another 10 Gb NIC. As such, it can support load balancing to alter the bandwidth. The V-channel module classifies packets depending on the amount of bandwidth. The bandwidth allocation is determined based on the rate of each bandwidth within the total bandwidth of each virtual network.

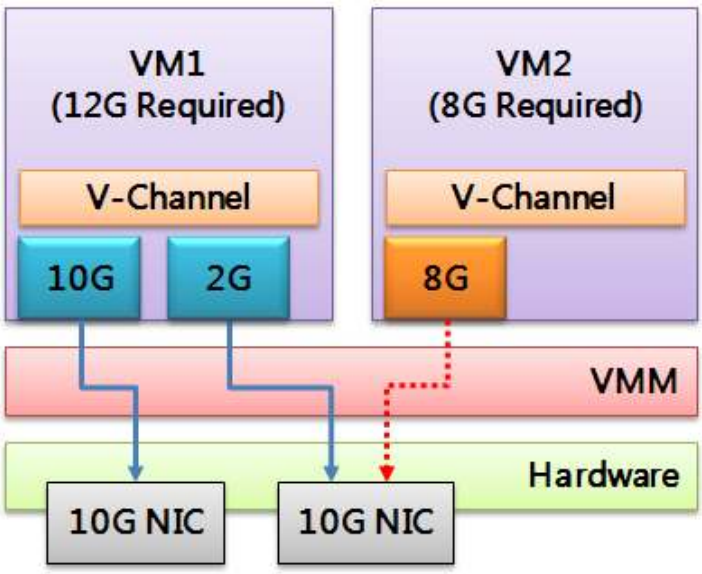

Fig. 9. Channel bonding architecture for a virtual network

The virtual network platform uses the port number and source and destination addresses in the virtual-channel bonding to minimize the problem of a mixed packet sequence for each session when arbitrarily allocating the NIC. The existing bonding driver does not know the amount of bandwidth provided by the NIC. Therefore, we implement a V-channel module that can receive bandwidth information provided to the NIC along with communication to the driver using ioctl.

In addition, the NIC driver of the virtual machine has to know how much bandwidth is assigned to it. For this information, the NIC driver of each machine sends its information regarding the bandwidth amount to the V-channel module through a message box in SR-IOV. When the V-channel module receives the initialization, it checks the allocated bandwidth of the VF. Next, if a change occurs, the VF drivers in each machine inform the V-channel module.

5. Evaluation

In this section, we discuss and verify the results of our experiment. The environment used for the performance validation is summarized in **Table 1**. In addition, **Fig. 10** shows the configuration of the experimental environment. The virtual network platform has two or three VMs (variable) and supports several virtual networks.

Table 1. Environment used for the experiment

Machine Specification		Virtual Machine Specification	
CPU	Intel XEON X5650(6-cores) x2	Virtual CPU	4-cores
Mem	12 GB	Mem	2 GB
Hypervisor	Xen 4.0	Guest OS	Ubuntu 10.04 LTS
		Kernel	Linux 2.6.37.1
NIC	Intel 82599 10 Gigabit (SR-IOV)	Virtual NIC	E1000 (Xen PV NIC model)
	Intel 82576 1 Gigabit (SR-IOV)		Igbvf (Intel 82576/82599 VF)
Sender/Receiver	Intel i7 CPU / Intel 82599 10G NIC, Intel 82576 1G NIC		

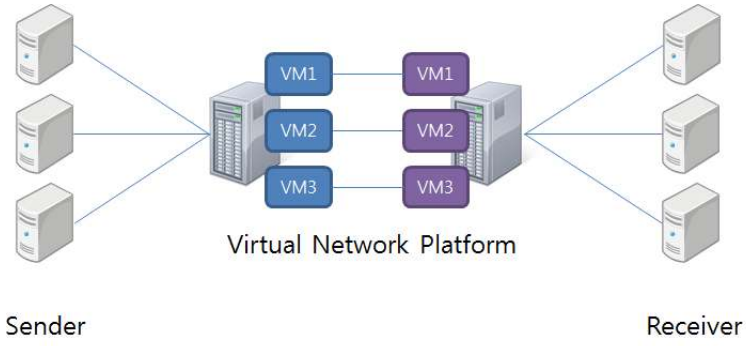

Fig. 10. Configuration of the experimental environment

5.1 Performance Improvement through SR-IOV

Table 2 lists the improvements in throughput performance for the virtual machine with and without the use of SR-IOV. We drew a comparison between a Xen PV NIC (E1000) and an Intel 82576 1 G NIC. E1000 basically supports a virtual NIC, which is emulated in Xen. This virtual NIC has 1 gigabit capacity. Therefore, we used an Intel 82576 1 G NIC, which supports SR-IOV, to compare the performance improvement of the SR-IOV. The results of the experiment are expressed in terms of bandwidth (Mbps) and packets per second (pps) for a packet size of 64 bytes, which is the minimum Ethernet packet size. We measured the performance using the Iperf [25] network benchmark for 300 s. As a result, there was a 25-fold performance enhancement compared to when SR-IOV was not used.

Table 2. Performance comparison with and without the use of SR-IOV

Configuration	Packet bytes	Throughput (pps)	Throughput (Mbps)
Xen PV (E1000)	64 Bytes	49,910 pps	25 Mbps
Xen PV with SR-IOV	64 Bytes	1,123,831 pps	575 Mbps

5.2 Performance Isolation Using BCN

Table 3 shows the results of the performance isolation among the VMs in an isolated virtual network with a 10 G NIC. State in **Table 3** indicates the method of bandwidth allocation for each VM. VM BCN Rate is the allocated bandwidth to the VMs, and Sum is a summation of the VM bandwidths. We divided the states into three types based on a bandwidth summation: non-saturated, saturated, and over-saturated state. We used pktgen [26] and set the traffic to constant bit rate (CBR) using 1500 bytes packet size. The results in **Table 3** show the measured bandwidth of each VM. The reached ratio to expected value shows that how close the measured value of bandwidth is to expected value, which is a measurement indicator of our proposed system.

Table 3. Performance isolation results

State	Configuration (Mbps)		Expected Total Value	Results			Reached Ratio to Expected Value (%)
	VM1 BCN Rate	VM2 BCN Rate	Rate Sum	VM1 BW (Mbps)	VM2 BW (Mbps)	BW Sum (Mbps)	
1:1	1000	1000	2000	995	995	1990	99.50
1:1	2000	2000	4000	1988	1994	3982	99.55
1:1	3000	3000	6000	2986	2973	5959	99.32
1:1	4000	4000	8000	3895	3869	7764	97.05
non-saturated	1000	3000	4000	996	2987	3983	99.58
non-saturated	1000	7000	8000	996	6887	7883	98.54
non-saturated	2000	1000	3000	1983	994	2977	99.23
non-saturated	2000	5000	7000	1994	4968	6962	99.46
non-saturated	3000	1000	4000	992	2985	3977	99.43
non-saturated	5000	3000	8000	4848	2992	7840	98.00
non-saturated	8000	1000	9000	7739	997	8736	97.07
saturated 1:1	5000	5000	10000	4811	4853	9664	96.64
saturated	2000	8000	10000	1990	7985	9975	99.75
saturated	3000	7000	10000	2986	6891	9877	98.77
saturated	4000	6000	10000	3873	5987	9860	98.60
saturated	8000	2000	10000	7806	1994	9800	98.00
over-saturated	3000	9000	12000	2412	7439	9851	98.51
over-saturated	5000	6000	11000	4507	5378	9885	98.85
over-saturated	5000	7000	12000	4809	4836	9645	96.45
over-saturated	8000	3000	11000	7199	2691	9890	98.90
over 1:1	7000	7000	14000	4825	4968	9793	97.93
over 1:1	8000	8000	16000	4783	4841	9624	96.24
over 1:1	9000	9000	18000	4895	4823	9718	97.18

The first case is non-saturated state, which occurs when the summation of the allocated bandwidth of each VM does not exceed the maximum capacity of the 10G NIC. In this case, if we allocate the bandwidth to each VM at the same rate, we achieve a resulting bandwidth rate

of 1:1. The summation of each bandwidth also shows a performance of at least a 97% or greater as compared with the expected bandwidth. For different rates, on the other hand, it works as allocated rate, which shows a similar performance result.

In the second case, a saturated state the summation of allocated bandwidths of each VM is 10 G. If the bandwidth rates of all VMs are equal, then the rate is 5000:5000. In this case, the result of each VM is close to the allocated rate, and the total throughput is close to the maximum capacity. Even if the bandwidth allocation rates differ, each VM operates using its allocated rate. Likewise, the total bandwidth reaches the maximum capacity of the NIC.

Finally, when the total allocated bandwidth exceeds the maximum capacity of the NIC, the maximum throughput does not exceed 10 G. Therefore, each VM operates at the allocated rate within a range that does not exceed the limitation whether the allocated bandwidth rates of all VM are the same. Consequently, the results of over-saturated state are similar to the saturation state and the maximum capacity is reached.

As shown in **Table 3**, performance isolation adjusts the bandwidth balance of each VM. Therefore, performance isolation can be ensured with the QoS of each virtual cloud network, and it manages bandwidth resources efficiently.

5.3 Weight-based Bandwidth Allocation

We configured three VMs, each with a bandwidth weight of 6:3:1, to show the fairness of a virtual cloud network using the WBBC algorithm in a 10 G NIC. The initial bandwidth of each VM is 1:1:1 until the bandwidth control module (BCM) is called, at which point the initial bandwidth rate begins to change. **Fig. 11** shows the change in bandwidth rate using the BCM. The bandwidth rate converges to a set weight as time passes. This mechanism is suitable for various cloud services that have different priorities based on the charging policy or popularity. Therefore, the WBBC algorithm used in our system can ensure flexible bandwidth allocation.

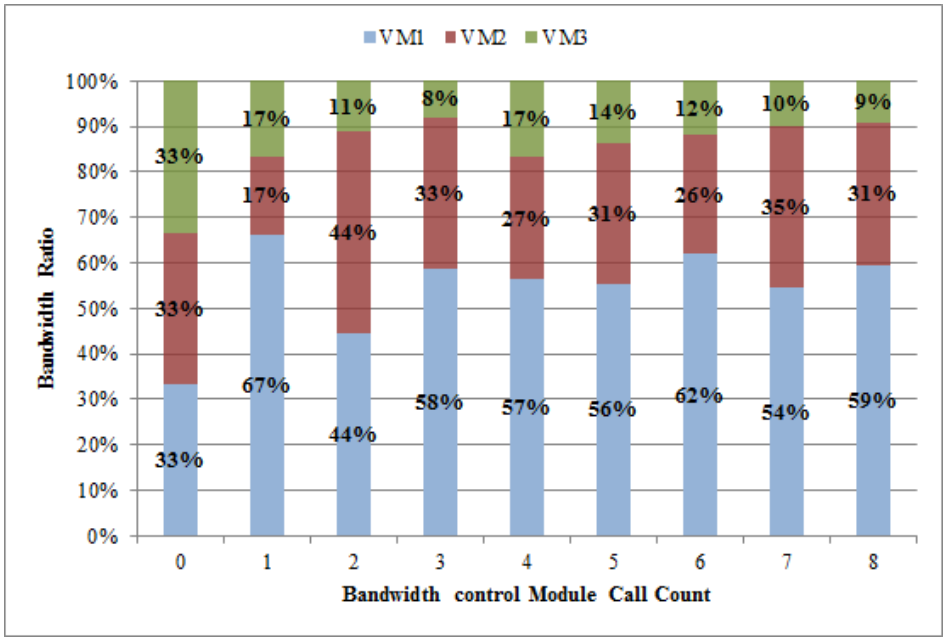

Fig. 11. Bandwidth rate control

Table 4 shows the throughput achieved through this mechanism. The total throughput bandwidths for each case approach nearly 10 Gbps as shown **Table 4**. The WBBC divides the bandwidth of each VM based on the pre-assigned weights. We can also see that the accuracies are very similar between the weight ratio and throughput bandwidth ratio. The largest error in the ratio of allocated weight to the resulting bandwidth is less than 4%, and the average total throughput is 9455 Mbps, which means the WBBC achieves a high performance with a maximum capacity of 94.55%. Therefore, we argue that our suggested method can allocate bandwidth efficiently based on this rate. Through this, our system can manage the bandwidth allocation of each VM based on the weight of each cloud service on each VM. This can be applied to the use of a virtual network for providing cloud services based on the importance or billing policy of each cloud service from the perspective of the ISP.

Table 4. Throughput based on the allocated weight

Allocated Weight	Throughput (Mbps)				Result of Bandwidth Ratio
VM1:VM2:VM3	VM1	VM2	VM3	Total	VM1:VM2:VM3
1 : 1 : 1	3144	3144	3144	9432	1 : 1 : 1
1 : 1 : 2	2383	2383	4695	9461	1 : 1 : 1.97
1 : 1 : 3	1902	1826	5725	9453	1 : 0.96 : 3.01
1 : 1 : 4	1587	1587	6316	9490	1 : 1 : 3.98
1 : 1 : 5	1341	1381	6732	9454	1 : 1.03 : 5.02
1 : 2 : 3	1576	3135	4711	9422	1 : 1.99 : 2.99
1 : 2 : 4	1355	2724	5366	9444	1 : 2.01 : 3.96
1 : 2 : 5	1177	2354	5909	9440	1 : 2 : 5.02
2 : 1 : 2	3772	1886	3772	9430	2 : 1 : 2
2 : 1 : 4	2696	1388	5356	9449	2 : 1.03 : 3.98
2 : 1 : 6	2094	1058	6273	9425	2 : 1.01 : 5.99
2 : 3 : 4	2093	3150	4185	9428	2 : 3.01 : 4
2 : 3 : 5	1890	2835	4706	9431	2 : 3 : 4.98
3 : 1 : 2	4728	1592	3152	9472	3 : 1.01 : 2
3 : 2 : 1	4758	3140	1586	9484	3 : 1.98 : 1
3 : 4 : 3	2847	3787	2856	9490	3 : 3.99 : 3.01
3 : 5 : 2	2826	4710	1894	9430	3 : 5 : 2.01
4 : 1 : 3	4732	1183	3573	9488	4 : 1 : 3.02
4 : 5 : 1	3792	4702	967	9461	4 : 4.96 : 1.02
5 : 1 : 2	5895	1167	2393	9456	5 : 0.99 : 2.03
6 : 3 : 1	5694	2819	958	9471	6 : 2.97 : 1.01
7 : 2 : 1	6657	1902	932	9491	7 : 2 : 0.98
8 : 1 : 1	7568	937	946	9451	8 : 0.99 : 1

5.4 Flexible Performance Using V-channel

Table 5 shows the total throughput of the channel bonding when our algorithm is implemented using two 10 G Ethernet NICs. When the allocated bandwidth is 20 G using the

two bonded 10 G NICs, the total throughput is 15.2 Gbps. This result is 76% of the expected value. The allocated bandwidth is 5 G in the same environment, resulting in 9 Gbps, which is 90% of the expected value of 10 G. The performance did not approach the expected value. Based on our analysis of previous studies, we believe the reason for this shortcoming in terms of the expected performance is insufficient hardware support [27].

Table 5. Throughput based on the allocated bandwidth in V-channel

Allocation of Bandwidth Bonding	Total Throughput (Gbps)
10 G – 10 G	15.2 Gbps
5 G – 5 G	9.0 Gbps

We conducted another experiment on the V-channel by dividing the throughput rate in one VM through two 10 G NICs. **Table 6** shows the results. We configured a 9 Gbps bandwidth, which is assumed to be the required bandwidth of a VM for certain cloud services. We did not go over 10 G as the hardware cannot support full capacity saturation of the bonded NICs, as shown in **Table 5**. We also configured a packet size of 64 bytes. **Table 6** shows the throughput based on the configuration of the interface when limited to 3 Gbps and 6 Gbps in a 1:2 ratio. In other words, a virtual cloud service (VM) requires a 9-Gbps bandwidth, and a V-channel system divides this bandwidth into 6 and 3 Gbps in a 2:1 ratio. The V-channel system then uses 6 Gbps through Eth1 (NIC1) and 3 Gbps through Eth2 (NIC2). Eth1 and Eth2 provide a 2:1 bandwidth ratio, where the throughputs are 9375 and 5664 Kpps, which is the same 2:1 ratio as we used.

Table 6. Results from dividing the bandwidth between two NICs using V-channel.

Interface	Allocated Bandwidth (Gbps)	Packet Size	Throughput (Kpps)
Eth1(NIC1)	6Gbps	64bytes	9375Kpps (4800Mbps)
Eth2(NIC2)	3Gbps	64bytes	5664Kpps (2900Mbps)

6 Conclusion and Future works

A network virtualization technology is the key technology used to support cloud computing. In this paper, we suggested a virtual network platform to support various cloud services independently on a physical network. In addition, we suggested communication methods with effective bandwidth management to ensure a better quality for each cloud service. Virtualization technology can provide diversity, which is a major advantage for cloud services. In terms of diversity, this advantage can satisfy the many requirements of cloud computing such as scalability, availability, reliability, and flexibility. To ensure these requirements of cloud computing, network isolation is required, and we therefore implement a network isolation scheme using VLAN ID.

The network isolation indicates the existence of perfectly isolated virtual networks among other virtual networks. Using such isolation, we can expect a high integrity of each cloud service on several virtual networks. A suitable performance of each cloud service should be ensured to provide the QoS in an isolated network. Thus, we suggest the use of performance isolation, which assures a dynamically allocated bandwidth to each virtual cloud network. We implemented several performance isolation techniques including dynamic bandwidth allocation using BCN, weight-based bandwidth allocation, and virtual-channel bonding. As

we showed previously, a balanced bandwidth allocation for a better quality of service can be granted through performance isolation.

In this study, we implemented our schemes on a Xen hypervisor, which can provide flexible virtualization. We also used an Intel 82599 10 G NIC as a network interface. Moreover, we utilized the SR-IOV technique to improve the throughput performance of our virtual router platform. For future research, hardware support and optimization techniques are required to improve the V-channel performance. We will also perform experiments on transmitting real data in conjunction with a large-scale test bed such as KREONET [28]. We expect that this current study will be a useful testbed platform for further Internet research on cloud computing.

References

[1] Zhang Q, Cheng L, Boutaba R (2010) "Cloud computing: state-of-the-art and research challenges." *Journal of Internet Services and Applications*, vol. 1, issue 1, pp 7-18, May 2010. Article (CrossRef Link)

[2] N. Chowdhury, "A survey of network virtualization," *Computer Networks*, Jan. 2010. Article (CrossRef Link)

[3] Van Jacobson , Diana K. Smetters , James D. Thornton , Michael F. Plass , Nicholas H. Briggs , Rebecca L. Braynard, "Networking named content", In *Proc/ of the 5th international conference on Emerging networking experiments and technologies*, Dec. 2009. Article (CrossRef Link)

[4] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68- 74, 2003. Article (CrossRef Link)

[5] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," In *Proc. of the 2008 ACM CoNEXT Conference*, p. 20, 2008. Article (CrossRef Link)

[6] 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. of the nineteenth ACM symposium on Operating systems principles*, Oct. 2003. Article (CrossRef Link)

[7] Xen homepage, http://xen.org

[8] Intel Corporation, Intel® 82599 Gigabit Ethernet Controller Datasheet. http://www.intel.com

[9] Y. Dong, et al. "SR-IOV Networking in Xen: Architecture, Design and Implementation". *1st Workshop on I/O Virtualization*, San Diego, CA, 2008. Article (CrossRef Link)

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

[11] Tutschku, K., Zinner, T., Nakao, A., & Tran-Gia, P. "Network virtualization: implementation steps towards the future internet". In *Proc. of the workshop on overlay and network virtualization at KiVS*, Kassel, Germany, Mar. 2009. Article (CrossRef Link)

[12] N. Chowdhury, "Network virtualization: state of the art and research challenges," *Communications Magazine,* vol 47, no. 7, pp. 20-26, Jan. 2009. Article (CrossRef Link)

[13] B. Chun, D. Culler, and T. Roscoe, "PlanetLab: An Overlay Testbed for BroadCoverage Services", *ACM SIGCOMM Computer Communcation*, Vol. 33, Issue 3, Jul. 2003. Article (CrossRef Link)

[14] NetServ, NSF project, http://www.cs.columbia.edu/irt/project/netserv

[15] GENI (Global Environment for Network Innovations), NSF project, http://www.geni.net

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, p. 263-297, Aug. 2000. Article (CrossRef Link)

[17] N. McKeown et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69-74, Mar. 2008. Article (CrossRef Link)

[18] M. Handley, O. Hodson, and E. Kohler, "XORP: an open platform for network research," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, p. 53-57, Jan. 2003. Article (CrossRef Link)

[19] Sangjin Han, Keon Jang, KyoungSoo Park and Sue Moon, "PacketShader: a GPU-accelerated Software Router,"*In Proc. of ACM SIGCOMM 2010*, Delhi, India. September 2010. Article (CrossRef Link)
[20] R. Housley , S. Hollenbeck, "EtherIP: Tunneling Ethernet Frames in IP Datagrams," RFC 3378, 2002. Article (CrossRef Link)
[21] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic routing encapsulation (GRE)," RFC 2784, *Internet Engineering Task Force*, Mar. 2000. Article (CrossRef Link)
[22] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001. Article (CrossRef Link)
[23] ANSI/IEEE Standard 802.1Q, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", IEEE 802.1q, 1998. Article (CrossRef Link)
[24] Intel Virtualization Technology. Article (CrossRef Link)
[25] Iperf version 2.0.5, http://sourceforge.net/projects/iperf/?abmode=1
[26] Pktgen, http://www.linuxfoundation.org/collaborate/workgroups/networking/pktgen
[27] Aust, S.; Jong-Ok Kim; Davis, P.; Yamaguchi, A.; Obana, S.; , "Evaluation of Linux Bonding Features," *Communication Technology, 2006. ICCT '06. International Conference on* , vol., no., pp.1-6, 27-30, Nov. 2006 Article (CrossRef Link)
[28] KREONET, http://www.kreonet.net/

Sung-Won Ahn received B.S. degrees in department of Computer Science, Korea University in 2006, and M.S degrees in department of Computer Science and Engineering from Korea University, Seoul, Korea in 2008. Since 2008, he has been studying his Ph.D. degree in Computer Science and Engineering from Korea University, Seoul, Korea. His current research interests include Network topology, Cloud computing, Virtualization and Media streaming.

ShinHyoung Lee received B.S. degree in earth environment science from Korea University, Seoul, Korea, and an M.S. degree in computer science from Korea University. His research interests include wireless networking, multimedia streaming, and network virtualization.

Seehwan Yoo received B.S. and M.S. degrees in computer science from Korea University, Seoul, Korea, in 2002 and 2004, respectively. He is currently a Ph.D. candidate at Korea University, Seoul, Korea. His current research interests are in embedded and mobile systems virtualization and real-time scheduling.

DaeYoung Park received B.S. degrees in department of Computer Science, Korea University, Seoul, Korea in 2010. Since 2010, he has been studying his M.S. degree in Computer Science and Engineering from Korea University, Seoul, Korea. His research interests include Cloud computing and Virtualization and network virtualization.

Dojung Kim received B.S. degrees in department of Computer Science, Korea University, Seoul, Korea in 2010. Since 2010, he has been studying his M.S. degree in Computer Science and Engineering from Korea University, Seoul, Korea. His current research interests include network virtualization and media streaming.

Chuck Yoo received B.S. and M.S. degrees in electronic engineering from Seoul National University, Seoul, Korea, and an M.S. degree and Ph.D. in computer science from University of Michigan. He worked as a researcher in Sun Microsystems Laboratory from 1990 to 1995. He is now a professor in the Department of Computer Science and Engineering, Korea University, Seoul, Korea. His research interests include high performance network, multimedia streaming, operating systems, and virtualization. He served as a member of the organizing committee for NOSSDAV 2001.