논문 17-42-08-19 The Journal of Korean Institute of Communications and Information Sciences '17-08 Vol.42 No.08
https://doi.org/10.7840/kics.2017.42.8.1636

NFV 환경에서의 데이터평면 가속화 기술들의 적용 및 성능비교 분석

박 영 기◆, 양 현 식*, 김 영 한°

Application and Comparison of Data Plane Acceleration Technologies for NFV

Young-ki Park◆, Hyun-sik Yang*, Young-han Kim°

요 약

본 논문에서 오픈스택을 NFVI(Network Functions Virtualization Infrastructure)를 위한 VIM(Virtualized Infrastructure Manager)으로 사용하는 환경에서 NFV기반의 데이터평면의 가속화를 위해 일반적인 네트워크 기술 및 SR-IOV, DPDK, FD.io와 같은 패킷가속화 기술을 적용하여 성능을 비교 분석 하였다. 실험구성은 두개의 컴퓨트 노드와 한 개의 컨트롤러 구조로 구성하고 네트워크 성능측정을 위해 Packet Generator(Send & Receive)를, 그리고 Compute1, Compute2에는 일반적인 VNF로 구성한 경우와 Telco서비스를 위한 vEPC와 같은 NF로 구성하여 VM2VM(VM to VM)간의 성능측정을 하였다. 실험결과 일반적인 네트워크 성능은 SR-IOV가 가장 높게 측정 되었으며, Raw Ethernet을 사용하는 vEPC와 같은 NF구조에서는 DPDK의 성능이 SR-IOV와 비슷하게 측정되었다. FD.io의 경우 Phy2Phy(Physical-vSwitch-Physical) 상태에서는 가장 높게 측정되었지만, VM2VM에서는 VM과 vSwitch의 Logical port의 병목으로 인해 성능에 저하가 있는 것을 확인 할 수 있었다.

Key Words : NFV, OpenStack, SR-IOV, DPDK, FD.io

ABSTRACT

In this paper, we compared and analyzed the performance of common network technology and packet-acceleration techniques such as SR-IOV, DPDK, FD.io which is used for accelerating data planes in an OpentSack based NFVI environment. we use two compute node and one controller and the packet generator (Send & Recive) is configured to measure the network performance. we deployed two kinds of VNF(non-vEPC,vEPC) and meassure the performance between the VM(VM to VM). Experimental results showed that SR-IOV is the highest network performance in common network environment and DPDK performance is similar to SR-IOV in NF structure such as vEPC using Raw Ethernet. FD.io was the highest performance in Phy2Phy (Physical-vSwitch-Physical) state, but the case of VM2VM showed that the performance was degraded due to logical port bottleneck between VM and vSwitch.

※ "본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2017-2017-0-01633)

◆ First Author : Soongsil University School of Electronic Engineering, red1028@ssu.ac.kr, 학생회원
° Corresponding Author : Soongsil University School of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
* Soongsil University School of Electronic Engineering, hyunchic86@ssu.ac.kr, 학생회원
논문번호 : KICS2017-05-153, Received May 23, 2017; Revised July 24, 2017; Accepted August 3, 2017

Ⅰ. 서 론

가상화 기술은 하나의 물리적인 시스템의 자원을 가상화 시켜 자원을 동적으로 활용이 가능하게 하여 전체적인 시스템의 유연성을 높이고 효율적인 운용이 가능하게 해주는 기술이다. 이는 클라우드 컴퓨팅의 주요 기술로 가상화 기술이 발전함에 따라 활용 가능한 분야 또한 증가하였다. 클라우드는 기존의 단일화된 서버 구축 형태를 벗어나 사용자 및 서비스별로 컴퓨팅 환경을 독립적으로 제공 가능하다. 클라우드 환경은 대표적으로 기업의 데이터 센터에서 많이 사용되고 있으며 그 외에도 IoT 플랫폼 서비스, 모바일 네트워크, 스마트 홈 서비스 등 다양한 환경에서 활용되고 있다[1-3]. 4세대 모바일 코어 네트워크 구조인 EPC(Evolved Packet Core)와 같은 네트워크 기능도 주로 전용 하드웨어 장치를 통해 제공되었지만 가상화 기술이 확산되면서 NFV(Network Functions Virtualization)기술 기반으로 개발된 vEPC (Virtualized EPC)도 등장하였다. 이러한 형태는 전체 구조의 유연성 확장 및 트래픽 증감에 따른 능동적인 인프라를 구축 할 수 있어서 기존의 하드웨어 기반의 구조에 비해 많은 이점을 가져올 것으로 예측된다. 또한 vEPC 환경의 SGi-LAN 내에서 SFC(Service function chaining)를 통해 기능을 관리하는 경우에도 모바일 요청 패킷에 따라 제공 서비스를 다르게 구성 할 수 있다[4,5]. 그러나 이와 같이 네트워크 계층을 통과하게 되면서 내부 패킷 처리로 인한 지연이 발생할 수 있다. 이러한 부분은 전체 환경의 성능 측면에서 문제가 될 수도 있기 때문에 고려되어야 한다. 이러한 문제를 극복하고자 어플리케이션이 실제 물리 네트워크 카드에 직접적으로 접근하여 데이터를 처리할 수 있게 해주는 SR-IOV(Single Root-IO Virtualization)와 Intel Architecture 기반의 패킷처리 최적화 기술인 DPDK(Data Plane Development Kit)가 제안되었다[6,7]. 그리고 DPDK기반의 VPP(Vector Packet Processing) 기술을 이용한 패킷고속화 처리 방법인 FD.io(Fast Data-Input/Output)가 제안 되었다[8]. 이와 같은 기술들은 데이터 패킷 처리 성능에 많은 영향을 미치는 기술이지만 NFV환경에서 사용되는 경우 실질적으로 얼마나 영향을 미치는지에 대한 부분은 확인이 되지 않았다. 따라서 본 논문에서는 NFV기반의 일반적인 네트워크 기능 및 vEPC와 같은 네트워크 기능 애플리케이션을 적용하여 네트워크 구조에 따른 성능을 비교 분석 하고자 한다.

2장에서는 클라우드 시스템에서의 일반적인 네트워크 구조와 패킷 최적화 및 고속화를 위한 I/O구조에 대해 연구한다. 3장은 NFV환경을 위한 네트워크 시스템의 구조를 설계하고 구현 한다. 마지막으로 본 논문의 네트워크 설계구조에 네트워크 기능 애플리케이션을 적용한 성능평가결과를 통해 마무리 한다.

Ⅱ. 관련 연구

본 절에서는 본 논문에서 제안하는 NFV환경에서의 네트워크 기능 성능향상을 위한 네트워크 가상화 기술에 대해 설명한다. 먼저 NFVI 환경 구성을 위해 NFV 플랫폼으로 사용되는 오픈소스인 오픈스택기반의 네트워크 구조에 대하여 간단히 기술하고, 오픈스택에서 사용하는 전가상화 및 반가상화 기반의 네트워크 기술과, 네트워크 성능 향상을 위해 제안된 네트워크 하드웨어 가상화 기술 및 패킷 최적화 기술에 대해 설명한다.

2.1 오픈스택 시스템

오픈스택은 오픈소스 소프트웨어로 구성된 Cloud-OS 또는 CMS(Cloud Management System)이다. 클라우드 컴퓨팅을 위한 오픈소스 프로젝트는 기본적으로 컴퓨팅 인프라를 관리하는 Nova와 네트워크 시스템 관리를 위한 Neutron, Guest VM의 OS 이미지 관리를 위한 Glance, Guest VM의 block 스토리지 할당을 위한 Cinder, 사용자 권한 관리를 위한 Keystone, 그리고 End-User를 위한 Web-GUI인 Horizon 프로젝트 등으로 구성되어 있다. 네트워크 서비스를 제공하는 Neutron 시스템은 Nova 에서 생성한 Guest VM들의 연결성을 관리하고 VM간의 네트워크처리 기술을 제공 한다.

2.2 오픈스택 네트워크 구조

오픈스택에서는 네트워크 가상화를 위해 기본적으로 KVM(Kernel-based Virtual Machine)기반의 전가상화 및 반가상화 기술을 사용한다. Intel e1000 및 rtl8139와 같은 전가상화 네트워크 드라이버를 사용할 경우, 이더넷 프레임은 User mode에서 생성된 I/O 에뮬레이션과 Kernel mode에서 생성된 Tap 디바이스의 End-to-End 연결에 의해 전송이 이루어진다[9-11]. 이와 같이 전가상화 기반의 네트워크 드라이버를 사용 하는 경우, Guest VM과 Qemu(short for Quick Emulator)구간, Emulation과 Kernel mode구간, 그리고 Emulation과 Host OS의 Kernel 구간에서 Context switch가 빈번히 발생하는 문제가 있다[11]. 이를 개선

하기 위해 반가상화 기반기술이 개발 되었는데, 이는 Context switch의 오버헤드를 최소화하기 위해 Guest VM과 Qemu 영역에 virtio front-end와 back-end 드라이버를 설치하고, 그 사이에 shard memory를 통해 데이터를 교환 하도록 설계함으로써, Context switch의 오버헤드 줄여 네트워크 처리량을 향상 시켰다[12]. 반가상화 기술은 네트워크 성능을 향상시켰지만, User mode에서 Qemu를 통해 데이터를 처리하기 때문에 Context Switching이 발생하는 문제는 여전히 존재한다. 이를 개선하기 위해 Kernel 기반의 Vhost를 구현으로 virtio front-end 드라이버에서 DMA(Direct Memory Access) 통해 직접적으로 Kernel에 데이터를 쓰고 읽는 구조가 개발되었다. Vhost는 네트워크를 위한 vhost-net과 블록디바이스를 위한 vhost-block를 제공 하고 있다[13]. MACVTAP을 사용 할 경우 Vhost의 MACVTAP 인터페이스를 통해 직접적으로 물리적 인터페이스로 데이터를 전송하기 때문에 Context Switching으로 인한 지연시간 및 프로세싱 오버헤드를 줄일 수 있다[14,15].

2.3 고속 네트워크 플랫폼

높은 데이터 전송 및 처리 속도를 요구하는 모바일 인프라 환경의 적용을 고려하는 NFV 환경에서, 앞서 언급한 기술을 통한 네트워크 데이터 처리 성능은 기존 물리적 네트워크 장비 기반의 성능보다 현저하게 떨어질 수밖에 없으며, 이에 따라 데이터평면에 대한 다양한 요구사항이 필요하게 되면서 추가적으로 패킷을 빠르게 처리하기 위한 데이터 처리 기술들이 개발되었다. 이러한 기술들의 요구사항을 정의하기 위해 OPNFV 에서는 DPACC(Data Plane Acceleration)라는 프로젝트를 생성하여 요구사항들에 대한 정의를 주도 하고 있다[16]. DPACC 에서는 VNF의 이식성(portability)을 보장하기 위한 프레임워크와 API를 중심으로 연구가 진행되고 있으며 이와 관련된 업스트림 프로젝트들과의 연계를 통해 기능을 개발하고 있다. 이와 관련된 대표적인 네트워크 기술로 virtio와 SR-IOV, DPDK, FD.io가 있다. 먼저 SR-IOV는 PCI Passthrough와 같이 Guest VM에서 Host 서버의 PCI 디바이스로 직접적으로 접근하는 방식이다. 하지만 PCI Passthrough는 Guest VM에 종속적으로 동작한다는 제약이 있다.

그림 1과 같이 SR-IOV는 물리적 하드웨어에 하나의 PF(Physical Function)와 다수의 VF(Virtual Function)로 구성된다. PF는 VF의 구성을 관리하고 VF를 통해 들어오는 패킷데이터를 전송하기 위한 스

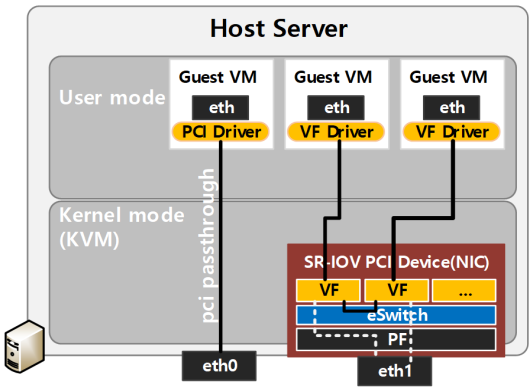


그림 1. PCI Passthrough 및 SR-IOV 구조도
Fig. 1. The structure of PCI Passthrough and SR-IOV

위칭 기능이 포함되어 있다. VF는 Guest VM에 할당되는 논리 인터페이스로 Guest VM의 VF 드라이버를 통해 물리네트워크에 직접적으로 접속하여 데이터를 처리 한다[17]. 하지만 네트워크 카드의 고가격 및 제조사의 하드웨어 스펙에 따른 VF 수의 제한은 시스템 설계에 고려될 필요가 있다. 다음으로 패킷 최적화 기술인 DPDK는 Kernel에서 네트워크 패킷을 처리하는 방식이 아닌 리눅스의 User mode에서 DPDK용 API를 이용하여 생성한 응용프로그램이 커널을 거치지 않고 하드웨어 및 메모리를 직접적으로 제어함으로써 비약적으로 네트워크 패킷처리 성능을 향상 시킬 수 있다. DPDK에서 패킷최적화를 위해 User mode에서 패킷을 처리하고 있지만, L3 Routing을 위해서는 여전히 커널의 네트워크 스택을 이용 하고 있다. 이는 FIB(Forwarding Information Base)의 사이즈가 커지

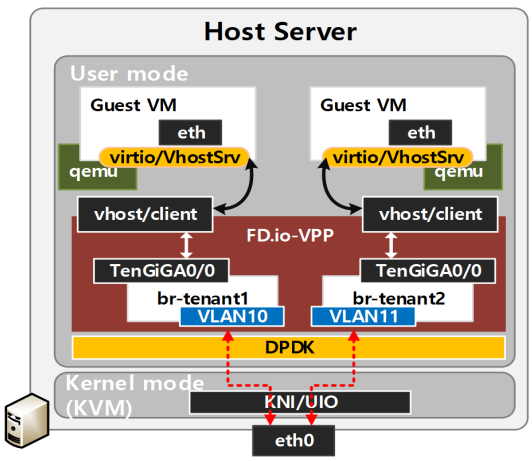


그림 2. FD.io 의 VPP 구조도
Fig. 2. The structure of FDio-VPP

는 경우 네트워크 처리 성능에 문제를 야기 시킬 수 있으며, 이를 해결하고 네트워크 성능을 향상시키기 위해 FD.io 기술이 제안되었다. 그림 2는 FD.io의 VPP(Vector Packet Processing)를 적용한 하드웨어 구성으로 DPDK기반의 kernel-bypass 기술을 이용하고 확장성이 높은 User mode에서 VPP에 의한 L2 및 L3 패킷최적화 처리 기술을 적용 하였다. DPDK 및 FD.io는 고가의 하드웨어 장비 없이 고성능의 패킷 처리가 가능하지만 DPDK를 이용한 응용프로그램 개발과 FD.io의 VM과 가상스위치간의 성능 확인 및 한정된 하드웨어 지원은 네트워크 설계에 있어 고려되어야 한다.

Ⅲ. 성능측정을 위한 구조 설계

본 절에서는 성능측정을 위한 실 구조를 설계하고, NFV환경에서 데이터 전달 성능을 측정하기 위한 테스트 베드 구성을 일반적인 VNF와 모바일 코어 네트워크 기능인 vEPC 컴포넌트를 구성 하여 성능측정을 진행 하였다. 일반적인 VNF는 특정 네트워크 기능 없이 데이터 포워딩 트래픽만 확인하였고, vEPC는 Fraunhofer FOKUS의 OpenEPC Rel.5를 사용하였으며, 컴포넌트 중 LTE관련 모듈을 VNF VM으로 구성하였다[18]. VNF구성은 ETSI에서 표준화 하고 있는 NFV Architecture를 기반으로 구성하였다. NFVI는 대표적인 클라우드 오픈 소스인 오프스택으로 구성하였으며 VNF 메니져는 오픈스택 프로젝트인 Tacker 프로젝트를 통해 구성하였다. 이때 두 대의 물리서버로 구성하였고, 한 대의 물리서버에는 Controller와 Compute1 노드, 그리고 나머지 한 대는 Compute2 노드로 구성하였다.

3.1 시스템의 네트워크 구조 설계

본 장에서는 NFVI 환경에서 사용되는 데이터 평면 가속화 기술들의 성능 측정을 위해 다양한 네트워크 구조를 구성하여 성능 측정을 진행하였다. 네트워크 가상화 기술에 따른 성능 비교를 위해, 전가상화와 반가상화 방법 및 SR-IOV, DPDK[19], FD.io의 VPP 적용 환경을 각각 구성하였으며, 반가상화 경우 virtio-net의 Vhost를 사용하는 경우와 MACVTAP을 사용하는 경우를 포함하여 실험 하였다. 네트워크 구성은 그림 3과 같이 외부인터페이스(EXT), SR-IOV를 위한 브릿지 네트워크(br-sriov)와 VF 또는 MACVTAP을 위한 인터페이스(MLX0), 그리고 DPDK0로 맵핑된 인터페이스로 구성 하였다.

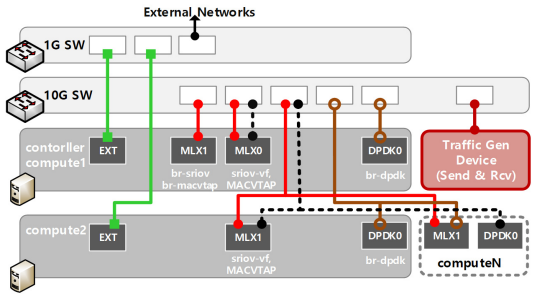


그림 3. 2대의 물리서버에서의 Controller, Compute 노드 구조도
Fig. 3. Controller and Compute node structure on two physical servers

Compute1 노드에서만 Guest VM을 제공할 경우 DPDK0 물리 인터페이스는 사용되지 않는다.

오픈스택 시스템을 한 대의 물리서버로 구성할 경우 SR-IOV 및 MACVTAP 네트워크를 위해서는 두 개의 NIC가 필요 하다. SR-IOV의 경우 하나의 NIC는 Guest VM을 위한 VF 네트워크 인터페이스로 사용 된다. 이는 Guest VM에서 데이터를 전송 할 때 Qemu에서 관리되는 bridge 스위치로 전송되는 것이 아니라, Host서버의 SR-IOV 네트워크로 직접적으로 전송되며, SR-IOV의 PF 스위칭에 의해 VF로 설정된 인터페이스로 패킷이 전달된다. 나머지 하나는 오픈스택 네트워크에서 사용되는 브릿지 네트워크로 L3 데이터를 처리하기 위해서 사용된다. 이는 VF 인터페이스에서 전송된 데이터를 브릿지에 설정된 가상 라우터에 의해 패킷을 처리하게 되며, 외부 연동 및 내부의 다른 Guest VM과 송수신 할 때 사용 된다. MACVTAP의 경우에도 이와 동일한 방법으로 패킷 흐름을 가진다.

하나의 서버로 구성하는 경우 컴퓨트 노드에 있는 VNF간의 데이터평면 성능 측정은 호스트의 커널기반의 성능측정값이기 때문에 실제 환경과는 상이 할 수 있다. 따라서 데이터평면의 속도측정을 위해 최소구성 단위인 두 대의 물리서버로 구성하였다. 두 대 이상의 물리서버로 실험 환경을 구축하는 경우, 컴퓨트 노드가 병렬연결 형태로 구성 되어야하기 때문에 결과 값이 물리적 서버의 개수의 증가에 따라 점차 증가하는 형태로 나타나게 된다. 그러나 본 실험은 데이터 평면 가속화 기술 적용 유무에 따른 성능 차이를 측정하고 분석하는 것을 목표로 하기 때문에 두 대 이상의 환경에서는 테스트를 진행하지 않았다.

본 논문에서는 서비스 처리 속도 향상을 위한 기법들의 성능 비교를 위해 크게 네트워크 가속화와 데이

터 처리 속도 가속화 기법들을 구성하고 실험을 실시하였다. 네트워크 가속화 기법들의 실험을 위해 Kernel 및 User mode, 하드웨어 기반의 데이터 처리 방식으로 분류하고 성능측정을 실시하였다. 먼저 대표적인 Kernel mode의 데이터 전송 방법 측정을 위해 virtio-net과 Vhost-net 환경에서의 측정을 하였다. 두 번째로 User mode의 데이터 전송방식 측정을 위해 DPDK 와 VPP 환경을 구성하였다. 두가지 방법 모두 User mode에서의 응용 데이터 처리 엔진을 통해 데이터가 처리되기 때문에 Guest VM에는 일반 소켓프로그램을 사용 한다. 하지만 각 엔진에 따른 성능 비교를 위해 개별 구성 후 실험을 진행하였다. 하드웨어 기반의 데이터 가속화 방식의 데이터 전송 방법 측정을 위해 SR-IOV로 네트워크 환경을 구성 하였다. SR-IOV로 구성하는 경우, 데이터가 통합 브릿지를 거치지 않고 VF에서 PF로 전달 되어 데이터를 처리하는 방법을 사용한다. 추가적으로 전체 네트워크 기능의 서비스 속도 향상에 영향을 미칠 수 있는 데이터 속도 가속화 기법에 대한 성능 측정도 실시하였다. 데이터 속도 가속화 기법의 성능 측정을 위해 테스트로 사용되는 환경에 가상화 VM의 멀티큐 할당 및 NUMA(Non-Uniform Memory Access)를 적용하고 진행하였으며 Guest VM을 통해 처리성능을 확인하였다.

Ⅳ. 성능 평가

본 장에서는 NFV기반의 데이터 패킷 가속화 기술의 성능 분석을 위해 다양한 네트워크 기술을 적용하여 실험을 진행 하였다. 먼저 실험환경에서 데이터처리 속도를 측정하기 위해 VSPERF (vSwitch Performance)를 설치하여 컴퓨터 노드에 각각 생성한 VNF 간의 네트워크 트래픽에 대한 성능을 측정 하였다. 일반 VNF에서의 성능측정 구조는 그림 4와 같다.

성능측정을 위한 테스트 시나리오는 Phy2Phy (Physical to Physical), PVP(VM Loopback), PVVP (Two VM Loopback) 방법을 사용하였다. Phy2Phy는

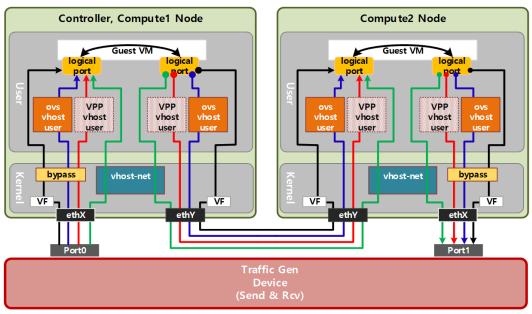


그림 4. NFV환경에서의 2개의 VM간 네트워크 트래픽 측정
Fig. 4. Measure network traffic between two VMs in an NFV environment

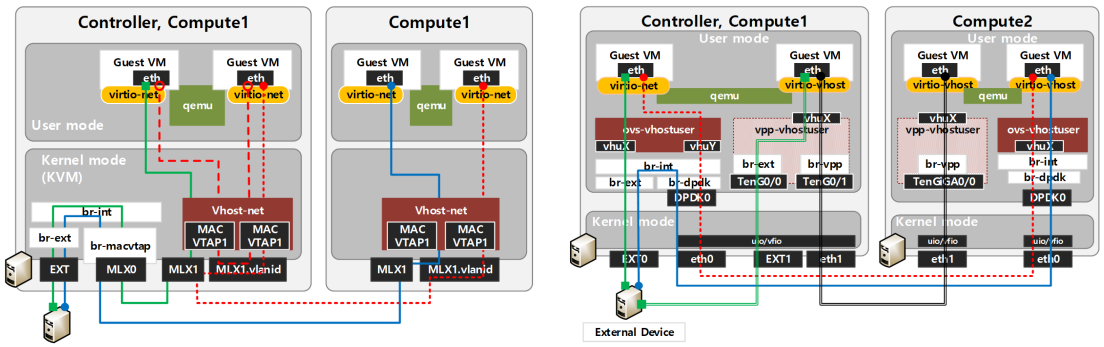
표 1. 오픈스택 Host 사양 및 테스트 사양
Table. 1. Specification of Openstack Host and TestBed

NODE	Classification		Specification
Controller Compute1 / Compute2	CPU		Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz * 2
	MEMORY		DDR4 2133 MHz 16GB * 10
	EXTERNAL NIC		Ethernet Controller 10-Gigabit X540-AT2
	DPDK NIC		Intel 82599ES 10-Gigabit SFI/SFP+ Network
	SR-IOV NIC		Mellanox ConnectX-3 Pro EN 10G SFP+ 2Port
	OS		Ubuntu 16.04 Server LTS
	Openstack		Ocata (include dpdk version 16.11 and vpp v17.01) and Tacker
	switch		znyx b1 10G
TestBed	Traffic Generator		VSPERF-dummy mode (https://wiki.opnfv.org/display/vsperf/VSperf+Home)
	Test VNF		vEPC_vnf (FOKUS OpenEPC Rel.5)
			vloop_vnf (http://artifacts.opnfv.org/vswitchperf/vnf/vloop-vnf-ubuntu.qcow2)
	bandwidth tool		netperf (http://www.netperf.org/netperf/)
	latency tool		sockperf (https://github.com/Mellanox/sockperf)
	CASE	Phy2Phy	Physical port ⇔ vSwitch ⇔ Physical port
		PVP	Physical port ⇔ vSwitch ⇔ VNF ⇔ vSwitch ⇔ Physical port
		PVVP	Physical port ⇔ vSwitch ⇔ VNF ⇔ vSwitch ⇔ VNF ⇔ vSwitch ⇔ Physical port

호스트 서버의 가상화 스위치에 대한 네트워크 성능을 측정하는 방법으로 패킷이 호스트의 NIC를 통해 가상화 스위치를 거쳐, 다시 NIC로 나가는 구조이고, PVP는 Phy2Phy의 가상화 스위치에서 VM을 한 번 더 거치는 구조로 되어있다. PVVP 같은 경우 PVP의 VM에서 다른 VM을 통해 호스트의 NIC으로 나가는 구조로 되어 있으며, VNF에서의 네트워크 성능측정을 위해 PVP 및 PVVP를 기준으로 테스트 실시하였다. 성능측정을 위한 하드웨어 및 시스템 환경은 표1과 같다.

4.1 Kernel mode vs User mode 기반 Guest VM의 네트워크 성능 비교

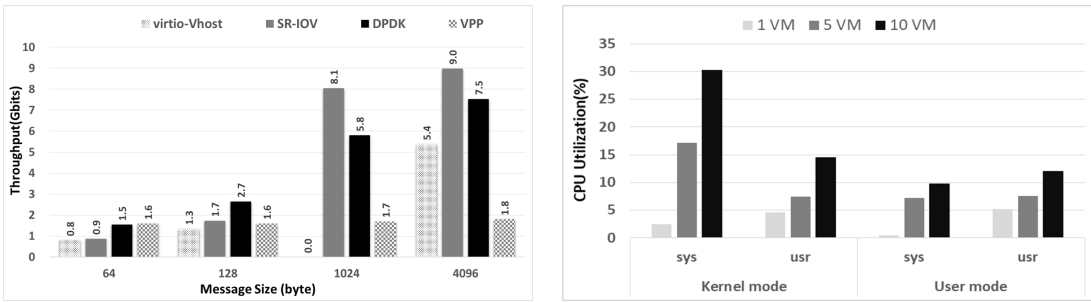
그림 6은 그림 5의 구조에서 메시지 패킷 사이즈에 대한 처리량과 호스트의 CPU 이용률을 측정한 결과이다. 그림 6(a)와 같이 virtio-vhost에서는 성능이 저하 된 것을 확인 할 수 있는데 이는 Guest VM에서 발생한 패킷이 Guest OS의 커널을 거쳐 호스트의 커널로 전달되면서 발생되는 Context Switching 오버헤드가 원인이다. 하지만 단일 호스트에서 Guest VM을 제공하는 경우, VM간의 패킷전달에 호스트의 loopback의 프로세스간 통신을 사용하기 때문에 하드웨어 성능에 따라 달라 질 수 있는데, 표 1과 같은 하드웨어 구성으로 사용한 경우 30Gb/s 처리성능이 측정되었다. User 기반인 DPDK에서 10G NIC를 사용한 경우 4KB 메시지 전송 시 7Gb/s 의 데이터가 측정되었는데, 이는 NIC에서 User mode의 Packet I/O 간 Kernel bypass 기술을 사용하여 고속화 처리를 하고 있지만, 라우팅에 대한 Kernel decision으로 여전히 커널의 FIB(Forwarding Information Base)를 참조하며 처리하고 있기 때문에 FIB의 증가 시 라우팅 처리에 대한 오버헤드가 성능저하로 이어졌다. 처리량 결과에 있어 DPDK기반의 VPP에서는 virtio-vhost 보다 낮게 측정되었는데, VPP의 가상 브릿지와 Guest VM의 인터페이스간의 네트워크 처리에 대한 병목으로 인한 오버헤드로 성능이 낮게 측정 되었다. 각 기술의 네트워크 외 리소스 자원 활용을 확인하기 위해 CPU자원 활용에 대한 측정도 함께 진행하였다. 그림 6(b)와 같이 다중 VM이 동작하는 환경에서 User mode 기반의 네트워크를 사용 할 경우 sys(호스트의



(a) Kernel mode architecture

(b) User mode architecture

그림 5. Kernel 및 User mode에서의 네트워크 구조
Fig. 5. Network environment in kernel and user mode



(a) Throughput in Kernel and User mode network

(b) CPU Utilization in Host

그림 6. Kernel 및 User mode에서의 성능 측정
Fig. 6. Performance measurement in Kernel and User mode

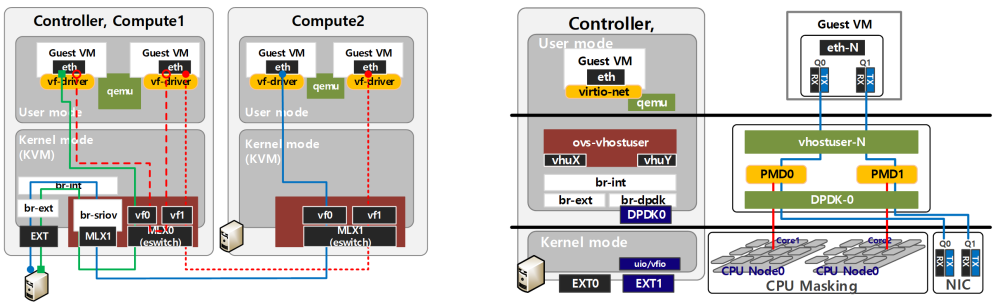
Kernel 기반의 CPU 사용률) 및 usr (호스트의 User 기반의 CPU 사용률)에서 사용되는 CPU 자원이 더 효율적임을 알 수 있다.

4.2 고속네트워크 성능 비교 (SR-IOV, DPDK and VPP)

본 절에서는 호스트에서 kernel-bypass에 의한 고속 네트워크 기술인 SR-IOV, DPDK, VPP 기술에 대한 성능 비교를 실시하였다. DPDK 및 VPP의 네트워크 구조는 그림 5(b)와 같고, SR-IOV는 그림 7(a)와 같은 구조로 구성하였으며, 포트구성은 하드웨어에서 지원하는 VF수와 1개의 PF로 구성하였다. Guest VM에서 데이터를 전송 시 Kernel mode의 Openvswitch와 같은 통합브릿지를 거치지 않고 네트워크의 VF로 직접적으로 전달되어 PF에 임베디드된 스위치에 의해 데이터를 처리 하게 된다. DHCP 및 라우팅을 위해서는 VF에서 사용하는 포트(MLX1 또는 MLX0)로 데이터를 전송하고, Controller 노드의 MLX1에 할당된 브릿지에 의해 DHCP 및 가상라우터에서 L3데이터를 처리한다. Controller 노드에서 SR-IOV를 위한 브릿지를 생성 할 경우 사용한 네트워크 포트이름과 Compute2 노드의 VF에 할당한 네트워크 이름을 동일하게 설정 해야만 정상적으로 네트워크 데이터를 처리 할 수 있다. SR-IOV는 Guest VM에서 호스트의 NIC로 직접적으로 데이터를 전송하여 처리하고, DPDK에서는 호스트의 ovs-dpdk서버의 vhostuser 소켓으로 데이터를 보내고 수신된 데이터를 DPDK 라이브러리를 이용해 호스트 NIC로 전송한다. 이때 ovs-dpdk 처리로 인한 오버헤드 발생으로 그림 8(a)와 같이 SR-IOV와 차이를 볼 수 있다. 그림 8(b)는 그림 7(b)와 같이 멀티큐를 적용한 구조에서 측정한 결과 값이다. 2개의 CPU 노드에 DPDK PMD(Poll Mode Driver)를 위한 CPU Core를 각 1개씩 설정하고, 그리고 Guest VM에서 사용할 큐만큼 CPU 개수를 할당한 후 Guest VM의 NIC에 멀티큐 설정을 위한 RX/TX Channel를 설정한다. 결과 값에서 테스트를 위한 패킷발생기의 동시연결수가 증가 할수록 멀티큐를 적용한 경우, 더 좋은 결과가 측정됨을 볼 수 있다.

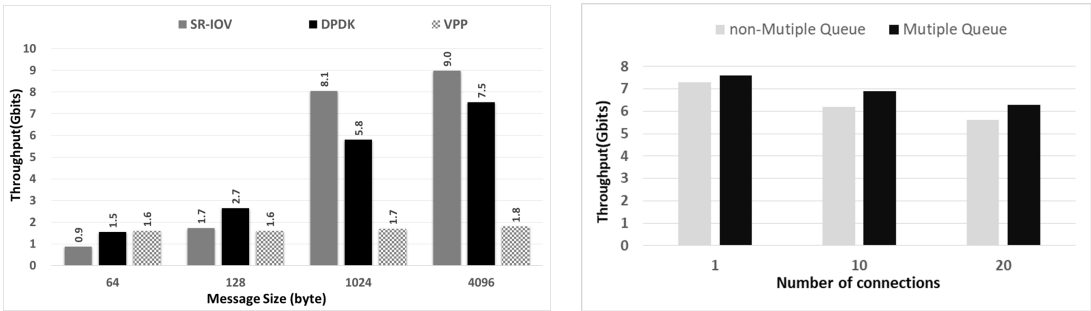
4.3 NUMA(Non-uniform memory access) 설정에 의한 고속 네트워크 구조

NUMA는 다중 프로세서의 메모리 접근에 대한 방



(a) SR-IOV network structure (b)RX/TX Multiple-Queue structure

그림 7. 2대의 물리서버에 적용한 SR-IOV 네트워크 구조와 vhostuser에서의 멀티큐 구조
Fig. 7. SR-IOV network structure and Multiple-Queue structure in vhostuser



(a) Throughput in SR-IOV, DPDK, VPP (b) Throughput by Multiple-queue in Guest VM

그림 8. 고속네트워크 (SR-IOV, DPDK, VPP) 및 멀티큐 구조에서의 성능 비교
Fig. 8. Performance comparison in high-speed network (SR-IOV, DPDK, VPP) and Multiple-Queue structure

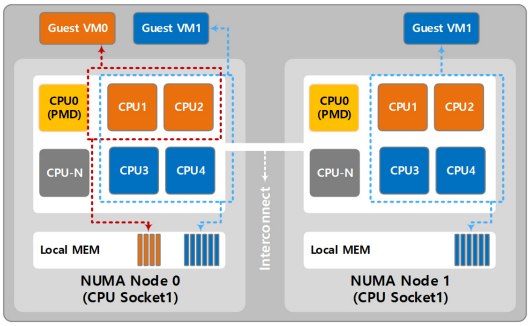


그림 9. 성능개선을 위한 NUMA 구조
Fig. 9. NUMA structure for performance improvement

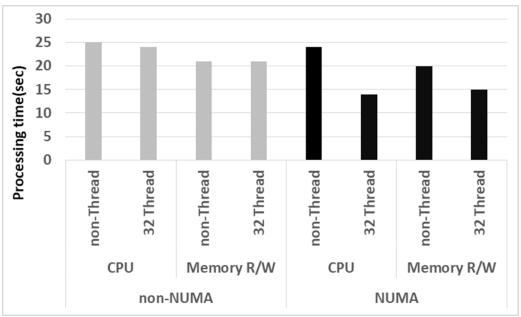
법으로, 응용프로그램이 실행되는 프로세서의 노드가 동일한 셀에 위치한 메모리에 보다 빠르게 접근하여 최상의 프로세스 성능을 얻기 위한 기법이다[20]. 이번 절에서는 NUMA 설계에 따른 Guest VM에서의 성능을 측정 후 분석 하였다.

그림 9는 성능실험을 위한 NUMA 구조로 2개의 NUMA 노드에 각 노드의 첫 번째는 PMD를 위한 코어로, 그리고 각 NUMA 노드에 대한 나머지 코어를 VM을 위한 코어로 할당 하였으며, CPU Core Frequency는 performance 모드로 설정 후 실험을 진행 하였다. 그림 10(a)와 같이 NUMA를 적용하고 Guest VM을 위한 전용의 CPU 할당 후 Guest VM의 성능측정(CPU 및 메모리 읽기/쓰기)을 진행 하였다. 쓰레드를 생성하지 않고 측정한 결과 non-NUMA의 Guest VM과 차이가 없지만, 32개의 쓰레드를 생성 후 진행 한 결과 약 1.3배 더 빠른 결과를 얻었다. 네트워크 처리에 있어 Guest VM을 10개까지 생성 후 네트워크 패킷 전송 테스트를 한 경우 호스트의 TLB(Translation Lookaside Buffer) miss에 대한 측정값이 Hugepages를 적용한 구조가 더 낮게 측정됨을 확인 할 수 있다. 이는 Guest VM의 수 증가 및 네트워크 패킷의 증가로 호스트 서버의 data-TLB 및 instruction-TLB miss에 의한 성능저하가 발생 할 수 있기에 NF를 위한 고속네트워크 지원을 위해서 고려되어야 할 사항이다.

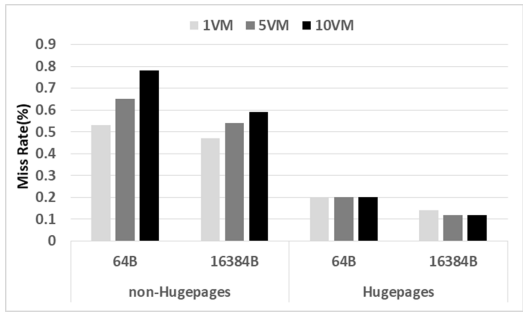
4.4 vEPC 환경에서의 네트워크 가속화 기술

테스트 환경이 아닌 실제 NFV가 사용되는 환경에서의 가속화 기술 성능 측정을 위해 vEPC 구조를 적용하여 추가 성능측정을 실시하였다. 성능측정을 위한 EPC컴포넌트는 그림 11과 같이 Compute1 노드에는 말단사용자를 위한 UE(User Equipment)와 EPC 코어 네트워크 컴포넌트를 배치하였으며, Compute2 노드에는 코어네트워크의 SGi-LAN으로 구성된 IGW(Internet Gateway)와 네트워크 트래픽 측정 서버를 설치하였다. 성능측정 구간은 Compute1 노드에 배치된 UE에서 Compute2 노드에 설치된 Perf 서버까지다.

그림 12와 같이 vEPC를 설치한 환경에서는 DPDK 및 SR-IOV를 사용한 경우 600Mbits/s 정도의 가장 높은 처리량을 보여 주고 있으며, VPP를 사용한 경우 가속화 기술의 절반 정도의 처리량을 보여 주었다. 이와 더불어 virtio-Vhost의 기본 환경 및 MACVTAP을 사용한 경우 처리량이 거의 0에 가까운 값이 측정 되었다. 이는 virtio 드라이버를 적용한 경우 raw ethernet 데이터를 처리하지 못해 정상적인 측정이 불가능하기 때문에 나온 결과로 분석된다. 반면 virtio-Vhost의 checksum과 offload의 기능을 끄고 측정한 결과 180Mbits/s 정도의 처리량이 측정되었지만, 일반적인 VNF를 사용한 경우에는 vhost-net의 절반 정도의 처리량이 측정되었다. 이외에 전가상화 기반의 네트워크 기술인 e1000 및 rtl8139 드라이버를 사용한



(a) CPU utilization in Guest VM



(b) TLB miss in HOST

그림 10. NUMA 구조에서의 성능측정
Fig. 10. Performance measurement in NUMA architecture

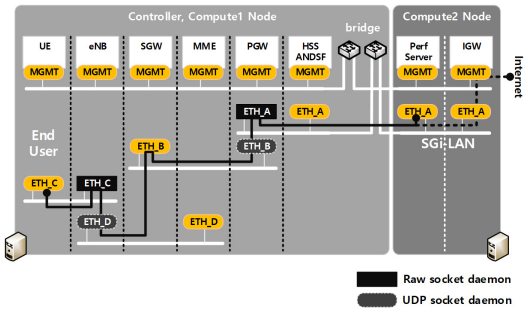


그림 11. NFV환경에서의 vEPC 컴포넌트 배치 구조
Fig. 11. vEPC network architecture in NFV environment

경우 50~60Mbits/s 처리량을 보여준다. 결과적으로 네트워크 가속화 기술인 SR-IOV 및 DPDK에서 raw ethernet 처리 지원으로 인해 전가상화 드라이버(e1000, rtl8193 등)보다 10배, 그리고 vhost-net의 offload checksum off 보다 3배에 가까운 데이터처리량을 확인하였다. 즉, 데이터평면 가속화기술인 SR-IOV 및 DPDK, VPP와 같은 네트워크 기능을 제공하는 VNF에서 더 나은 성능을 보여줌으로써 NFV 환경에서 필수 고려 요소임을 확인 할 수 있었다. 그리고 네트워크 가속화 기술에 RX/TX에 대한 멀티큐 적용 및 하드웨어 성능향상을 위한 NUMA 기술도 위 성능 검증에 의해 고려되어야 할 기술임을 볼 수 있다.

V. 결 론

본 논문에서는 데이터 평면 가속화 기술이 NFV 환경에 미치는 영향을 알아보기 위해 일반적인 환경과 네트워크 기능이 설치된 환경에서의 성능 측정 후 결과를 분석하였다.

실험을 위해 NFVI를 위한 VIM을 오픈스택을 사용하여 구성하였고 구성된 환경에 EPC를 구현하여 데이터평면의 가속화를 위한 다양한 네트워크 기술들을 적용하고 분석 하였다. 이를 통해 현재 나와 있는 가속화 기술인 SR-IOV 및 DPDK는 실제 네트워크 기능에서 사용되는 NFV 환경에서 일반적인 네트워크 기술들에 비해 더 나은 성능을 보이는 것을 확인 할 수 있었다. DPDK를 기반으로 하는 VPP에서는 PVP 또는 PVVP경우, VM의 Logical Port에서 VPP의 vSwitch에 생성된 브릿지 네트워크간의 성능저하로 인해 virtio-Vhost의 offload checksum off 한 경우와 비슷한 결과를 보여 주었다. 현재 VPP 프로젝트의 생성이 초기인 만큼, 이후 병목현상 제거로 DPDK 경우와 비슷하거나 더 좋은 결과를 보여줄 것으로 예상된다. 이는 일반적인 VM 환경에서 네트워크 기능을 설치해 측정한 결과와 비슷한 형태의 속도 증가를 보여주었으며, 이를 통해 데이터 평면 가속화 기술은 NFV 환경에서 필수 고려 요소임을 확인 할 수 있었다. 하지만 대표적인 데이터 평면 가속화 기술인 SR-IOV는 하드웨어의 고가격화 및 한정된 VF수와 같은 문제를 가지고 있었다. 또 다른 가속화 기술인 DPDK는 Hugepages를 위한 메모리 사용 및 제한된 네트워크 인터페이스 등의 문제를, 그리고 DPDK의 기반의 VPP 기술은 VM과 vSwitch 간의 성능저하 등을 확인할 수 있었다. 따라서 데이터 평면기술을 사용하는 경우 제공하는 네트워크 기능에 맞춰 시스템 설계 시 고려되어야 한다. 차후 연구에서는 본 논문에서 제안한 구조를 바탕으로 확장성을 고려한 네트워크 서비스 구조에 대한 연구를 진행 할 것이다.

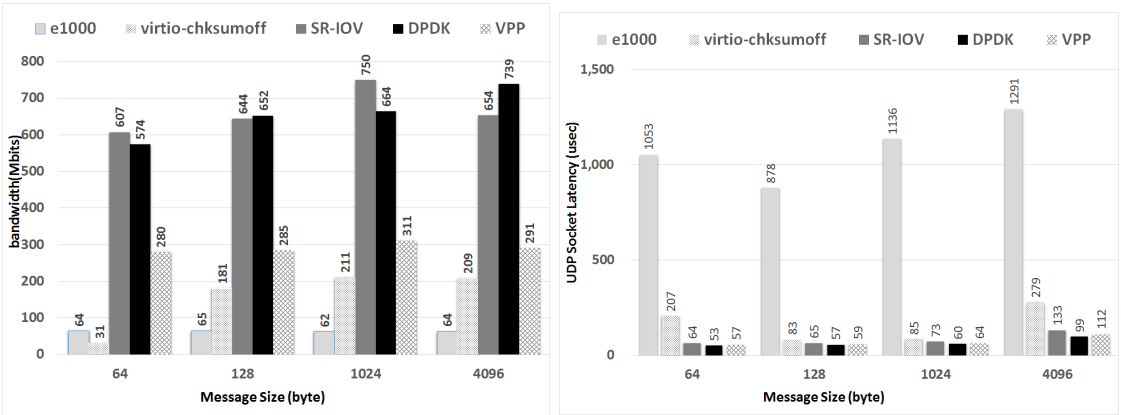


그림 12. NFV 환경에서의 vEPC 네트워크 처리율
Fig. 12. The throughput of vEPC network in NFV environment

References

[1] R. Kumar, N. Gupta, S. Charu, K. Jain, and S. K. Jangir, "Open source solution for cloud computing platform using OpenStack," *IJCSMC*, vol. 3, no. 5, pp. 89-98, May 2014.

[2] H. Kim and H. Kim, "Control algorithm for virtual machine-level fairness in virtualized cloud data center," *J. KICS*, vol. 38, no. 6, pp. 512-520, Jun. 2013.

[3] J. H. Yang, H. J. Park, Y. R. Kim, and J. K. Choi, "A virtual object hosting technology for IoT device controlling on wireless AP's," *J. KICS*, vol. 39, no. 2, pp. 164-172, Feb. 2014.

[4] M. S Kim, G. W. Lee, S. J Choo, S. H. Pack, and Y. H. Kim, "Optimal flow distribution algorithm for efficient service function chaining," *J. KICS*, vol. 40, no. 6, pp. 1032-1039, Jun. 2015.

[5] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 98-106, Feb. 2015.

[6] Y. Dong, Z. Yu, and G. Rose, SR-IOV *Networking in Xen: Architecture, Design and Implementation*, Retrieved Jun. 5, 2016, form https://www.usenix.org/legacy/event/wiov08/tech/full_papers/dong/dong_html.

[7] INTEL, *D.P.D.K. Data Plane Development Kit*(2014), Retrieved Jul. 4, 2016, from http://dpdk.org.

[8] Linux Foundation Project, FD.io. FastData IO(2017), Retrieved Mar. 3, 2017, from http://fd.io.

[9] I. HABIB, *Virtualization with kvm*, Linux J.(2008), Retrieved Aug. 3, 2016, from http://dl.acm.org/citation.cfm?id=1344217.

[10] J. Liu, "Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support," *IEEE IPDPS*, pp. 1-12, Atlanta, USA, Apr. 2010.

[11] B. Zhang, X. Wang, R. Lai, L. Yang, Y. Luo, X. Li, and Z. Wang, "A survey on i/o virtualization and optimization," *IEEE 2010 Fifth Annual ChinaGrid Conf.*, pp. 117-123, Guangzhou, China, Jul. 2010.

[12] G. Motika and S. Weiss, "Virtio network paravirtualization driver: Implementation and performance of a de-facto standard," *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 36-47, Jan. 2012.

[13] N. Har'El, A. Gordon, A. Landau, M. Ben-Yehuda, A. Traeger, and R. Ladelsky, "Efficient and scalable paravirtual I/O system," *USENIX ATC 13*, pp. 231-242, San Jose, USA, Jun. 2013.

[14] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with SR-IOV," *J. Parall. and Distrib. Comput.*, vol. 72, no. 11, pp. 1471-1480, Nov. 2012.

[15] V. Kashyap, A. Bergman, S. Berger, G. Stenzel, and J. Osterkamp, "Automating virtual machine network profiles," *In: Linux Symp.*, pp. 147-152, Ottawa, Canada, Jul. 2010.

[16] DPACC, *OPNFV DPACC*, Retrived Jul. 15, 2016, from https://wiki.opnfv.org/display/dpacc/DPACC+Home

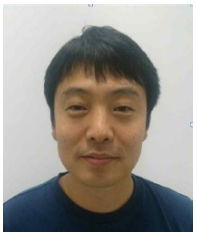
[17] L. Abeni, C. Kiraly, N. Li, and A. Bianco, "On the performance of KVM-based virtual routers," *Computer Commun.*, vol. 70, no. 1, pp. 40-53, Oct. 2015.

[18] OpenEPC, *Open Evolved Packet Core Ref. 5*, Retrived Jul. 15, 2015, from http://www.openepc.net/.

[19] J. DiGiglio and D. Ricci, *High performance, open standard virtualization with NFV and SDN*, White paper, Intel Corporation and Wind River, 2013.

[20] A. Banerjee, R. Mehta, and Z. Shen, "Supporting NUMA-Aware I/O in virtual machines," in *IEEE Micro,* vol. 36, no. 4, pp. 28-36, Aug. 2016.

박 영 기 (Young-ki Park)



2013년 8월 : 숭실대학교 정보통신공학 석사
2014년 3월~현재 : 숭실대학교 정보통신공학 박사과정
<관심분야> Cloud, OPNFV, IoT, FastData Stack

양 현 식 (Hyun-sik Yang)

2013년 2월 : 숭실대학교 정보통신전자공학부 학사
2013년 3월~현재 : 숭실대학교 정보통신소재융합학과 석박통합과정
<관심분야> OPNFV, VNF Monitoring, Blockchain

김 영 한 (Young-han Kim)

1986년 2월 : 한국과학기술원 전기 및 전자 공학과 석사
1990년 2월 : 한국과학기술원 전기 및 전자 공학과 박사
1994년~현재 : 숭실대학교 정보통신전자공학부 교수

<관심분야> OPNFV, SDN/NFV, IoT, CI/CD, Blockchain