A Research on the Integrated Virtual Platform for Managing Multiple Services

SATOSHI KODAMA
Tokyo University of Science
Department of Information Sciences
2641 Yamazaki, Noda-shi, Chiba-ken
JAPAN
kodama@is.noda.tus.ac.jp

REI NAKAGAWA
Tokyo University of Science
Department of Information Sciences
2641 Yamazaki, Noda-shi, Chiba-ken
JAPAN
j6316627@gmail.com

TANOUCHI TOSHIMITSU
Tokyo University of Science
Department of Information Sciences
2641 Yamazaki, Noda-shi, Chiba-ken
JAPAN
6316625@ed.tus.ac.jp

Abstract: With the development of the computing architecture, the virtual technology has been widely infiltrating the whole network infrastructure and showing the future vision (e.g., SDN (Software Defined Network), NFV (Network Function Virtualization)). There is a general tendency to integrate services being formerly operated by multiple computing divisions into the collaboration system using the correct working of virtualized multi-environment. For example, VM (Virtual Machine) enables the multi-OS environment on single-OS and the reduction of physical resources such as an occupation area, a power consumption, a restriction of hardware, and so on. However, there remain the problems of the complexity around the dependencies between the VM and the host single-OS, (e.g., the resources utilization by multi-OS and the consistent management of these). For example, if we would like to build some Apache servers on single-OS, we implement each Apache server on each VM. There are several hot topics around the challenges of VM (e.g., OS migration, the optimization of the energy consumption). Here we consider the virtualization to skip VM. We design the server architecture providing the services identified by the port number. This system aims the programmable virtual node which is called Virtual Control Space (VCS) and provides the multiple server systems on Cent OS 7.2. The VCS works require virtualized Network Interface Card (vNIC) which is non-standard Linux extensions. We executed the original architecture including the VCS on the regular PC and confirmed whether our system normally manages the services which are divided by the port number by some of the steps. Finally, we have achieved the multiple server systems on single-OS without VM.

Key–Words: Cross-Layer, Network Architecture, Software Defined Network

1 Introduction

In recent years, as a rapid rise of the computing technologies, a diverse number of network services are deployed on the network infrastructure. This situation has been causing the problems due to the proprietary nature of existing hardware appliances, the cost of offering the space and energy for a variety of middleboxes, and the lack of skilled professionals to integrate and maintain these services. There is a kind of the saturation state which requires the innovative network system which is completely separated from the existing system. The researchers aim the provision of a service developed by the software infrastructure separating from a hardware one[1, 4, 5, 14, 15]. There are some of the innovative technologies which are the base of our research significance. We have been taking a close look at the virtualization for managing multiple services in referring to these technologies. For example, VM is the favorite tool to build the virtualized environment allowing multiple OS to coexistence with each other on single-OS. VM has a multi-forms, hosted model and supervised model) and provides many solutions (e.g., Consolidation, Shared CPU, Memory, NIC, Disk, Centralized Management, Migration and Less space). That can also be said that it is an OS-level virtualization. The related research is very hot and showing the further progress[6, 16]. We have concluded about the nature of VM through these research trends; VM has a hot side as an innovative technology; on the other hand, there is a severe and unstable condition to make use of the basement for our research because it has not yet been clarified. In other words, this issue is the adjusting of the difference caused by each of existing OS. The multi-OS environment for multi-services is deployed for the multi-objects and has the multi-requirements. Indeed, the OS-level virtualization provides the scope of the almighty management on the computer foundation. However, there are open to discussion about the scope of the virtualization which is more optimized than the scale of an OS-level. Therefore, we designed the new server system on single-OS. It has an integrated vir-

tual platform for managing multiple services on virtual Network Interface Card (vNIC). Furthermore, the new system enables the various server systems without the multiple OS environments as is the case of VM.

1.1 Principles of Designing System Abstraction

As a result of the discussion about what is the most efficient component for our concepts, we have concluded the network virtualization by SDN and NFV. These technologies gain a high reputation as a solution to many complex services baked into the network infrastructure[2, 8, 9, 10, 12, 13]. N.M.Mosharaf Kabir Chowdhury and Raouf Boutaba mentioned the future architecture for the Cloud Computing, called Open Application Delivery Network (OpenADN)[2]. That architecture has multiple virtual hierarchies which consist the nest of virtual nodes on the network infrastructure and exploits the mutual networking among the virtual nodes. They indicated that most application could easily get computing and storage facilities using the networking by multiple virtual nodes. Besides Raj Jain and Subharthi Paul mentioned the architecture for the virtualization around NIC (Network Interface Card)[7]. In their study, they took up the three architectures of the virtualization around NIC. Especially in these; we aim the design providing virtual NIC (vNIC) as a software fundamental via the supervisor, which uses the Virtual Ethernet Bridge (VEB) like the tunneling on the encapsulation. We design the system abstraction through these researchers. Firstly, we set the design goal as below.

(i) Flexibility
The network virtualization must provide a freedom in every aspect of networking. In other words, on the physical infrastructure, each of the virtualized components should be free to implement customized control protocols.

(ii) Manageability
By separating the elements for management from a physical infrastructure, each of the separating parts is individually virtualized to modify the functions and to introduce the accountability of itself every layer of networking.

(iii) Scalability
The coexistence of multiple virtualized components ensures the scalability to support an increasing number of the coexisting virtual component without disturbing their performance.

(iv) Programmability
The lowest layout of the computer network architecture is composed of the programmable fundamentals. These are an origin of the modern networking services and the robustness derived from the functional consistency. Therefore, root regression is one of the most important requirement. Programmability extends significantly the capacity to access and customize.

(v) Sharing
Today ' s computer devices are optimized for the multiple tasks; that enables the concurrent operation of the multiple virtual components. These can be used by a different user and the efficient utilization of the computer resources.

Secondary, we design the outline of the system abstraction which has a high affinity with these design goals in Fig. 1.
There are important some properties which are " Infrastructure Layer," " Virtual Hierarchy " and " Service Orchestrator." " Infrastructure Layer " describes the networking of the hardware device which maintains the virtual nodes as service components. " Virtual Hierarchy " represents the virtual space which is prepared to deploy the NF including "Service Orchestrator." NF means the operation of the virtual node. It ' s software service, not an application. " Service Orchestrator " describes the management tool for the service worked on "Virtual Hierarchy." For example, in the case of HTTP application protocols represented by 80 number of the port on " Virtual Hierarchy #1, " the customized Apache system works on "Service Orchestrator, " and its server operates on the data folder of " Virtual Hierarchy #1. " These properties provide the following principles[3].

(i) Coexistence
The coexistence of multiple NF and " Service Orchestrator " enables the definition of the several application policies. Each application policy referring the NF and " Service Orchestrator " from different " Virtual Hierarchy " can coexist together.

(ii) Recursion
Each NF and " Service Orchestrator " on " Virtual Hierarchy" can consist the different "Virtual Hierarchy." This work makes the hierarchical relationship between these " Virtual Hierarchy ", like parent and child. Its form is recursion. The nest of the parent-child relationship enables the sequential operation referring the top of the nest and ensures the reliability of the application including availability.

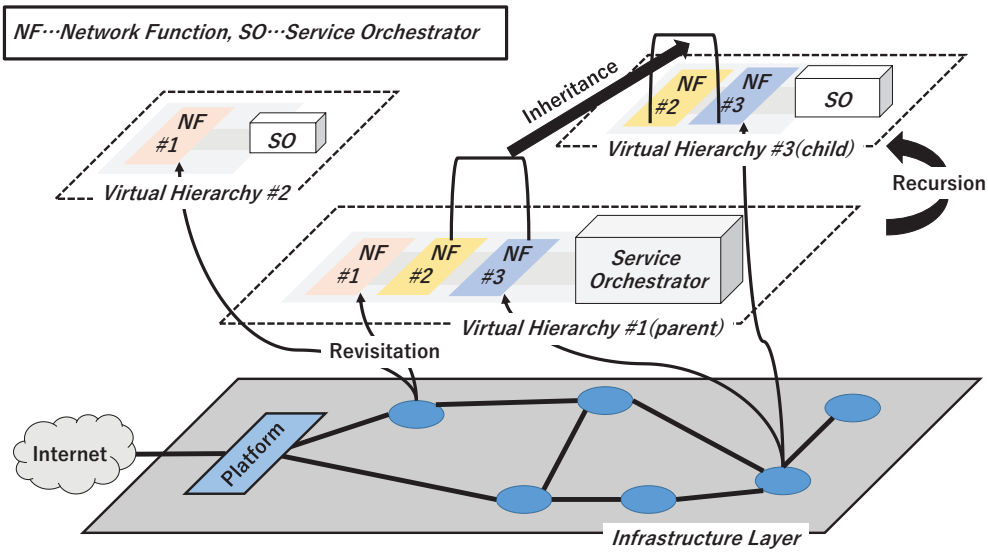

Fig. 1 System Abstraction

(iii) Inheritance
Following to the recursion from “Service Orchestrator” and the parent NF, “Service Orchestrator” and the child NF can inherit the architectural attributes from the parent. This work also means that the functional information of the parent can be automatically optimized and translated into the child’s information.

(iv) Revisitation
Virtual Networking Environment (VNE) of Fig. 1 provides the hosting multiple virtual nodes on “Virtual Hierarchy” to the physical node on “Infrastructure Layer” [11]. Such a reuse of resources makes easy to change the configuration of NF and “Service Orchestrator,” and to simplify the management of “Virtual Hierarchy.” Also, a mapping policy between a physical node and multiple virtual nodes has the possibility to add the cooperation among multiple vrtual nodes for the physical one.

Moreover, this system abstraction has a side of the SDN. In concrete terms, the rewriting/embedding process can be operated on “Virtual Hierarchy” as a function of data-plane. This process enables the end-to-end overlay networking on the existing network foundation. Thus, the owner of “Virtual Hierarchy” has a freedom to implement end-to-end services by deploying the embedded packet and its management as well as the system abstraction.

2 System Overview

In referring to the related study, we consider the control system to build the network driver with the virtual private connection between the clients and each level of the network application related to the properties mentioned above. In this paper, we propose the integrated platform which manages the application layer services divided by the port number. It works on the distributed virtual service orchestrator on Virtual Control Space (VCS) and the end-to-end overlay communication. Especially, the most important component is “vNIC.” We developed it originally as not a standard Linux extension but a kernel module. “vNIC” works as the virtual host of multiple server systems and provides the multiple hosts for the multiple clients on single-OS. Thus, the proposing system realizes the management system for multiple services on the overlay connection from an end node to another end node across an Internet. Fig. 2 shows the overall architecture of the original system in this paper.
“Software Foundation” provides the background for the components among the virtualization for “Infrastructure Layer” and ends the role locally on single-OS which implements “Virtual Hierarchies.” Therefore, “Software Foundation” is easy to customize its function without considering the external dependencies. There are various directions to provide various functionality to the system.

2.1 Details of Architecture

In this section, we will explain the details of the architecture in the actual environment.
Fig. 3 shows the overall architecture of the system in

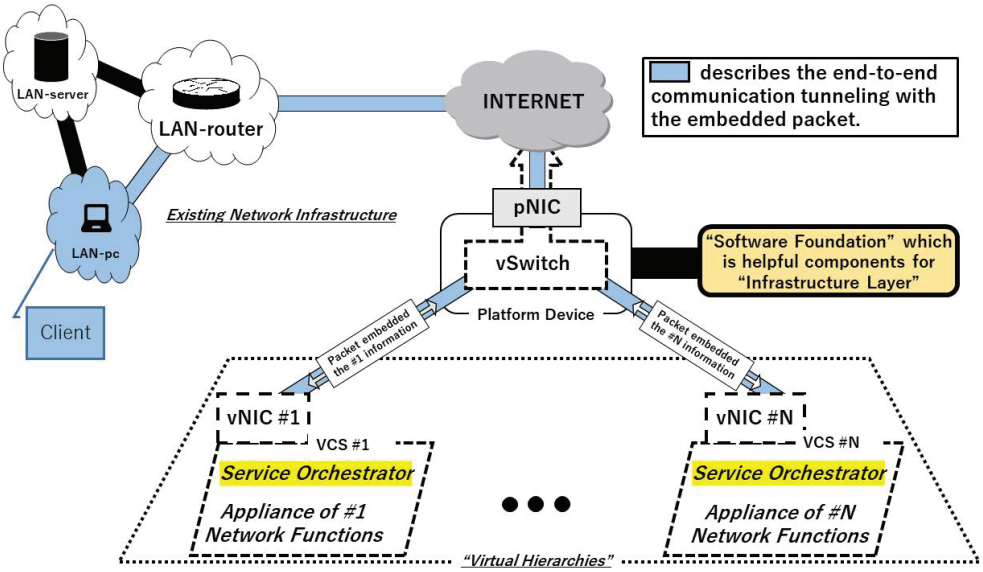

Fig. 2 Overall Architecture

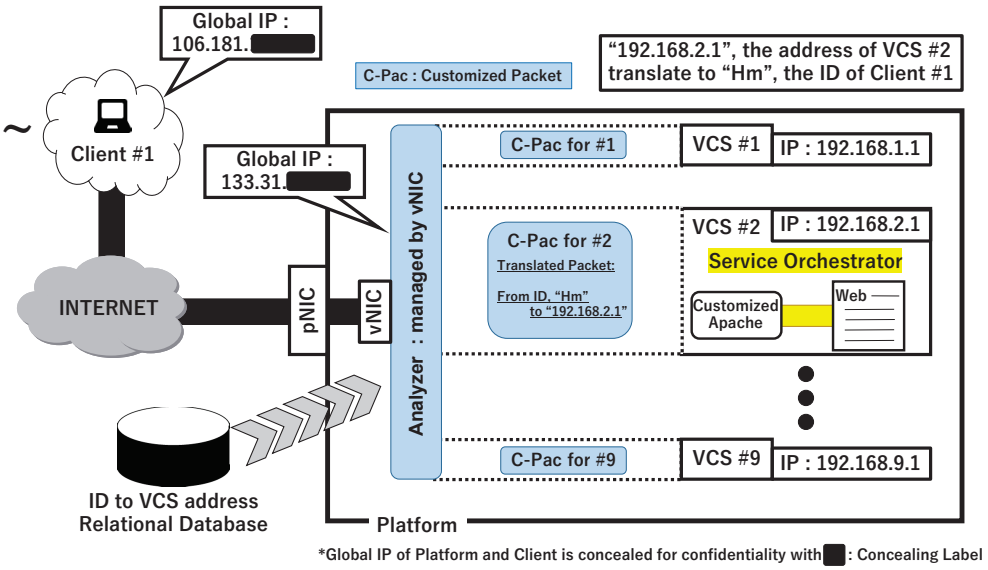

Fig. 3 Execution Environment

this paper. There are important properties as below.

(i) Client and Platform
In Fig. 3, the architecture is deployed on the overlay network both “Platform” and “Client” to capture and analyze the embedded packet. To be concrete, “Client” needs to embed the data on the packet for its certification. On the other hand, “Platform” needs to embed the data on the packet for the supplied service. Therefore, there is the overlay connection by the embedded packet between “Client” and “Platform.”

(ii) Virtual Control Space (VCS)
Each of “vNIC” manages each of “VCS” and deploys the service orchestrator. The service orchestrator is flexible and accessible to develop the NF’s association. That means each of “VCS” can manage the arbitrary service which is provided by the arbitrary port number mutually independently. Now, at “VCS_#2”, it works on HTTP protocol. Therefore, we prepare the data folder which is unique to “VCS_#2” (e.g., /var/www/html/VCS_#2) and the customized Apache configuration.

(iii) physical Network Interface Card (pNIC) and virtual Network Interface Card (vNIC)
“pNIC” is provided as the relay point among the communication for the client on “Infrastructure Layer.” This device processes all data flow related to the proposing system. “vNIC” is a software-defined middleware managing each of “VCS.” In Fig. 3, “vNIC” works as an “Analyzer” to behave itself as the entrance for “Client” and to filter the packet which is embedded the information for each of “VCS.” Also, other “vNIC” maintain each of “VCS” with a pseudo IP address which is used locally on “VCS.”

(iv) Embedded Packet
Embedding the unique information to a packet provides the overlay communication on the system. Between “pNIC” and “vNIC,” the unique information of “VCS” is embedded into the urgent pointer in the TCP/IP packet; the embedded packet will be analyzed and forwarded to an appropriate destination every time processing it. Besides, on the process of analyzing and forwarding a packet, the “Analyzer” refers the relational database to translate the ID to the pseudo-IP address of “VCS.” Its details are mentioned in section 3.

3 Experiments

3.1 Preparation for Execution

We built the Client-Server system through INTERNET to confirm the experiment on the operation of each of VCS numbers in the proposing system. We present that “How a client mounts to the overlay network,” “How a client recognizes the VCS number” and “What about the performance which our system accounts for.” To provide the presentation for these, we use the execution environment as shown in Fig. 3.

(i) Allocation of IP Address
Each of “VCS” has a pseudo IP address “192.168.x.x” (for security, the raw address is confidential with “x”) which is locally available in the “Platform” and the ID (e.g., “eA”) which is unique to itself. The pseudo IP address owes the range from the third octet to the fourth octet and maintains the identifier for the application operated on “VCS.” The ID is 2 bytes character information which maintains the identifier of “VCS” across Internet As to the correspondence between a pseudo IP address and an ID; the relational database manages it. On the other hand, for the practical address of the platform, vNIC has a global IP address “133.31.x.x” for end-to-end communication with the client. Similarly, the client has a global IP address “106.181.x.x.” Fig. 4 shows the loaded each of “VCS” which are managed each of vNIC.

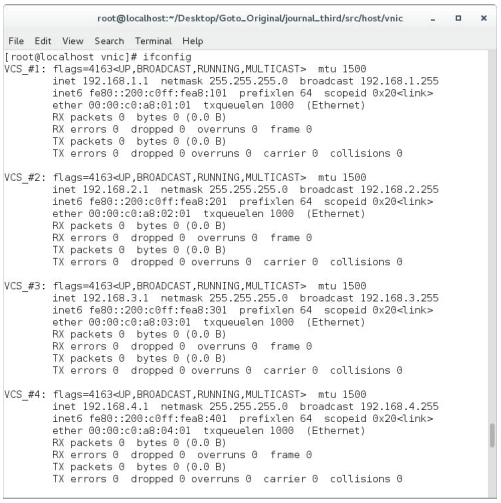

Fig. 4 VCS loaded as NIC

(ii) List of “VCS”
“Platform” maintains the database which describes the dynamic relationship between both information of each of “Client” and each of “VCS.” There is two list information, “List of

“VCS”” and “Data list for Routing.” “List of “VCS”” are made from the two files in Fig. 5. This list is used for the authentification of new “Client.” A linear list maintains the information including the information on the number (#) of VCS, the ID of “VCS,” the IP address of “VCS,” the MAC address of “VCS,” the descriptor of “VCS,” and the pointer to the next “VCS.” The section of “USER” in “permission.txt” describes the authorized name which is assigned to the client on the process flow. At the initial phase of the communication between the client and the platform, the client must know the authorized user name in advance and inquire the platform whether it is authorized user. If the inquiring procedure is normally accomplished, the client can use the embedding packet bridging to only specific “VCS.” The section of “VCS number” in “permission.txt” describes the available VCS number in each of the authorized users. Existing means the limitation of the provision of the applications managed by the platform which has not been described. “vcs_id .txt” describes all VCS numbers defined in the platform and the ID corresponded to each of the VCS numbers. In the text file, four strings at the right end are the ID and two strings from the left owe 2 bytes and is used in our current system. The full use of 4 strings is ready for the scalability of the components on “Infrastructure Layer.”

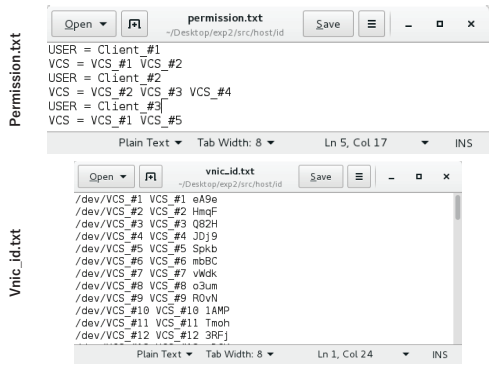

Fig. 5 permission.txt and vcs_id.txt

(iii) “Data list” for routing

On the platform, the relational database uses the “Data list for routing” for the routing between each of “VCS” and “Analyzer.” After the authentification, the information around the authorized user is held on “Data list,” which is including the IP address of “VCS,” the MAC address of “VCS,” the IP address for the routing which is unique to “VCS,” the MAC address for the routing which is unique to “VCS,” the descriptor of “VCS,” the IP address of “Client,” the port number, the number to count the packet, and “VCS” in use. Here, the routing needs a bit of contrivance. While the platform works on the only one IP address (global IP address) with the view from the outside, the platform has multiple IP addresses which are unique to each of “VCS.” Besides, another problem arises, that is the configuration of the default gateway. Each of “VCS” belongs a different IP segment and a single gateway that’s “Analyzer.” In contrast, vNIC has only one IP address. Hence, the only one of “VCS” can communicate without the contrivance. However, the proposing system assumes the condition that the multi-connection around “VCS” exists at the same time. For that reason, we took the original routing on the flow work in Fig. 6. The relational database compre-

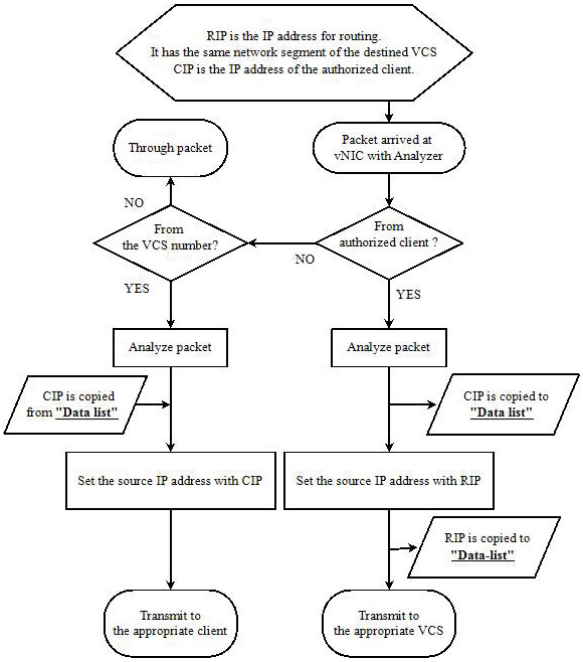

Fig. 6 Routing between VCS and vNIC with Analyzer

hensively manages the correspondence information among “RIP” and “CIP.” The reason to have a copy of “RIP” is for providing diversity to “Client.” In other words, there are relationships on one “RIP” to one “Client” on same “VCS”. If the multiple “Client” communicate with same “VCS,” “RIP” secures the range of the number of that “Client” and prepare the pseudo IP address for each of “Client.” To be concrete, each of “RIP” is the IP address whose fourth octet is added one by one in the order of arrival of “Client”. For example, in the case

of “Client_#2” which is the second to arrival at the “VCS_#2”, “RIP” is the IP address “192.168.2.3.” “CIP” is used when VCS sent back to “Client_#2.”

3.2 Authentification

Firstly, we verify the operation between “Client_#1” and “VCS_#2” on HTTP protocol (port: 80). In the experiment, “Client_#1” tries to communicate the web server by customized Apache built-in “VCS_#2.” At first phase, “Client_#1” tries to connect to “Platform” to prove the authority and receive the authorized ID. Fig. 7 shows its successful example.Firstly, we verify the operation between “Client_#1” and “VCS_#2” on HTTP protocol (port: 80). In the experiment, “Client_#1” tries to communicate the web server by customized Apache built-in “VCS_#2.” At first phase, “Client_#1” tries to connect to “Platform” to prove the authority and receive the authorized ID. Fig. 7 shows its successful example.

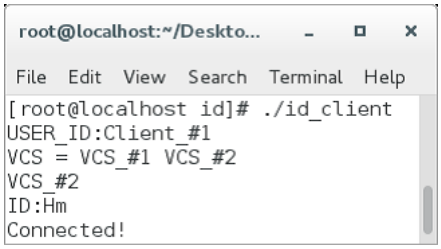

Fig. 7 Successful Example

After the authorized ID is brought to “Client_#1”, it is embedded into the packet and used for routing every time the packet process “Platform.” The detail of the embedding packet is as shown in Fig. 8.

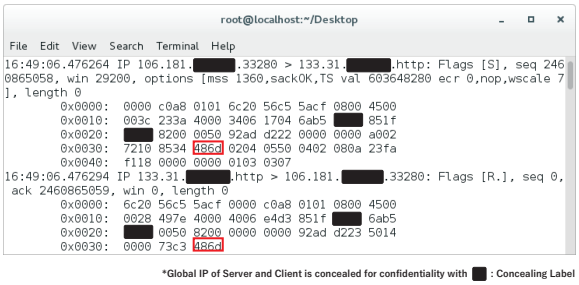

Fig. 8 Packet captured by TCPDUMP command

3.3 HTTP on VCS_#2

At second phase, “Client_#1” tries to send an HTTP request to the web server managed by VCS_#2. Fig. 9 and Fig. 10 show its successful example.

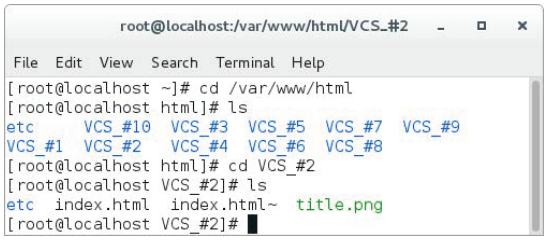

Fig. 9 Work Folder for VCS_#2

Fig. 10 HTTP application on VCS_#2

Fig. 9 shows the folder structure where “VCS_#2” works. Fig. 10 shows the HTTP service between “Client_#1” and “VCS_#2” A remarkable point on this result is that the path to “VCS_#2” field is concealed on the web page. It means only “133.31.x.x” is a host of Apache system and the multiplexing is operated on the overlay connection. In other words, if the programmers change the configuration of “VCS,” they can operate mutually independently each configuration of “VCS” in each folder of “VCS.” There is no change in the apache configuration.

3.4 Performance Test under the Execution Environment

To confirm a performance under the execution environment, we set the situation that “Client” tries to download the file from “Platform” in “VCS_#2.” We prepare the text file of 1.2 Gbytes for comparison of download speed. Besides, we use “Buffalo USB2.0 LAN Adapter” which has the 12.3 Mbytes/(s) bandwidth at most. Details of the device for “Platform” as shown in Table. 1.

Fig. 11 and Fig. 12 shows that the performance of our multi-server system under the limited bandwidth. Our system always indicates the max data transfer speed of 12.3 Mbytes/(s). CPU utilization is calculated through the filter for the process identification (PID) of “Plat-

	Computing Device as a platform
OS	Cent OS 7.2
CPU	Intel (R) Core i7 4790 3.60GHz
MEMORY	8192MB RAM
LAN System	10/100/1000BASE-T Gigabit Ethernet (LUA3-U2-ATX)

Table. 1 The specific information of the execution environments

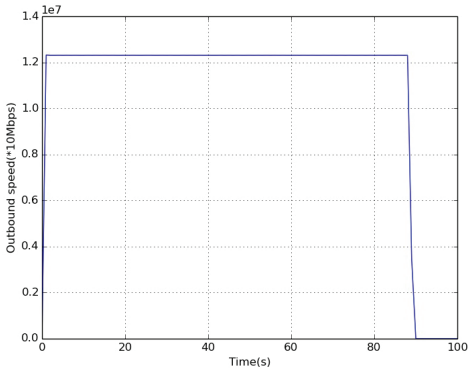

Fig. 11 Download Speed

form ” program. “ Number of processing(call) ” is the number of calls that our system is operated during downloading. Although there is some variation, our system does not exceed 12% utilization at most. Overall, our system always maintains the max speed of the bandwidth of NIC with the about 8% of CPU utilization.

4 Conclusion

We propose the multiple server systems on single-OS. This system provides VCS by vNIC which manages the application represented by the port number. VCS is guided by the following principles, coexistence, recursion, inheritance, and a revisitation. VCS maintains its own data folder for a corresponded application and freedom to implement the service which is provided by arbitrary port number. vNIC is not a standard Linux extension but an original module which provides the overlay system by the embedded packet[7]. Overall, this system aims the effectiveness of the virtualization including flexibility, manageability, scalability, programmability, and sharing. On the other hand, we aim the scope of the virtualization on single-OS. In contrast with OS-level virtualization like VM, we virtualize NIC originally on the kernel

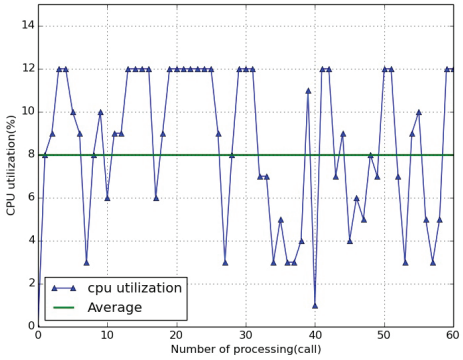

Fig. 12 CPU utilization

and propose VCS hosted vNIC. Finally, the experiment shows that the multiplexing server is successful without OS-level virtualization and each of VCS manages each of configuration independently.

5 Discussion

In this study, Our system has a component which we have dealt with the “Software Foundation.” The component has a lot of scalability in the design due to the flexibility of vNIC. For the further development of our system, there is room for formulating the scheme which adds more functionalities to the virtual infrastructure. Therefore, we can consider the cloud system which is built on the virtual infrastructure with “Software Foundation.” The architecture of the cloud system should be as shown in Fig. 13.

There is the extended system which is based on the current system with “ Software Foundation. ” This extended system has the features which are the functionality of the cloud system and the independence which indicates more free system about the configuration among the existing equipment of the network device. However, there are subjects about security because if there is also allowing a customized packet to dive into a private network through safety devices (e.g. firewall, IDS, IPS). Therefore, we must consider the scope of the arrangement of the packet carefully because the unstable scope causes inconsistency of the packet modification and the waste of resources. For that reason, we design the semantic foundation for the software definition on our system. With the point of view of feasibility, the important issue for the present is how to extend the embedded region. While the routing for the VCS numbers uses the 2 bytes information resources associated with a third octet and fourth octet in IP address, the routing for the platform numbers needs 4 bytes information resources. Therefore, the

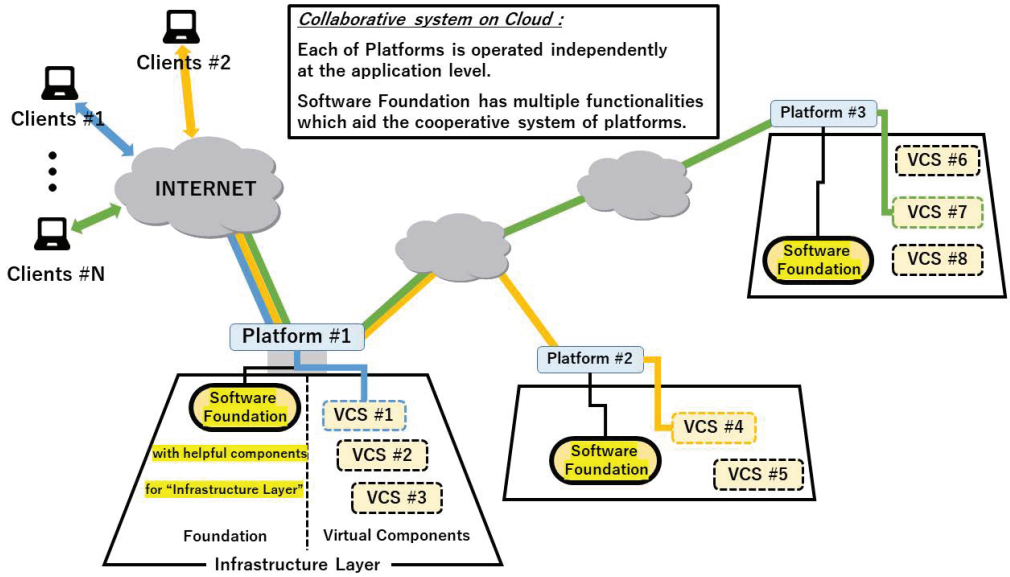

Fig. 13 Future Architecture

proposing system in future must adapt to extend the embedding region. The two methodologies are during the verification as below.

(i) The introduction of the relational database associated currently embedded information resources with the Global IP address of the platform numbers.

(ii) The addition of new 2 bytes information resources by the wrapping of the data part in a packet to currently embedded information resources.

The (i) methodology must be devised to be able to cover the full range of Global IP addresses. For the (ii) methodology, there is a dilemma between the simplicity to analyze the embedded packet and the complexity to analyze the part of wrapping on the packet. By solving these issues, the current proposing system makes it easy to extend itself to the cloud system exploiting the combination of the platform numbers.

Acknowledgements: The authors would like to express their hearty thanks to the referee who pointed out several typographical errors and made a very suggestive proposal to revise the manuscript of this paper. Especially, we would like to express the deepest appreciation to Yusuke Goto.

References:

[1] J. Altmann, K. Vanmechelen, O.F. Rana, *Estimating the Value Obtained from Using a Software Service Platform,* International Conference on Grid Economics and Business Models (GECON), 2013, pp. 244-255.

[2] N.M.Mosharaf Kabir Chowdhury, Raouf Boutaba, *Network Virtualization: State of The Art and Research Challenges,* IEEE Communication Magazines, Vol. 47, No. 7, 2009, pp. 20-26.

[3] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, Seungjoon Lee, *Network function virtualization: Challenges and opportunities for innovations,* IEEE Communications Magazine, Vol. 53, No. 2, 2015, pp. 90-97.

[4] Bhaskar R, Dr. Shylaja B. S, *Knowledge based reduction technique for virtual machine provisioning in cloud computing,* International Journal of Computer Science and Information Security(IJCSIS), Vol. 14, No. 7, 2016, pp. 472-475.

[5] Ross Dargahi, Kevin M. Henrikson, Roland J. Schemers, Jong Yoon Leem, *System and method for seamlessly integlate separate information systems within an application,* US Patent 8,631,065 B2, 2014.

[6] Mohamed Esam Elsaid, Christoph Meinel, *Multiple virtual machines live migration performance modelling – VMware vMotion Based Study*, IEEE International Conference on Cloud Engineering (IC2E), 2016.

[7] Raj Jain, Subharthi Paul, *Network virtualization and software defined networking for cloud computing: A Survey,* IEEE Communications Magazine, Vol. 51, No. 11, 2013, pp. 24-31.

[8] Stuart Clayman, Lefteris Mamatas, Alex Galis, *Efficient management solutions for software-defined infrastructures,* Network Operations and Management Symposium(NOMS) IEEE/IFIP, 2016.

[9] Albert Greenberg, Gisil Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhang, *A Clean State 4D Approach to Network Control and Management,* ACM SIGCOMM Computer Communication Review, Vol. 35, No. 5, 2005, pp. 41-54.

[10] Hyojoon Kim, Nick Feamster, *Improving network management with software defined network,* IEEE Communications Magazine, Vol. 51, No. 2, 2013, pp. 114-119.

[11] J. Touch, *Dynamic internet overlay deployment and management using the X-Bone,* International Conference on Network Protocols, 2000.

[12] Lefteris Mamatas, Stuart Clayman, Alex Galis, *A flexible information service for management of virtualized software-defined infrastructures,* International Journal of Network Management, Vol.26, No.5, 2016, pp. 396-418.

[13] Andreas Blenk, Arsany Basta, Martin Reisslein, Wolfgang Kellerer, *Survey on Network Virtualization Hypervisors for Software Defined Networking,* IEEE Communications Surveys & Tutorials, Vol. 18, No. 1, 2016, pp. 655-685.

[14] Shibo Luo, Kaoru Ota, Mianxiong Dong, Jun Wu, Jianhua Li, Bei Pei, *Toward high available SDNNFV-based Virtual Network Service in multi-providers scenario,* World Automation Congress (WAC), 2016.

[15] Lei Ren, Jin Cui, Yongchang Wei, Yuanjun LaiLi, Lin Zhang, *Research on the impact of service provider cooperative relationship on cloud manufacturing platform,* The International Journal of Advanced Manufacturing Technology, Vol. 86, No. 5, 2016, pp. 2279-2290.

[16] Robert Bradford, Evangelos Kotsovinos, Anja Feldmann, Harald Schiöberg, *Live wide-area migration of virtual machines including local persistent state,* Proceedings of the 3rd international conference on Virtual execution environments, 2007, pp. 169-179.