International Journal of Computer Networks and Communications Security

VOL. 5, NO. 7, JULY 2017, 153–164
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Proposal of a Foundation to Provide a TCP Service with Cooperative Applications

Satoshi Kodama[1], Rei Nakagawa[2] and Toshimitsu Tanouchi[3]

[1, 2, 3] Department of Information Sciences, Faculty of Science and Technology, Tokyo University of Science, Japan

[1]kodama@is.noda.tus.ac.jp, [2]j6316627@gmail.com, [3]j6316625@gmail.com

ABSTRACT

Recent developments in the computer industry owe a lot to innovations in networking infrastructure. Numerous protocols currently exist to exploit the Internet's potential applicability. However, this situation causes unnecessary complexity in the configuration around units that compose networking systems in case of implementation of extended design. Network virtualization provides a solution to such complexities and forms a fundamental part of future network design. In particular, an overlay network in a physical infrastructure provides the freedom to maintain a customized network policy between distinct domains distributed across the Internet. This paper proposes an overlay network foundation comprising two systems. The first is multiple virtual servers that provide authenticated TCP services between an authorized client and a service provider. The second is a cloud-based system that provides cooperative applications on UDP between the supervisor on the client side and the service providers. The overlay network is operated by the virtual network interface card (vNIC) identifier, which results in simple embedding with no change in packet size. The system's architecture is based on an original design that constitutes a vNIC, a physical network interface card (pNIC), and a software. The vNIC creates the virtual host, while the relay system manages the software. Thus, pNIC provides an interface between the physical layer and the software layer. Finally, we conduct empirical testing of the proposed system's viability using real-life Internet-based research. The results show that our proposed system performs effectively and has good operability.

Keywords: *Cross-Layer, System Design, Software Defined Network, Overlay Network, Cloud System.*

1 INTRODUCTION

1.1 Background

Network infrastructure has become a melting pot of many diverse network services and protocols, as well as the pledges between these. This has caused redundancy in parts of the communication process and led to excessive complexity in the physical and logical design configuration especially in the implementation of extended design. For example, virtual local area network (VLAN) is a well-known technology for network overlay schemes on the data link layer of the OSI reference model. VLAN addresses scalability, flexibility, security, and network management issues that are associated with a traditional LAN [12]. However, systems on data link layers such as the Internet have been scaled up dramatically in recent years and its scope is enormous. This creates a very complicated structure that is comparable to networks on the network layer of the OSI reference model. This situation causes three main critical issues: (i) a shortage in the number of VLAN IDs, (ii) a lack of Spanning Tree Protocol (STP) functionality, and (iii) a lack of scalability of Top of Rack (ToR) switches. However, these issues have encouraged the development of innovative technologies such as Virtual extensible local area network (VXLAN), network virtualization using generic routing encapsulation (NVGRE), and stateless transport tunneling (STT) [16]. These solutions support the end-to-end overlay network and provide flexible management of communication beyond the restrictions of the existing network model. VXLAN aims to resolve the issues of scalability and enables encapsulation of the MAC-based data link layer frames into the UDP datagram of the transport layer. It covers 16M overlay networks or segments, which provides sufficient capacity to resolve the abovementioned VLAN issues. GRE encapsulates various network-layer protocols to relay them to

other network-layer protocols. A GRE packet constitutes a delivery header, which relates to any network layer routing protocol; a GRE header; and a payload, which is any network layer protocol packet. It deploys an extended network of a data link layer on a network layer and provides a solution to the above issues as well as VXLAN. STT builds overlay networks on top of virtual networks. STT has an STT header, which is the encapsulated IP header, and it is extremely scalable for provisioning of virtual networks and creating isolation between a physical network and a virtual network. Customized policies and packet extensions have been developed for devices specialized for use of these technologies (e.g., Cisco Nexus 5624Q.). The most innovative aspect of these technologies is the concept of expansion by encapsulating the unique information of a packet beyond the restrictions of Internet protocol (IP) networks. Expansion can resolve not only the issue presented by VLAN but also the complexity of multihoming, traffic engineering, and live migration of a virtual network's configuration. Thus, VXLAN, NVGRE, and STT have been attracting much attention as innovative technologies, and therefore, we consider the overlay network operated by the embedded packet to be essential for deploying the original management policy on the network virtualization. Our aim is that the new multiple-server system will provide authenticated services on the TCP with cooperation among the servers. This system takes the design of the client–server system from the IP overlay network. There are TCP and UDP communications between the authorized users and the multiple servers used as service providers. Two backgrounds are considered for this system: simplification of encapsulation and expansion using a software-defined network (SDN). In most cases, the encapsulation sets the optional header and the data part of the packet. This method allows the designer to construct a highly scalable design. However, there is a fatal issue relating to the overheads of the encapsulated translation. With regard to this, various solutions have been studied [10][11][13]. To simplify the encapsulation and mitigate the increase in packet capacity in our system, we propose a simplified ID that is embedded to a URG pointer of a TCP header and the implementation of "Authentication" in the UDP. The ID is a character string of 2 bytes. It becomes active after establishment of the overlay network. "Authentication" is defined as a cooperative application and is the first operation for establishment of service delivery. Authentication also avoids the risk of always having to encapsulate all frames of a packet. In other words, our system reduces the communication load by dividing the communication line to the cooperative application of UDP and TCP services. These functions should be based on a flexible software foundation. We propose a packet tuner in bridging a network interface card (NIC) to the foundation. The design of the packet tuner is based on the effectiveness of SDN. Some researchers are combining the effectiveness of a SDN with overlay-network technologies [7][14][15]. These researchers aimed for simplicity and availability in tunneling between the nodes in charge of communication. We consider that the data plane of a SDN has a high affinity with encapsulation because the data plane's essence is data processing. We take the data plane concept into a design that bridges the physical NIC (pNIC) with the virtual NIC. There is a functionality to combine the embedding to anywhere on the Ethernet frame and freely decide routing in a way to configure use of a port number for arbitrary TCP services. The details of the design are described in Section 2. The following section describes the concepts we refer to in designing our system.

1.2 Related Technical Works

In the process of designing our system, we used the software-defined design represented by SDN components [1][2][3][4][5][6][8][9][17]. A SDN mitigates network complexities regarding too many protocols installed on a network's infrastructure as it separates the data plane from the control plane. The data plane refers to the data processing system (e.g., IP, TCP, and Ethernet), while the control plane is the system that determines how and to where packets are forwarded (e.g., routing, traffic engineering, and firewall). The separation of the data plane and the control plane is an effective method that makes the control of a network flexible and scalable not only within a network device but also over an entire network. We aim to utilize an SDN's effectiveness and incorporate both the data plane and the control plane into our software-defined design. In particular, Jain and Paul have mentioned that open application delivery networks (OpenADN) [3] will represent the future design of cloud computing. They indicated that most applications could easily obtain computing and storage facilities using cloud services from multiple providers distributed across the Internet. They mentioned a design for virtualization around a NIC. They assumed the existence of virtual service providers (vSPs) in the cloud and indicated the lack of scalability affecting physical NICs. In other words, they insisted that each vSP needs its own NIC. Therefore, they proposed three virtualization

designs based around a NIC. In relation to this, our design aims to provide virtual NICs (vNICs) as a software fundamental via the supervisor, which is proposed by virtual machine (VM) software vendors and uses the virtual Ethernet bridge (VEB) similar to tunneling in encapsulation. The most important part of this design is the structure's simplicity, its functionality, and its dependencies. The functionality of a vNIC represents a bridge between a pNIC and virtual components such as a virtual switch. There are no dependencies between the pNIC and the vNIC. We aim at simplicity and manageability of the vNIC, and therefore decided to exploit vNIC as our system's infrastructure. On the other hand, Chowdhury and Boutaba mentioned the details of the network virtualization environment (NVE) [1]. They indicated the applicability of the overlay network, which provides infrastructure for innovative technologies including cloud services, where they discussed the underlying concepts of the overlay network. We extended their concepts as shown in Fig. 1.

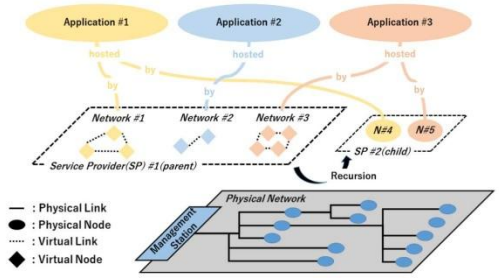

Fig. 1. Overlay network for a service provider

There are physical networks that provide the infrastructure for the application service provider and SPs that work on virtual networks offered by some physical networks. The physical node is managed centrally at the management station and plays the role of the resource comprising each application for each SP. Each resource on each physical node implements the policy with diversity at the application level, such as a working configuration on the Apache server. A virtual node constitutes an SP and its network provides the application service for an end-user. The virtual network from a virtual node makes it easy to design the presentation of application services and provides high extensibility in terms of mapping to the physical network. In addition, the following characteristics are present.

A. Coexistence

The coexistence of multiple virtual networks enables the definition of several application policies. All application policies referring a virtual network from a different SP can coexist. In other words, the multiple policies for applications are provided by one or more physical networks. In Fig. 1, the networks from #1 to #6 are coexisting virtual networks.

B. Recursion

Each virtual network from an SP can constitute different SPs. This makes the hierarchy between these SPs like a parent–child relationship in the form of 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. In Fig. 1, Network #1 and Network #3 on SP #1 make a nest with Network #4 and Network #5. Network #1 and Network #4 are associated with Application #1. Network #3 and Network #5 are associated with Application #3. This relationship maintains the availability of each application for interactive assistance.

C. Multihosting

The virtual networking environment (VNE) in Fig. 1 provides a hosting capacity to multiple virtual nodes to the physical node on the physical network. Such re-use of resources makes it easy to change the virtual network topology and helps simplify management of the virtual network. However, the mapping policy between a physical node and multiple virtual nodes should be carefully discussed against the nature of the physical network. To maintain logical correctness, both sides of the physical network and virtual network are open to their policies.

So far, the extended design can provide an overlay network design with high-level policies on the physical network design. The following section describes the issues that arose during actual implementation of the system.

2 SYSTEM DESIGN

2.1 Design Concepts

According to the design introduced in Section 1.2, we set the original design on the overlay network. Firstly, we outline the design concepts as follows.

A. Flexibility

Network virtualization must provide flexibility to make changes to the NVE. Each SP should be flexible enough to implement arbitrary network topology, routing, and forwarding functionalities, as well as customized control protocols

independent of the underlying physical network and other coexisting SPs.

B. Autonomy

The practical design of a SDN is often taken up the centralized management of virtual resources in a virtualized system. However, even the virtual network policy has been complicated recently because of the diversity of the network. Hence, we should consider the distributed system in terms of functional integration. All distributed systems have the same system infrastructure and operate each integrated function independently. This idea is derived from objective-oriented programming languages. The virtual network as a virtual object can illustrate a simplification of the dependencies on the system using hierarchically constructed functionalities.

C. Programmability

The programmable resources ensure flexibility and provide network virtualization necessarily. Only through programmability can each SP deploy its functional policy and customized protocol. However, network programming has too much flexibility, which can violate backward compatibility. Therefore, the question "How far should we set programmability?" is of considerable importance. The programmability of our system is related to the packet modification when processing and bridging between a NIC and the other cards on a device.

In our system, "flexibility" is associated with the virtual components (e.g., vNIC and Virtual Switch). While our system does not require specialized hardware, these components all are developed in C language and work as a software. In addition, our system takes the structure of an overlay network. In other words, all components of our system are required at the endpoint of the end-to-end communication. "Autonomy" is associated with the network function that each service provider provides in our system. Each provider runs the software groups, which have the same structure. Besides, the information structure that is embedded in the packet is consistent for unifying the reading format of the embedded information. "Programmability" is associated with the development environment on progressing the virtual components on our system. Almost all the Internet components (e.g., TCP, IP, and Ethernet) can be developed using C language. In other words, C can shape itself for everything that works as an expansion of the Internet. Next, we present the overall design of our system in the following subsections.

2.2 Details of System Design

Fig. 2 shows the provider-side design of the distributed system.

This comprises the virtualized components on the physical infrastructure and includes the following three primal components, which we should talk about as a premise. The apparatus depicted by a container (i.e. the SP) is the physical infrastructure that provides the workspace of a software group implementing our system's functions. "pNIC" constitutes the physical infrastructure that provides the interface for virtual components. "vNIC" is a primal component for the software groups. In other words, each vNIC is a base point for each of services provided in our system. Each property accords the following design concepts.

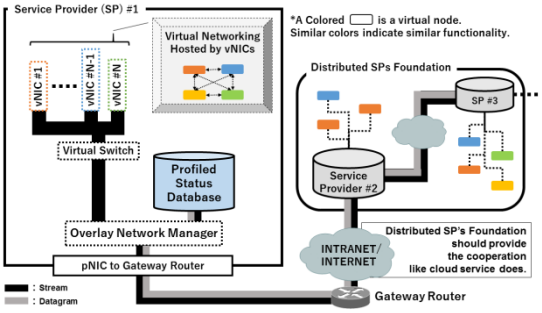

Fig. 2. Provider side: design

A. pNIC to Gateway Router

As mentioned earlier, pNIC means the physical NIC. pNIC is managed independently without the control of the network manager, which is a standard function on RHEL. It also works as the interface that receives electronic signal. The minimum requirement is one pNIC each for the client and the SPs.

B. Overlay Network Manager

The overlay network manager is a supervisor on the server side that has a global IP address for the SP. It handles two systems: the driver of an identifier and the cooperative application on the overlay network. First, the new identifier is formed by a character string of 2 bytes. It is embedded in the URG pointer in the TCP packet without enabling the URG flag in the TCP packet. This ID presents a vNIC that is hosting a TCP service on the overlay network. Second, the cooperative application is designed to supplement each TCP service that our system is executing on the overlay

network. It is based on the database (profiled status database) profiling each status of devices in our system corrected by interactive communication between them and operated on a UDP because there are multiple profile objects in any cooperative application. The notation of each status obeys a unique structure shared on our system. A software group implements the ID and the cooperative application. The details of this follow Section 2.2. The overlay network manager manages the information from the cooperative application at the profiled status database. By exploiting the information at the profiled status database for every service, the overlay network manager leads the client to the appropriate vNIC and the services that the client has requested.

C. Profiled Status Database

The profiled status database of the SP maintains client information and the vNIC receiving this information. The information is updated with data communication in UDP. For example, the profiled status database maintains the following columns: "ID LIST" (e.g., "eA.") of the available vNIC; "PASSCODE" of the authorized client; "vNIC NAME" (e.g., "vNIC_#1.") of the name of each vNIC; "DATA FOLDER" (e.g., "/dev/vNIC_#1."), where each vNIC manages each service; "SERVICE LIST" (e.g., "HTTP/SSH/..."), which can be provided by each vNIC; and "CLIENT LIST" which is the client information managed by the SP.

D. Virtual Switch

The function of the virtual switch is simply bridging the client to the appropriate vNIC number. The virtual switch is operated by a particular vNIC called the "relay-vNIC." The relay-vNIC is not the host for the TCP service as well as other vNICs. This is the packet receiver for all communication of vNIC numbers.

E. vNIC #1…vNIC #N and Virtual Networking

"vNIC #1 . vNIC #N" (vNIC numbers) are the virtual components processing the following features. Each vNIC number hosts TCP services represented by a known port number. The vNIC number groups that have the same number work on the same service for the cooperative application. Each vNIC number has a pseudo IP and MAC address for a TCP service. vNIC numbers are registered as our original RHEL kernel module, which is designed for the optimized virtual environment as VMs. Virtual networking associates the functionality of the cooperative application by routing vNIC numbers Its range spans from a local network closed in an SP to the IP network shared between distributed SPs.

F. Distributed Server Foundation

The distributed server foundation comprises the distributed SPs. Each SP maintains each host group of vNIC numbers. vNIC numbers represent virtual networking across distributed server foundations. Their distribution is for cooperative applications such as cloud services. For example, if the vNIC number has a similar functionality on each SP (e.g., blue in Fig. 2.), we can set the cooperative functions (e.g., fault tolerance and load balancing) by profiling the status of each vNIC number in blue.

Fig. 3 shows the design of the client side. A client is connected to the gateway router through the supervisor server (SS).

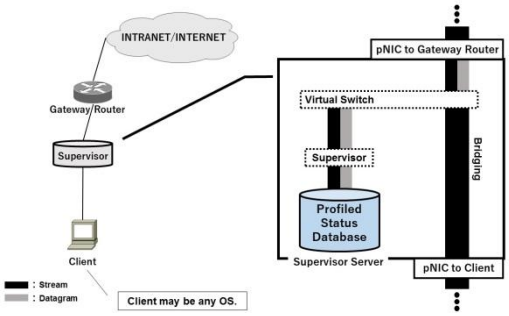

Fig. 3. Client side: design

While there is physical infrastructure that includes the components related to the pNIC as well as the provider side, the design does not require our system apparatus for the client, who is an end node. This SS apparatus works as the vice-router right under a gateway router on the client's local area network (LAN).

A. Bridging

The communication between "pNIC to Client" and "pNIC to Gateway Router" is not the only communication in the overlay network of our system. Therefore, the bridging process there must always be open. Any communication without an ID is always passed through by the "Supervisor."

B. Virtual Switch

The function of the virtual switch is similar to that of the provider side. However, the virtual switch does not have an IP address. It is the branch point on the bridged communication between "pNIC to Client" and "pNIC to Gateway Router." It works as a bridge of common communication

between the driver (e.g., embedding, extracting, and analysis) of ID on the communication of the overlay network and the filter of the communication with an ID and one for the cooperative application to the "Supervisor."

C. Supervisor

The "Supervisor" is operated by a vNIC. It profiles the status of the distributed providers using the profiled status database. A unique data structure (see Section 2.3.3) determines the profiling data scheme. The supervisor has an IP address to receive and customize the packets that contain the status information of each provider. This IP address is used only for communication with the SPs. Hence, the IP address is concealed from the client. On the other hand, the supervisor works as a substitute of a client for the SP. It means that a client may not meet the particular requirements. In summary, the "Supervisor" plays the role of the junction of communication from the client, which operates the cooperative application in the overlay network.

D. Profiled Status Database

The profiled status database in SS maintains client information and the vNICs receiving this information in addition to the SP. The client's information is updated on initialization of the system. The vNIC information is regularly updated via interactive communication in the UDP. So far, we have introduced the overall design of system, which works as the foundation of our system. The design is for cooperative applications that can be arbitrarily set in the client and the provider sides, such as load-balancing and redundancy. Cooperative applications should be defined as options for the primal function in our system, which is the service delivery on the TCP to the client which has been negotiated in advance. A software group operates with these functions. In the following subsection, we introduce the details of this software group.

2.3 Details of Software Group

In our system, there are two workflows by the software. The first workflow is "Open of the communication channel" and "Control of the cooperative application." A second workflow is "Handling the embedded packet with ID on the overlay network." The first workflow is operated by a interactive communication between an SS and multiple SPs. As well, the second one is operated between an SS and an SP. A software group operates the interactive communication, which is installed on both side of the client and the provider. The following figure shows the overall design of the software group. Fig. 4 shows the components of the software group. "C" language develops these components because of the implements of the kernel modules and the network communication reprogramming. There are following properties.

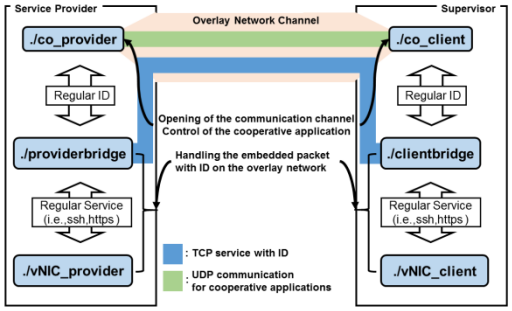

Fig. 4. Software group on both sides, the client and the provider

A. "./co_provider" and "./co_client"

"./co_provider" and "./co_client" are the execution files in RHEL. The interactive communication among these operates "Opening of the communication channel" and "Control of the cooperative application." A software routine operates "Opening of the communication channel," which is called "Authentication." Authentication is one of the cooperative applications and the first process determining the communication propriety in our system. This is detailed in the following subsection 2.3.1. A client must send the passcode of a requested service to "Authentication." The starter of "Authentication" is "./coclient," and "./coclient" send a series of information to "./co_provider" at "Authentication." Further, "./co_provider" validates the correctness of the series of information. If "./co_provider" opens the communication channel with "./co client," an ID is registered with the profiled status database of both sides. This indicates the start of the TCP service related to ID and the shift of the processing from "./providerbridge" to "./clientbridge." On the other hand, "Control of the cooperative application" is managed using the unique data structure mentioned in Section 2.2. The cooperative application should be designed based on sequential status profiling, which is recorded using the data structure on UDP as well as "Authentication."

B. "./providerbridge" and "./clientbridge"

"./providerbridge" and "./clientbridge" are the execution files in RHEL. Management of TCP communication on the overlay network is an essential process represented by the driver of the ID embedding. After "Authentication" is accomplished

successfully, "./providerbridge" and "./clientbridge" refer an ID to "./co_provider" and "./coclient." The shared ID is first embedded to the packet and sent to the SP by "./co_client." "./co_provider" then receives the embedded packet and extracts the shared ID to provide the appropriate TCP service. In addition, the secondary functionality of "./providerbridge" and "./clientbridge" is a fixer of the existing network protocols for both SPs and SS. The existing network protocols include the ARP resolvation, the suspension of IP forwarding, and so on.

C. "./vNIC_provider" and "./vNIC_client"

"./vNIC_provider" is the loader of both "Virtual Switch" and "vNIC #1...N" in SP. "Virtual Switch" is configured as a relay-vNIC. "vNIC #1...N" is registered as the apparatus in the kernel. Each "./vNIC_#1.. .N" maintains the configuration file for the hosting service. For example, the multiple servers in an SP require configuration of the virtual host by Apache. This configuration enables the multiple data folder connected with each IP address of vNIC for HTTP and HTTPS access. "./vNIC_client" is the loader of both "Virtual Switch" and "Supervisor". In SS, "Virtual Switch" and "Supervisor" are managed by "./clientbridge" as well as the provider side. "Supervisor" has a local IP address in the client's LAN and processes only communication for cooperative applications including authentication.

These properties are the parallel processes achieved via multi-thread programming. Although each of the functionalities is different, they share a primary objective: the sorting of a packet and its processing similar to the data plane of an SDN. The following subsection describes the packet processing details.

2.3.1 Authentication

So far, authentication has been described as the essential function to start a TCP service in our system. "Authentication" is operated using the communication channel from "Supervisor" in SS to "Overlay Network Manager" in the SP via UDP/IP communication. "Supervisor" communicates with SP to receive ID through several iterations of communication as TCP three-way handshake communication. The process flow is shown in Fig. 5. The communication of "Authentication" comprises four kinds of packets, "Auth-REQ," "Auth-REP & REQ," "Auth-REP," and "Auth-RST."

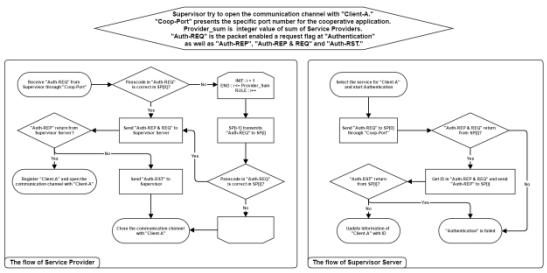

Fig. 5. Authentication work flow

These packets relate to the data structure in Section 2.3.3. "Auth-REQ" is the first request packet in "Authentication," which is the client's passcode and the requested packet and is sent from an SS to a representative SP. The representative SP refers to the header SP that maintains the address lists of other SPs. "Auth-REP & REQ" is the second reply packet against "Auth-REQ," which is the ID of an appropriate vNIC. It is sent to the SS from the SP to which the passcode belongs. "Auth-REP" is the final packet against "Auth-REP & REQ," which indicates "Authentication" success. It is sent to the SP from the SS. The success of "Authentication" implies acquisition of the regular ID by "Supervisor." "Auth-RST" is the packet for an emergency, which informs that a problem has occurred with "Authentication." It is sent to the SS from the SP and then resets "Authentication." A failure in "Authentication" means that the communication channel was not opened and it is necessary to perform "Authentication" again. In addition, there is a time limit for all reception systems. If a packet does not arrive before the time limit, the reception system considers this situation to be a failure in "Authentication."

2.3.2 Packet Processing after "Authentication"

After a few rounds of communication, "Authentication" is completed. After the success of "Authentication," "Supervisor" receives the ID of the requested vNIC and is ready and waiting to receive the TCP service from the SP. Packet processing then follows the order of the network models from the Ethernet layer. Fig. 6 shows the packet-processing workflow. Both the SP and the SS feature basic packet processing on the Ethernet, IP, TCP, and UDP. The fixer of the existing protocols does the same work on both sides of the SP and the SS. Its main job is address resolution on the LAN. At the layer higher than the IP, packet processing takes a different step depending on the packet protocol on the transport layer, UDP or TCP. At TCP, the SP determines the arrival of a packet in a TCP service with "Client-A" and extracts the embedded ID from the packet. The SP

handles the service to be offered to "Client-A" if there is no contradiction between the ID and the "Client-A" referring to the profiled status database. In another case, the SP then rejects the packet. In addition, the SS determines the arrival of the packet from the SP and checks the embedded ID. At the UDP, the SP is a passive system that is waiting for access from the SS. The SP receives the series of the packet as well as "Authentication" and determines the requested cooperative application by checking the data structure in the packet. The SS is an active system that begins communication for the cooperative application. "Supervisor" enables an appropriate cooperative application depending on the demand from "Client-A" and then sends a series of packets with the data structure customized for the cooperative application. After the processing of the series of packet, both sides are ready to offer the cooperative application and operate it.

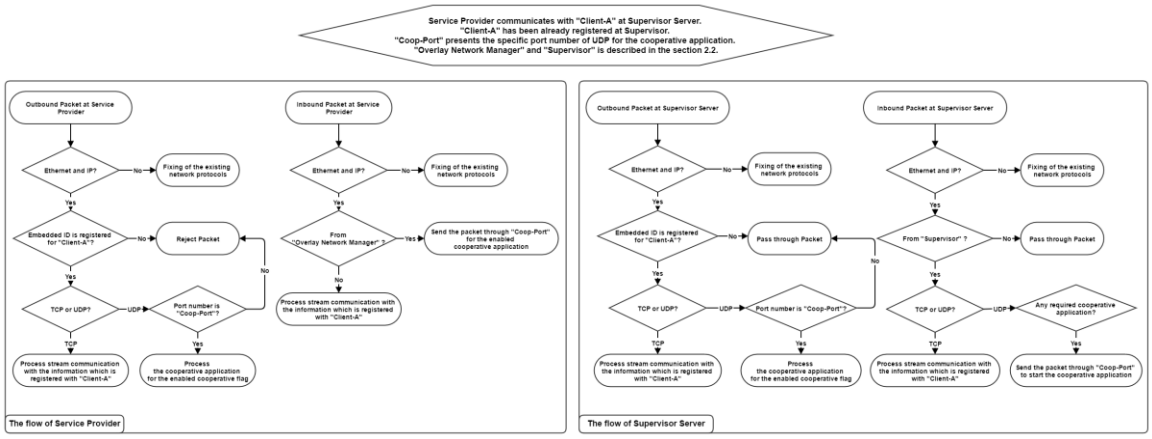

Fig. 6. Packet-processing work flow

2.3.3 Data Structure for Cooperative Application

The cooperative application uses the unique data structure on the UDP. This data maintains the following fields: a cooperative flag, a control flag, and the status of a vNIC. The unused domain is initialized as zero. A developer can define the scheme of the non-reserved part freely. Fig. 7 shows the design of a cooperative application.

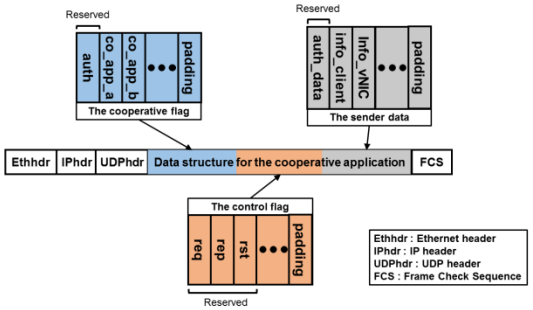

Fig. 7. Design: data structure for the cooperative application

A field of the cooperative flag corresponds to the name of the cooperative application. For example, "Authentication" determines whether the "auth" flag on the cooperative flag is turned on or not. A field of the control flag presents the graded communication control, "request," "reply & request," "reply," and "reset." These should be used as the control functionality for the cooperative application. A field of the sender data represents the data part that a cooperative application needs to gather depending on the sender. The cooperative applications after "Authentication" should be based on data profiling among each SP and SS. If the sender is SS, the client information should be managed in SS as "info_client." If the sender is SP, the vNIC information should be hosting each TCP service as "info_vNIC." It is desirable to use this data selectively as a secure communication path because of the properties of this data. A reserved field of "auth_data" is shared on both sides of the SS and SP. It presents the exchange data at the time of "Authentication," such as the passcode of vNIC and the service requested by the client. Examples of a data structure use in "Authentication," are as follows: "Auth-REQ" is auth = 1, req = 1, authdata.passcode = "AAA," and authdata.service = "HTTP". "Auth-REP & REQ" is auth = 1, req = 1, rep = 1, and id = "aa." "Auth-REP" is auth = 1 and rep = 1. "Auth-RST" is auth = 1, rst = 1, and error = 1.

3 PERFORMACE TEST FOR THE OPERABILITY OF OUR SYSTEM

In this section, we conduct a simple experiment to confirm the operability of our system. Fig. 8 shows the environment for the experiment: SS and SP #1.

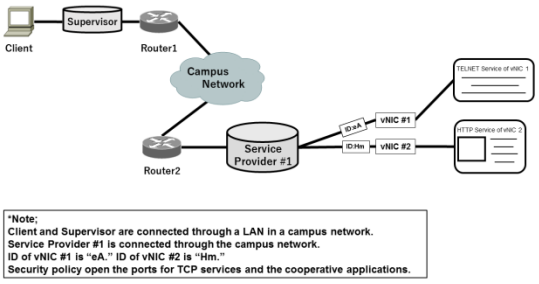

Fig. 8. The Environment for the Experiment

SP #1 manages the HTTP service to provide high-resolution animation delivery in addition to the TELNET service. vNIC #1 and vNIC #2 in SP #1 host these services. A service requested by a client is one HTTP service and is registered with the SS at the initialization stage of the system. The client connects the services through the series processes that we have described so far: "Authentication" of the cooperative application and the overlay communication management by the ID. The HTTP service provides high-resolution animation delivery (Full HD Resolution, 390 Mb, 7 min 23 sec). Table 1 shows the specifications of the apparatus used in the experiment. Both SS and SP #1 use this to prove the versatility of our system. SS also uses two NICs as pNICs, and SP #1 also uses a NIC. The objective of the experiment is measuring the CPU load of the simplified embedding on TCP by running the HTTP service that has been mentioned. The following subsection describes the operability of our system validated by conducting an experiment.

Table. 1: The specification of an apparatus in the experiment

	SS	**SP #1**
CPU	Intel (R) Core i7-4790 CPU 3.60GHz	
Memory	DDR3 SDRAM 8 GB	
OS	Cent OS Linux release 7.3.1611 (core)	
Equipment	LUA3-U2-ATX (Buffalo) & RTL 8111/8168/8411 (Realtek)	RTL 8111/8168/8411 (Realtek)

Table 2 shows the CPU load of "./hostbridge" of the SP and "./clientbridge" of the SS in five trials according to the experiment. We found that the average CPU load of SP #1 was around 0.35% and that of SS was 0.5%. These experimental results demonstrate the operability of our system because the biggest load on the system depends on the TCP service delivery by embedding to the URG pointer.

Table. 2: The CPU load in the embedding according to the experiment

	SS			**SP #1**		
Num of trials	**Min(%)**	**Avg(%)**	**Max(%)**	**Min(%)**	**Avg(%)**	**Max(%)**
1	0.2	0.47	0.6	0.2	0.35	0.6
2	0.2	0.46	0.6	0.1	0.37	0.7
3	0.4	0.52	0.6	0.2	0.34	0.6
4	0.2	0.45	0.6	0.2	0.36	0.7
5	0.2	0.46	0.6	0.1	0.32	0.6

4 CONCLUSION

We propose a one-client–server system. The client side constitutes a client and a supervisor apparatus called the "Supervisor Server (SS)." The server side constitutes the distributed "Service Provider (SP)." The architecture of the client side and the server side exploits the flexibility of customization possible because of using a virtual network interface card(vNIC). The system is designed as a foundation to provide two systems. The first is the multiple virtual servers that provide authenticated TCP services between an authorized client and an SP. The second is a cloud system that provides cooperative applications on UDP between the supervisor on the client side and the SPs. TCP services used a 2-byte simplified identifier to demonstrate the vNIC to which the client is connected. This ID is embedded to the URG pointer in every packet in TCP services without resizing

the packet, such as the encapsulation of VXLAN, NVGRE, and STT. Cooperative applications on the UDP provide the tool for the overlay network in the system to profile each status of vNIC numbers in distributed SPs. A cooperative application is managed by a unique data structure on the UDP. A developer can define the scheme in terms of the functionality of the cooperative application. Finally, we conducted a performance test of the system and measured performance in one HTTP service. Thus, we show performance on embedding the ID and operability as a virtual solution composed of general-purpose apparatuses.

5 DISCUSSION

So far, we have described the process that builds the distributed server system on the overlay network. We now discuss some interesting considerations.

5.1 vNIC

In the proposed system, vNIC is a virtualized NIC and provides the opportunity to access each header of a packet and directly write/read data there. This can be very useful and form a self-governing platform among pNICs. vNIC derives this from the innovation of SDNs: the independence of the data plane and the control plane. We consider this innovation to need the stack to deploy customization on the packet. In other words, we implement vNIC as the relay point that can place customization between write/send and read/recv functions in the C programming language. The innovative aspect of vNIC is just this relay point. Fig. 9 shows packet customization both with and without vNICs.

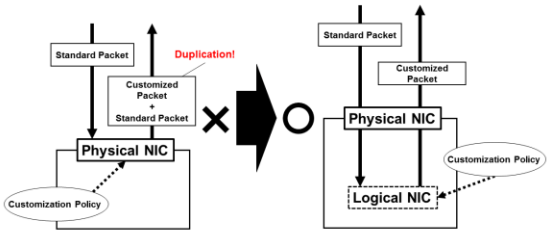

Fig. 9. vNIC utilization

We need to deploy customization on the original buffer of the pNIC without a vNIC. However, network programming has a limitation in that it maintains a copy of the original buffer of a physical NIC instead of the original. This limitation results in duplication of the customized packet and the standard packet. Therefore, to modify the initial buffer of the NIC, we implement a buffer of vNIC that bridges to the pNIC. The vNIC buffer allows a very flexible response to build the encapsulation of the packet, which is the foundation of the new protocol.

5.2 Embedding

The embedding process is the most delicate and significant issue in deploying the overlay network. In our system, we use the URG pointer of the TCP/IP packet and the data part of the UDP/IP packet. The URG pointer is used for service delivery. In other words, the URG pointer is used for all communications in our system. On the other hand, the data part of the UDP/IP packet is only used for particular communication that has been defined in our system. These features have the following merits. It does not resize the packet that is used for service delivery. The well-known technologies of the overlay network use encapsulation with original header information. The size of encapsulation is too huge to encapsulate the customized policy with an original frame: 50 bytes of a VXLAN header, 42 bytes of a NVGRE header, and 64k bytes of a STT header. Altering the packet size causes problems in the existing network protocol, such as IP fragmentation on IPv4. On the other hand, embedding the ID to the URG pointer of the TCP/IP packet uses only a 2-byte ID. This light processing is derived from "Authentication," which lets the system exchange the data causing an increase of the embedding size beforehand, at the start of communication. In other words, separation of "Authentication" and the service delivery prevents packet resizing in the service delivery accounting for most communications in our system. Therefore, embedding to the URG pointer is a solution to this problem. Besides, there is the risk that the service with the URG pointer may not normally work because the use of the URG pointer should be set by every application individually. However, the problem about the service availability should be easy to overcome because it is easy to customize the application on the SP owing to its inherent programmability.

5.3 IP Multicast Routing

Our system is designed to enable cooperative application for a client established between the SS and distributed SPs. The scope of the cooperative application depends on a characteristic of vNIC. This means that using a vNIC is possible on all devices that RHEL works on. In the characteristic of vNIC, the interactive works of vNIC numbers should be performed across the distributed SPs on the Internet. Now, there is a problem in deploying

the cooperative application. The problem is IP multicast routing among vNIC numbers in distributed SPs due to secrecy and availability of the services provided in our system. Firstly, there is a danger that all access (including an attack) concentrates on a header SP (Section 2.3.1). A solution to this problem is system scattering access in plural header SPs. Secondly, Internet group management protocol (IGMP) is not applicable as a solution because it has a poor affinity with firewall due to security issues. Therefore, there is a need to build an original IP multicast system. There are some discussions about this in the literature [14]. In our system, the requirements of the multicast are as follows. First, the multicast should have some authentication mechanisms. Multicast routing including IGMP has issues, such as an assailant could easily occupy the bandwidth of the network because the network has no authentication mechanism. Because our system has already established the authentication between the SS and a header SP, it is necessary to establish the authentication system between a header SP and distributed SPs. Second, the multicast should constitute distributed SPs in the SP group that one header SP manages. Each group by a header
SP of the SP groups prevents the excessive access to the other group. Thus, the multicast functionality should support availability of the service delivery by access dispersion method to the plural SPs. Finally, the implementation of the multicast system should result in a system that can utilize the cooperation by vNIC to a greater extent.

5.4 Introduction of a Security Policy

In our system, there are vulnerabilities related to the security policy around the data exchanged in the overlay network. For example, the ID of the packet for the TCP service is always accessible to the network without the overlay network because it is in plaintext during communication. Certainly, the existing protocols, such as IPsec, securing the confidentiality of IP datagram covers this. However, these security protocols are not suitable for ensuring security on all systems due to its performance costs, configuration complexity, and lack of robust scalability. Our system needs to establish the secured session for selected raw data during communication. At present, the characteristics of vNIC deserve attention for this task. As vNIC has coexistence characteristics, vNIC is applicable for selectively multiplexing the session for raw data. For example, we differentiate in the policy of a particular TCP service or the cooperative application and let multiple-vNIC-number support this. The vNIC group in the same number as this vNIC number then secures the TLS communication between each SP and SS. This selective security should benefit both highly confidential data transmission and system performance.

6 REFERENCES

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

[2] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee, “Network function virtualization: Challenges and opportunities for innovations”, IEEE Communications Magazine, Vol. 53, No. 11, 2015, pp.90-97.

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

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

[5] Albert Greenberg, Gisil Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, and 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.

[6] Hyojoon Kim and Nick Feamster, “Improving network management with software defined network”, IEEE Communication Magazine, Vol. 51, No. 2, 2013, pp.114-119.

[7] Joe Touch, “Dynamic internet overlay deployment and management using the X-Bone”, International Conference on Network Protocols, 2000.

[8] Lefteris Mamatas, Stuart Clayman, and 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.

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

[10] Heitor Moraes, Marcos A. M. Vieira, Italo Cunha, and Dorgival Guedes, "Efficient Virtual Network Isolation in Multi-Tenant Data Centers on Commodity Ethernet Switches", IFIP Networking Conference (IFIP Networking) and Workshops, 2016.
[11] Bai Yi, Bao Congxiao, and Li Xing, "FlowLAN: A non-tunneling distributed virtual network based on IPv6", Information Technology, Networking, Electronic and Automation Control Conference, IEEE, 2016.
[12] Isiaka A. Alimi and Akeem O. Mufutau, "Enhancement of Network Performance of an Enterprises Network with VLAN", American Journal of Mobile Systems, Applications and Services, Vol. 1, No. 21, 2015, pp.82-93.
[13] Aurelien Monot, Manuel Oriol, Camille Schneider, and Michael Wahler, "Modern Software Architecture for Embedded Real-Time Devices: High Value, Little Overhead", 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), 2016.
[14] Yukihiro Nakagawa, Kazuki Hyoudou, and Takeshi Shimizu, "A Management Method of IP Multicast in Overlay Networks using OpenFlow", Proceedings of the first workshop on Hot topics in software defined networks, 2012.
[15] Kyuho Jeong and Renato Figueiredo, "Self-configuring Software-defined Overlay Bypass for Seamless Inter- and Intra-cloud Virtual Networking", Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing, 2016.
[16] Muhammad Aamir Nadeem and Taimur Karamat, "A Survey of Cloud Network Overlay Protocols", 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), 2016.
[17] Dmitry Drutskoy, Eric Keller, and Jennifer Rexford, "Scalable Network Virtualization in Software-Defined Networks", IEEE Internet Computing, Vol. 17, No. 2, 2013, pp.20-27.