KNOM Review '18-02 Vol.21 No.02
https://doi.org/10.22670//knom.2018.21.2.26

가상 네트워크 트래픽 모니터링을 위한 eBPF 기반 Virtual TAP 설계 및 구현

홍지범◆°, 정세연*, 유재형**, 홍원기*

Design and Implementation of eBPF-based Virtual TAP for Inter-VM Traffic Monitoring

Jibum Hong◆°, Seyeon Jeong*, Jae-Hyung Yoo**, James Won-Ki Hong*

요 약

클라우드 컴퓨팅 및 서비스의 확산으로 인터넷 트래픽과 서비스 품질 향상에 대한 요구가 증가하면서 데이터 센터 내부 서버의 리소스를 보다 효율적으로 사용하는 서버 가상화와 네트워크 가상화 기술에 대한 관심이 증가하고 있다. 트래픽 모니터링을 위해 패킷을 복제하는 기존의 하드웨어 TAP (Test Access Port) 장비는 서버 가상화 환경에서 구성된 가상 데이터 경로 (datapath)에 적합하지 않기 때문에 하드웨어 TAP 장비를 소프트웨어로 구현한 Virtual TAP (vTAP)을 통해 가상 스위치에서 패킷을 복제한다. 그러나 가상 스위치에서 vTAP을 구현하면 호스트 머신의 컴퓨팅 리소스를 가상 스위치 및 가상 머신과 공유하기 때문에 성능 저하 문제가 발생한다. 이 문제를 극복하기 위해 고속 패킷 처리 기술인 eBPF (Extended Berkeley Packet Filter) 기반의 vTAP 구현 기술을 제안하고 기존 방법과 성능을 비교한다.

Key Words : Virtual Network Monitoring, extended Berkeley Packet Filter (eBPF), Linux Kernel Networking

ABSTRACT

With the proliferation of cloud computing and services, the internet traffic and the demand for better quality of service are increasing. For this reason, server virtualization and network virtualization technology, which uses the resources of internal servers in the data center more efficiently, is receiving increased attention. However, the existing hardware Test Access Port (TAP) equipment is unfit for deployment in the virtual datapaths configured for server virtualization. Virtual TAP (vTAP), which is a software version of the hardware TAP, overcomes this problem by duplicating packets in a virtual switch. However, implementation of vTAP in a virtual switch has a performance problem because it shares the computing resources of the host machines with virtual switch and other VMs. We propose a vTAP implementation technique based on the extended Berkeley Packet Filter (eBPF), which is a high-speed packet processing technology, and compare its performance with that of the existing vTAP.

※ 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2015-0-00575, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)
※ 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00749, 인공지능 기반 가상 네트워크 관리 기술 개발)
◆ First Author : Pohang University of Science and Technology Department of Computer Science Engineering, hosewq@postech.ac.kr
° Corresponding Author
* Pohang University of Science and Technology Department of Computer Science Engineering, {jsy0906, jwkhong}@postech.ac.kr
** Pohang University of Science and Technology Graduate school of Information Technology, styoo@postech.ac.kr
논문번호 : KNOM2018-02-08, Received December 8, 2018; Revised December 17, 2018; Accepted December 21, 2018

Ⅰ. 서 론

오늘날 클라우드 컴퓨팅은 웹 서비스, 가상화, 서비스 지향 아키텍처 등과 같은 많은 기술들을 결합하고 SaaS, PaaS 및 IaaS와 같은 IT capability를 실현하기 위해 비즈니스 모델을 결합한 형태이다 [1]. 또한 클라우드 인프라와 클라우드 서비스가 확산됨에 따라 서비스 품질 향상에 대한 요구 사항이 급속하게 증가하고 있다. 이러한 상황에서 가상 머신 (Virtual Machine, VM)을 사용하여 서버의 리소스를 효과적으로 사용하는 서버 가상화 (Server Virtualization) 기술과 이를 뒷받침하는 네트워크 가상화 기술이 많은 주목을 받고 있다. 데이터 센터 (Data Center)에서 동일한 호스트 서버에 위치한 VM들 또는 물리적으로 다른 서버에 위치한 VM들 사이의 통신은 다양한 서비스 요청을 처리하기 위해 동시다발적으로 발생한다. 따라서 데이터 센터의 VM들 사이의 가상 네트워크에서 발생하는 트래픽 (East-West 트래픽)이 크게 증가하고 있는 추세이다.

기존의 하드웨어 TAP (Test Access Port) 장비는 터미널과 라우터 (또는 스위치) 사이의 링크를 통과하는 패킷을 복제하고, 복제된 패킷을 모니터링 시스템에 전송한다. 하지만 클라우드 컴퓨팅 환경에서 동일한 호스트 머신에 위치하는 VM들 사이의 링크는 가상 스위치 (virtual switch)에 의해 구성된 가상 링크 (virtual link)이기 때문에 이러한 가상 링크에 하드웨어 TAP 장비를 설치할 수 없다는 문제점이 존재한다. 따라서 기존 하드웨어 TAP 장비로는 VM들 사이의 트래픽을 모니터링하기가 어렵다.

이러한 문제를 해결하기 위하여 하드웨어 TAP 장비를 소프트웨어로 구현한 virtual TAP (virtual Test Access Port, vTAP)은 VM 사이의 트래픽 모니터링을 패킷 레벨로 제공한다. vTAP은 패킷 레벨의 트래픽 모니터링 및 트래픽 엔지니어링을 위하여 각 호스트 서버에 위치하는 가상 스위치를 활용하여 구현할 수 있다.

본 논문은 클라우드 컴퓨팅 환경에서 트래픽 모니터링이 가능한 vTAP의 설계 및 구현을 제안한다. 가상 스위치 기반 vTAP 구현은 패킷을 복제하고 복제된 패킷을 모니터링 시스템으로 전달할 때 가상 스위치 및 다른 VM들이 사용하는 동일한 호스트 머신의 리소스를 공유하기 때문에 vTAP의 성능이 저하된다. 이 성능 저하 문제를 해결하기 위해 호스트 머신의 제한된 리소스를 최적으로 사용하기 위한 커널 기반의 고속 패킷 처리 기술인 eBPF (extended Berkeley Packet Filter)를 기반으로 하는 vTAP을 제안한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 제 2 장에서는 vTAP와 연관된 기술을 설명하고 이와 관련된 문제에 대해 설명한다. 제 3 장에서는 본 논문에서 제안하는 eBPF 기반 vTAP의 설계 및 구현에 대해 설명하고, 제 4 장에서는 eBPF 기반 vTAP과 가상 스위치를 기반으로 하는 기존 vTAP을 비교하여 성능을 평가한다. 마지막으로 제 5 장에서는 본 연구에 대한 결론 내리고 향후 과제에 대해 논의한다.

Ⅱ. 연구 배경 및 관련 연구

1. 연구 배경

vTAP은 일반적으로 패킷 레벨에서 호스트 머신 (서버)의 VM들 사이의 트래픽을 복제하고, 복제된 트래픽을 모니터링 시스템에 전달하여 트래픽을 분석한다. 본 논문에서 제안하는 vTAP의 설계 및 구현은 Linux 커널 기반의 하이퍼바이저 (hypervisor)인 KVM과 고속 패킷 처리를 지원하고 오픈 소스 프로젝트를 통해 확장성을 제공하는 eBPF를 기반으로 한다.

eBPF는 Linux 커뮤니티가 주도하는 오픈 소스 프로젝트인 IO Visor Project의 하나로, eBPF는 Linux에서 패킷 필터링 및 패킷 모니터링에 사용하는 기존 BPF의 성능과 보편성 (universality)을 확장하였다 [2]. eBPF는 주로 커널 기반의 tracing과 마이크로 세그멘테이션 (micro-segmentation), 보안 그룹 (security group), 방화벽과 같은 커널 내부에서 동작이 가능한 보안 기능에 활용되어 왔으며, Docker 및 Kubernetes와 같은 컨테이너 (container) 기술에 eBPF를 활용하는 사례도 존재한다.

기존 BPF는 상호 작용 (interaction)이 불가능하기 때문에 다른 BPF 프로그램을 호출할 수 없다. 또한 JIT (Just-In-Time) 컴파일이 복잡하고 프로그램 관리가 용이하지 않다는 단점이 있다. 이러한 이유로 Linux 커널 기반의 네트워크 응용프로그램이나 커널 기반의 네트워크 기능을 개발할 때 기존 BPF는 그 기능이 제한적이라는 단점이 있다. 반면 eBPF는 C 언어 및 Python과 같은 상위 레벨의 언어로 작성된 프로그램을 바이트 코드로 변환하고, 프로그램을 커널에 로드하여 커널 함수를 호출한다.

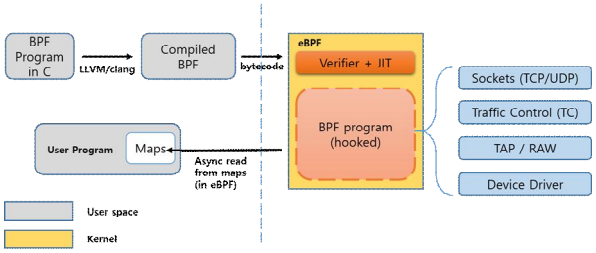

그림 1. eBPF의 전반적인 처리 과정
Fig. 1. Overall Process of eBPF

이 때 커널 내부의 verifier는 바이트 코드가 안전하지 않은 것으로 판단하는 경우 바이트 코드를 거부하여 프로그램을 커널에 로드하지 않는다. 커널 내부에 정상적으로 로드된 eBPF 프로그램은 패킷이 오갈 때 발생하는 네트워크 이벤트를 소켓, 트래픽 제어, 디바이스 드라이버와 같은 이벤트 핸들러와 연관시켜 해당 커널 함수를 호출한다. 본 논문에서 활용하는 eBPF의 전반적인 실행 구조는 그림 1에 나타나있으며, 위와 같은 eBPF 뿐만 아니라 다른 고속 패킷 처리 기술의 활용을 통한 네트워크 및 시스템 모니터링에 대한 관심이 증가하고 있다.

Open vSwitch (OVS) [3]는 VM 간 가상 네트워크 구축 환경을 제공하는 Linux 기반 멀티 레이어 가상 스위치이다. OVS는 프로그래밍이 가능한 확장 방식을 채택하여 대규모 네트워크 자동화가 가능하도록 설계되었다. OVS는 SDN (Software Defined Networking)에서 사용되는 OpenFlow 프로토콜 및 많은 표준 인터페이스를 지원하고, SDN 스위치로써 네트워크 구성 및 관리를 자동화하는데 사용된다.

2. 관련 연구

현재 서버 가상화 환경에서 가상 네트워크 모니터링을 통해 VM 간 트래픽에 대한 가시성을 제공하는 상용 vTAP 솔루션들이 존재하고 있다.

Veryx vTAP은 하이퍼바이저 상에 생성된 레이어에 있는 소프트웨어 컴포넌트로 vNIC (virtual Network Interface Controller)를 연결하는 가상 스위치와 함께 동작한다. Veryx vTAP은 트래픽을 모니터링하고 복제한 다음 이를 Network Analyzer 혹은 침입 탐지 시스템 (Intrusion Detection System, IDS)과 같은 외부 모니터링 시스템에 전달한다 [4]. Gigamon vTAP은 각 VM에 모니터링 에이전트를 설치하여 VM을 통해 들어오거나 나가는 패킷을 복제하고 복제된 패킷을 모니터링 플랫폼에 전송하여 가상화 환경에서 트래픽 가시성을 제공한다 [5]. 마지막으로 IXIA vTAP는 호스트 머신의 커널을 수정하는 방법을 통해 VM들 사이의 패킷을 수집하고 복제된 패킷을 외부 모니터링 프로그램으로 전송한다 [6].

위와 같은 상용 솔루션들의 경우 높은 처리 성능 (throughput)을 제공하고 시스템 리소스를 적게 사용하지만 vTAP이 포함된 상용 솔루션 대한 종속성을 가지기 때문에 사용자는 유지 관리 및 확장성에 어려움을 겪는다. 본 논문에서 제안하는 vTAP 설계 및 구현은 오픈 소스인 eBPF를 기반으로 하여 유지 관리 및 확장성 문제를 해결한다.

또한 상용 솔루션과는 별개로 SDN 및 OpenFlow 네트워크에서 효율적인 네트워크 모니터링을 위해 vTAP 기능을 활용하는 다양한 연구들이 존재한다. Planck는 상용 네트워크에서 millisecond 단위의 모니터링을 제공하는 네트워크 측정 시스템에 대한 연구로, 하드웨어 OpenFlow 스위치의 포트 미러링 (port mirroring) 기능을 사용하여 특정 트래픽 흐름을 샘플링하고 샘플링 정확도 및 데이터 수집 속도를 향상시킨다 [7]. NetAlytics는 클라우드 응용프로그램의 성능 분석 시스템으로 인텔의 고속 패킷 처리 도구인 DPDK (Data Plane Development Kit)를 사용하여 데이터를 처리한다. NetAlytics 또한 하드웨어 OpenFlow 스위치의 포트 미러링을 사용하여 VM들 사이에 이동하는 패킷의 복제본을 모니터링 시스템에 전달한다. 하지만 이 연구는 하나의 호스트 머신에 하나의 VM만을 가정하여 한계가 존재한다 [8].

vTAP과 같은 패킷 복제 기술은 기계 학습 (Machine Learning, ML)에도 적용되고 있다. 최근 각광받고 있는 self-driving 및 intelligent networking과 더불어 많은 ML 기술이 네트워크 분야에 도입됨에 따라, 관리자가 설정하기 어렵거나 사전에 정의되지 않은 규칙에 대한 네트워크 행동 모델을 구축할 때 vTAP 기능을 이용하여 패킷 헤더 값과 같이 학습에 필요한 다양한 feature 데이터를 제공한다. Amaral et al. [9]는 포트 미러링을 사용하여 복제된 패킷으로부터 time stamp, inter-arrival time, flow duration과 같은 feature 데이터를 추출한다. 이를 통해 기존의 룰 기반 (rule-based) 또는 시그니처 기반 (signature-based)의 패킷 검사 시스템 (packet inspection system)과 비교하여 패킷 분류에 대한 정확성을 향상시킨다.

Abubakar 및 Pranggono [10]는 패킷 및 flow 레벨의 메타데이터를 수집하기 위해 포트 미러링을 사용하여 ML 기반의 IDS를 구현한다.

위의 연구들은 네트워크 모니터링 및 머신러닝을 위한 패킷 수준 데이터 수집을 위해 스위치의 포트 미러링 또는 SPAN (Switched Port ANalyzer) 기능에 의존하고 있다. 포트미러링 기능은 스위치의 고유 인터페이스를 통해 각 스위치 별로 기능 설정이 필요하며, 해당 스위치의 패킷 포워딩 성능을 저하시킬 수 있다. 반면 본 연구는 스위치가 아닌 eBPF 활용이 가능한 리눅스 머신 상에서 주로 VM간 패킷을 고속으로 복제하기 위한 vTAP 기능 구현에 초점을 맞춘다.

이전 연구에서 우리는 OpenFlow 프로토콜을 사용하여 Open vSwitch 가상 스위치 기반으로 동작하는 vTAP을 구현하여 SDN 네트워크에서 TAP 정책을 적용하고 관리할 수 있는 방법을 제공하였다 [11]. 이 연구에서 우리는 가상 스위치 기반 vTAP이 가지는 성능 저하 문제를 해결하기 위해 고속 패킷 처리 도구인 DPDK를 활용하여 패킷 처리 속도를 향상시켰다. 본 논문은 이전 연구인 가상 스위치 기반 vTAP과 다른 접근법으로, eBPF를 사용하여 리눅스 VM 상에서 동작하는 vTAP을 설계 및 구현하고 그 성능을 검증한다. 이를 통해 가상 네트워크 기능 (Virtual Network Function, VNF)의 형태로 서비스 체인에 포함되어 특정 트래픽의 패킷을 효과적으로 복제, 모니터링할 수 있다.

Ⅲ. 시스템 디자인

그림 2는 서버 가상화 환경에서 하드웨어 TAP 장비의 한계점과 비교하여 vTAP과 가상 스위치의 동작 및 관계를 나타낸다. 서론에서 언급했듯이 가상 스위치를 기반으로 하는 vTAP은 패킷을 복제하고 복제된 패킷을 모니터링 시스템으로 전달하는 과정에서 가상 스위치 및 다른 VM들이 동일한 호스트 머신의 리소스를 공유하기 때문에 가상 스위치에서 vTAP 기능을 수행하는 과정에 성능 저하가 발생한다 [11]. 이를 해결하기 위해 본 논문에서는 Linux 호스트 머신 또는 VM을 가상 스위치 대신 패킷 복제 역할을 하도록 하여 eBPF를 기반으로 커널 내부에서 vTAP 기능이 동작하도록 구현한다.

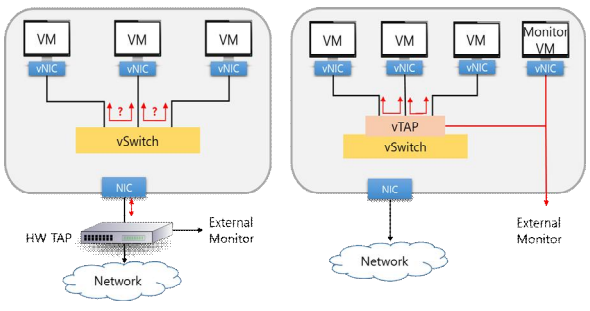

그림 2. 하드웨어 TAP과 Virtual TAP의 비교
Fig. 2. Hardware TAP and Virtual TAP

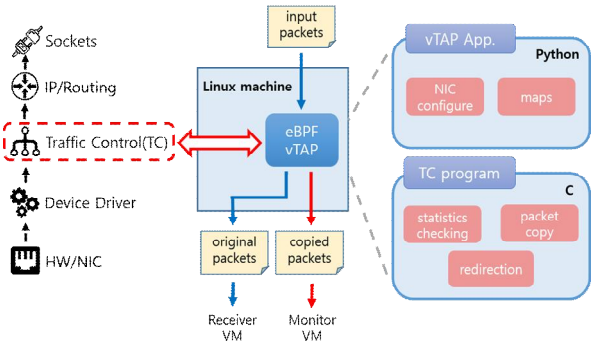

그림 3. eBPF 기반 vTAP 구조
Fig. 3. Overview of eBPF-based vTAP

그림 3은 본 논문에서 제안하는 eBPF 기반 vTAP 설계를 나타낸다. Linux 머신은 패킷을 전송하는 VM과 연결된 입력 포트를 가지고, 원본 패킷의 목적지인 Receiver VM과 모니터링 역할을 하는 Monitor VM과 연결된 출력 포트를 가진다. Linux 머신을 통해 전달되는 트래픽은 본 논문에서 구현한 eBPF 기반 vTAP에 의해 수집되며, vTAP은 패킷의 헤더 정보를 통해 패킷을 식별하고 패킷을 복제하여 복제된 패킷을 Monitor VM에 전달한다. vTAP은 2가지 컴포넌트로 구성되는데, Python으로 작성된 vTAP Application은 네트워크 인터페이스를 모니터링하고, C 언어로 작성된 TC Program은 BPF 라이브러리를 통해 패킷을 식별하고 패킷을 복제하여 모니터링 시스템에 전달하는 실질적인 vTAP 기능을 수행하는 로직을 설정한다.

앞서 설명한 eBPF 기반 vTAP의 컴포넌트를 보다 상세히 설명하자면 Python으로 작성된 vTAP Application은 바이트 코드가 바인딩되어 동작할 커널 내부 네트워크 스택의 후킹 (hooking) 포인트를 지정한다. 본 논문에서 제안하는 eBPF 기반 vTAP의 기능은 패킷이 Linux TC (Traffic Control)에 도달할 때 BPF API를 호출하여 구현하기 때문에 TC에 바인딩 된다. vTAP Application은 패킷이 전송되거나 도달할 때 발생하는 네트워크 이벤트에 대

해 TC Program을 호출한다.

호출된 TC Program은 소켓 버퍼를 읽은 후 IP 및 MAC 주소와 같은 패킷 헤더 정보를 식별하기 위해 Key-Value 기반의 Hash Map을 살펴보거나 초기화한다. 패킷의 헤더 정보에 source 또는 destination의 IP 주소가 모니터링 대상 리스트에 있는 경우 TC Program은 BPF 라이브러리를 이용하여 패킷을 복제하고 복제된 패킷을 모니터링 VM으로 전달하며, 리스트에 없는 경우에는 패킷은 vTAP을 통과하여 destination으로 전달된다.

Ⅳ. 결과 및 검증

본 논문에서 제안하는 eBPF 기반 vTAP의 기능을 검증하고 성능을 평가하기 위해 아래와 같은 테스트베드를 구성하였다. eBPF 기반 vTAP의 주목적은 vTAP의 처리 성능을 향상시키는 것이므로, 먼저 모니터링 VM에 전송되는 패킷의 처리량을 실험한다. 그리고 침입 탐지 시스템 (IDS)을 이용한 애플리케이션 레벨에서의 실험을 통해 제안하는 eBPF 기반 vTAP을 검증한다.

실험 환경은 하드웨어 및 가상 머신에 구축되었으며, 하드웨어로는 Intel Core i7-4790 (3.6GHz, 4개 코어) 프로세서, 16GB 메모리 및 256GB SSD를 사용하였다. 가상 환경은 Oracle VirtualBox (ver. 5.2.12)를 통해 구축하였고, 각 VM은 Ubuntu 18.04 (kernel 4.15.0) OS를 사용하며 QEMU/KVM 하이퍼바이저를 통해 관리된다.

1. 동일한 VM에 위치하는 Receiver 및 vTAP의 처리 성능 비교

본 절의 실험과 다음 절의 실험에서 우리는 제안하는 eBPF 기반 vTAP을 통해 모니터링 VM에 전달되는 패킷의 처리량을 측정하였다. 그리고 eBPF 기반 vTAP의 유효성을 검증하기 위하여 OVS 포트 미러링의 처리량과 비교하였다. OVS (ver. 2.9.0)는 동일한 환경에서 설치되었으며, OVS 포트 미러링은 OVSDB (Open vSwitch Database) 프로토콜을 통해 다른 모니터링 포트로 복제된 패킷을 전달한다. OVS 포트 미러링 기능을 사용하기 위해 우리는 source 포트 (Sender), destination 포트 (Receiver) 및 output 포트 (Monitor)를 지정하여 포트 미러링 정책을 적용하였다. 패킷은 고성능 네트워크 테스트 도구인 Pktgen [12]에 의해 생성되었

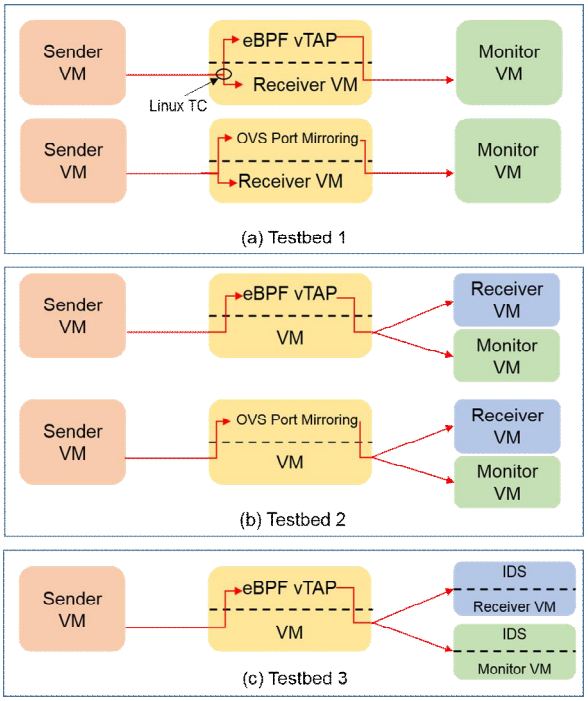

그림 4. 실험을 위한 테스트베드 구성도
Fig. 4. Testbeds for Experiments

고, Monitor VM에 전달된 복제된 패킷의 수신 측 (RX) 처리량을 측정하였다.

그림 4(a)는 본 절에서 수행하는 실험의 테스트베드를 나타낸다. Sender VM은 Receiver VM으로 UDP 패킷을 생성하여 보낸다. Receiver VM에서 동작하는 eBPF 기반 vTAP은 Receiver VM에 도달하는 패킷을 복제하고 복제된 패킷을 Monitor VM으로 전달한다. Monitor VM에서는 전달된 패킷의 처리 성능을 측정한다. 그림 5와 표 1은 본 절의 실험에서 eBPF 기반 vTAP과 OVS 포트 미러링을 통해 복제되어 Monitor VM으로 전달된 패킷의 처리 성능을 측정한 결과를 나타낸다. 우리는 이 실험에서 처리량, 초당 처리되는 패킷의 개수 (Packet Per Second, PPS) 및 CPU 사용량을 매초 마다 측정하였으며, 각 패킷 사이즈에 대해 20회 시행한 평균 측정값을 나타내었다.

먼저 OVS 포트 미러링은 패킷 사이즈가 64바이트인 경우 초당 약 96,000개의 패킷을 처리하였다. 패킷 사이즈가 증가하면서 초당 처리되는 패킷의 수는 약 87,000개로 감소하였다 (표 1). 처리량은 패킷 사이즈가 64바이트인 경우 약 38Mbps를 나타내었으며, 패킷 사이즈가 증가함에 따라 처리량은 약 1,002Mbps까지 증가하였다. 반면 본 논문에서 제안하는 eBPF 기반 vTAP은 패킷 사이즈가 64바

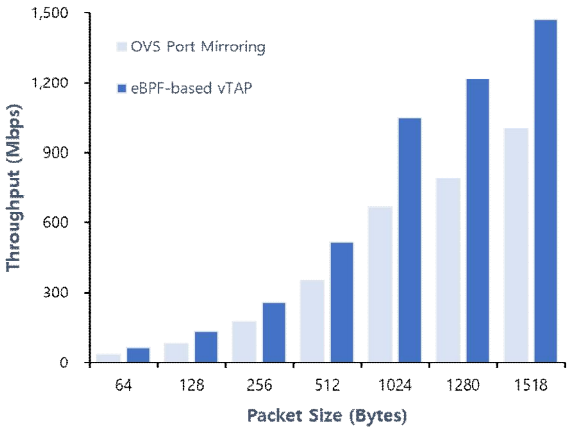

그림 5. vTAP과 포트미러링의 처리 성능 비교 (Testbed 1)
Fig. 5. Throughput Comparison of Testbed 1

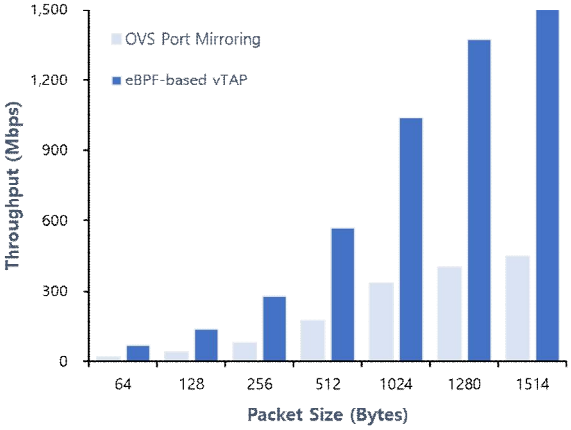

그림 6. vTAP과 포트미러링의 처리 성능 비교 (Testbed 2)
Fig. 6. Throughput Comparison of Testbed 2

표 1. 세부 성능 측정 결과 (Testbed 1)
Table 1. Detailed Performance Measurement of Testbed 1

Packet Size (bytes)	OVS Port Mirroring		eBPF-based vTAP	
	PPS	Throughput (Mbps)	PPS	Throughput (Mbps)
64	96,502	37.99	127,504	62.09
128	97,966	85.16	134,980	131.72
256	94,425	175.57	132,575	259.00
512	90,755	348.17	132,510	518.66
1024	87,158	669.23	134,326	1,050.34
1280	81,541	787.96	124,588	1,217.92
1518	87,475	1,002.42	126,952	1,473.10

표 2. 세부 성능 측정 결과 (Testbed 2)
Table 2. Detailed Performance Measurement of Testbed 2

Packet Size (bytes)	OVS Port Mirroring		eBPF-based vTAP	
	PPS	Throughput (Mbps)	PPS	Throughput (Mbps)
64	45,719	20.71	139,853	66.55
128	45,976	41.18	141,582	135.80
256	42,641	79.25	141,350	278.48
512	48,457	176.16	147,748	569.34
1024	43,652	337.67	133,010	1,040.92
1280	41,505	403.45	135,927	1,375.43
1514	38,928	449.51	128,091	1,507.60

이트인 경우 초당 약 130,000개의 패킷을 처리하였고, OVS 포트 미러링보다 약 25Mbps 더 많은 처리량을 보였다. 처리량은 OVS 포트 미러링의 경우와 마찬가지로 패킷 사이즈가 증가함에 따라 증가하였다. 또한 eBPF 기반 vTAP은 약 1.6%의 CPU 사용량을 보이며 OVS 포트 미러링보다 (약 2.3%) 적게 CPU를 사용하는 것을 확인하였다.

2. 서로 다른 VM에 위치하는 Receiver 및 vTAP의 처리 성능 비교

그림 4(b)는 본 절에서 수행한 실험의 테스트베드를 나타낸다. 이전 절의 실험에서와 같이 Sender VM은 패킷을 생성하여 Receiver VM으로 보낸다. 이전 절의 실험과는 달리 Receiver VM과 다른 VM에서 동작하는 vTAP은 Receiver VM에 도달하는 패킷을 복제하고 복제된 패킷을 Monitor VM으로 전달한다. 본 절의 실험은 이전 절의 실험과 같이 Monitor VM에서 처리량, 초당 처리되는 패킷의 개수, CPU 사용량을 측정하였다. 이전 절의 실험에서는 각 VM 마다 2개의 vCPU와 2GB의 메모리를 할당하였지만 하드웨어의 리소스 제한으로 인하여 본 절의 실험에서는 각 VM 마다 1개의 vCPU와 2GB의 메모리를 할당하였다.

그림 6와 표 2은 본 절의 실험에서 eBPF 기반 vTAP과 OVS 포트 미러링을 통해 전달된 패킷의 처리 성능을 측정한 결과를 나타낸다. 실험 결과 OVS 포트 미러링은 리소스 제한으로 인하여 처리 성능에 크게 영향을 받는 것으로 나타났다. 패킷 사이즈가 64 바이트인 경우 초당 약 46,719개의 패킷을 처리하였고, 최대 패킷 사이즈인 경우 초당 처리되는 패킷의 수가 약 40,000개 미만으로 감소하였다 (표 2). 처리량은 패킷 사이즈가 64바이트인 경우 약 20Mbps, 최대 패킷 사이즈인 경우 약 450Mbps를 나타내어 이전 절에서 측정한 값보다 약 50% 감소하였다. 반면 eBPF 기반 vTAP은 이전 절의 실험에서 측정한 값과 거의 비슷하게 패킷을 처리하는 것으로 나타났다. CPU 사용량은 OVS 포트 미러링이 약 13.5%를 사용하여 eBPF 기반

vTAP이 사용하는 CPU 사용량 (약 6.8%)보다 2배가 차이 나는 것을 확인하였다.

이전 절과 본 절의 실험을 통해 제안하는 eBPF 기반 vTAP이 OVS 포트 미러링보다 처리 성능이 우수하다는 것을 확인하였다. eBPF는 JIT 컴파일을 수행하면서 native instruction에 보다 쉽게 매핑될 수 있고, TC 프로그램이 패킷이 커널의 트래픽 제어 계층에 도달할 때 트래픽을 인터셉트하기 때문에, 유저 레벨에서 수행되는 OVS 포트 미러링보다 커널 기반의 eBPF vTAP이 더 나은 패킷 복제 성능을 보임을 알 수 있다.

3. 침입 탐지 시스템 (IDS) 분석에 vTAP을 활용한 실험

마지막 절의 실험에서는 제안된 eBPF 기반 vTAP을 사용하여 원본 패킷을 Receiver VM에서 동작하는 IDS로 전달하고, 복제된 패킷을 Monitor VM에서 동작하는 IDS로 전달한다. IDS는 네트워크의 보안을 위해 사용되는 애플리케이션으로, vTAP과 함께 활용하여 내부 네트워크의 보안 상태를 모니터링하는 것이 가능하다.

본 절의 실험에서 우리는 오픈 소스 IDS Suricata [13]를 사용하였다. Suricata는 고성능의 멀티 스레딩을 지원하고 주기적으로 업데이트된 룰셋(rule sets)을 통해 새로운 종류의 네트워크 공격에 대응한다. 본 실험에서 우리는 Suricata를 IDS 모드로 작동시켜 vTAP을 통해 전달되는 악의적인 공격 트래픽을 탐지하여 경고 보고서 (alerts report)를 생성하게 하였다.

그림 4(c)는 본 절에서 수행한 실험의 테스트베드를 나타낸다. 본 절의 실험은 이전 절의 실험과 마찬가지로 하드웨어 리소스의 제한으로 인해 각 VM에 1 개의 vCPU와 2GB의 메모리를 할당하였다.

우리는 Sender VM이 악의적인 공격 트래픽을 생성하는 과정에서 IDS 및 IPS 테스팅 프레임워크인 Pytbull [14]을 사용하였다. Pytbull은 DoS, shell code 및 다양한 네트워크 공격 트래픽을 생성할 수 있다. 실험은 동일한 네트워크 구성에서 동일한 트래픽으로 전송 속도와 같은 파라미터 값을 변경하면서 여러 번 실험을 진행하기 위해 Pytbull을 통해 생성된 공격 트래픽을 Tcpreplay [15]를 사용하여 저장하고 이를 반복적으로 활용하였다.

그림 7은 서로 다른 전송 속도로 네트워크 공격 트래픽을 보냈을 때 각 IDS가 공격 트래픽을 감지

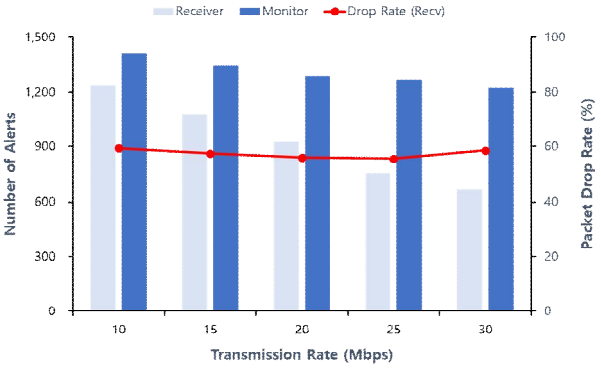

그림 7. Suricata IDS가 탐지하는 비정상적인 트래픽의 개수 비교 (Testbed 3)
Fig. 7. Suricata IDS Alerts Comparison of Testbed 3

표 3. Suricata IDS를 활용한 실험의 측정 결과
Table 3. Suricata IDS Experimental Results of Testbed 3

Transmission Rate (Mbps)	Receiver		Monitor	
	Alerts (#)	Drop Rate (%)	Alerts (#)	Drop Rate (%)
10	1,238	59.31	1,413	0.00
15	1,080	57.42	1,345	0.00
20	924	55.89	1,289	0.74
25	753	55.61	1,263	0.01
30	666	58.50	1,223	0.00

하는 개수를 측정한 결과를 나타낸다. Receiver IDS (Receiver VM에 설치된 IDS)는 10Mbps의 전송 속도에서 평균 1,238개의 경고 보고서를 생성하였다. 전송 속도가 증가함에 따라 Receiver IDS에 의해 생성되는 경고 보고서의 수는 30Mbps에서 666개로 감소하였다. 반면 Monitor IDS (Monitor VM에 설치된 IDS)는 Receiver IDS보다 더 많은 경고 보고서를 생성하였다. 전송 속도가 증가함에 따라 10Mbps의 전송 속도에서 1,413개의 경고 보고서를 생성하고 30Mbps의 속도에서 1,223개의 경고 보고서를 생성했다. 전송 속도에 따른 경고 보고서 수의 차이는 Monitor IDS가 Receiver IDS보다 작은 것으로 나타났다.

본 실험에서 각 IDS가 생성하는 경고 보고서의 수와 함께 각 IDS에서의 패킷 drop rate도 측정하였다 (표 3). Receiver IDS에 도착한 패킷의 drop rate는 10Mbps의 전송 속도에서 평균 59.31%으로 나타났으며, 30Mbps의 전송 속도에서 drop rate는 58.50%로 전송 속도가 점차 증가하였음에도 큰 차이가 없었다. 반면 Monitor IDS 패킷의 drop rate는 전송 속도에 관계없이 0%로 나타났다.

Receiver IDS와 Monitor IDS 간의 drop rate 비교를 통해 Receiver VM은 IDS를 통한 패킷 분석 시 Monitor VM보다 많은 패킷 손실을 겪는 것으로 나타났다. 실험에서 CPU 사용률, 패킷의 수와 같은 여러 리소스에 대한 추가 분석을 통해 원본 패킷을 받는 Receiver IDS가 복제 패킷을 받는 Monitor IDS보다 높은 패킷 수신량을 보였고, 따라서 더 많은 패킷을 분석하기 위한 Suricata 쓰레드와 패킷 인터럽트 핸들러의 리소스 경쟁이 많은 패킷 손실을 발생시키는 것으로 나타났다. 이는 (1) VM에서 1개의 vCPU를 사용하면 CPU 부족으로 인해 병목 현상이 발생하고, (2) eBPF 기반 vTAP에서 Monitor VM에 보내는 패킷은 패킷 복제에 대한 오버헤드가 발생하여 Receiver VM이 받는 원본 패킷보다 낮은 패킷 처리량을 보이기 때문이다. 본 실험 결과에서 각 IDS 분석 결과의 일관성을 보장하기 위하여 VM에 더 많은 리소스를 할당하고, 미터링 (metering)과 같은 방법을 사용하여 본 논문에서 제안하는 eBPF 기반 vTAP의 동작 방식 개선을 고려할 수 있다.

Ⅴ. 결 론

클라우드 컴퓨팅 환경에서 VM은 vNIC 및 가상 스위치를 통해 연결된다. 하지만 기존 하드웨어 TAP은 VM 간 트래픽 모니터링에 사용할 수 없으므로 패킷 복제 기능을 소프트웨어로 구현한 vTAP이 필요하다. 가상 스위치 기반 vTAP은 동일한 호스트 서버에 위치한 다른 VM과 리소스를 공유하기 때문에 성능이 저하된다. 이 문제를 극복하기 위하여 본 논문에서는 오픈 소스이자 고속 패킷 처리 기술인 eBPF를 사용하여 VM에서 동작하는 vTAP을 설계하고 구현하였다. 실험 결과에 따르면 제안하는 eBPF 기반 vTAP은 PPS, 처리량 및 CPU 사용량 측면에서 OVS 포트 미러링보다 우수한 성능을 보여주었다.

본 논문에서 수행한 실험은 리소스가 제한된 환경에서 수행되었기 때문에 향후 연구 방향으로는 다양한 시나리오를 통해 테스트베드를 확장하고 제안된 eBPF 기반 vTAP의 동작을 미터링 및 큐 스케줄링과 같은 방법을 통해 개선한다. 이후 커널을 bypass하는 다른 고속 패킷 처리 기술인 XDP (eXpress DataPath)를 활용하여 vTAP를 확장 구현할 예정이다 [16].

References

[1] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “Open-stack: Toward an open-source solution for cloud computing,” Int. J. Comp. Appl. (0975-8887), vol. 55(03), October 2012.

[2] IO Visor Project, “extended Berkeley Packet Filter.” [Online]. Available at https://www.iovisor.org/technology/ebpf.

[3] Ben Pfaff et al., “The design and implementation of Open vSwitch,” In 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’15) , 2015, pp. 117-130.

[4] VeryX, “Veryx vTAP datasheet,” Tech. Report, 2017. [Online]. Available at http://www.veryxtech.com/wp-content/uploads/2017/11/Datasheet-Veryx-vTAP_20171115.pdf.

[5] Gigamon, “GigaVUE-VM datasheet,” Tech. Report, 2016. [Online]. Available at https://www.gigamon.com/content/dam/resource-library/english/data-sheet/ds-gigavue-vm-virtual-machine.pdf.

[6] IXIA, “Ixia Panthom vTAP with tapflow filtering,” Tech. Report, 2016. [Online]. Available at https://www.viavisolutions.com/pt-br/literature/ixia-phantom-vtap-tapflow-filtering-data-sheet-en.pdf.

[7] J. Rasley et al., “Planck: Millisecond-scale monitoring and control for commodity networks,” In ACM Conference on SIGCOMM, 2014, pp. 407-418.

[8] G. Liu and T. Wood, “Cloud-scale application performance monitoring with SDN and NFV,” In 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, 2015, pp. 440-445.

[9] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, “Machine learning in software defined networks: Data collection and traffic classification,” In 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1-5.

[10] A. Abubakar and B. Pranggono, “Machine learning based intrusion detection system for software defined networks,” In 2017 Seventh Inter- national Conference on Emerging Security Technologies (EST), Canter- bury, 2017, pp. 138-143.

[11] S. Jeong, D. Lee, J. Li, and J. W. Hong, “OpenFlow-based virtual TAP using open vSwitch and DPDK,” In 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, 2018, pp. 1-9.

[12] Robert Olsson, “Pktgen the linux packet generator,” In

Proceedings of the Linux Symposium, Ottawa, Canada, 2005, p. 11-24.
[13] Suricata, "Open source IDS/IPS/NSM engine." [Online]. Available at https://suricata-ids.org/.
[14] pytbull, "IDS/IPS testing framework." [Online]. Available at http://pytbull.sourceforge.net/?page=documentation.
[15] Tcpreplay, "Pcap editing and replaying utilities." [Online]. Available at https://tcpreplay.appneta.com/wiki/overview.html.
[16] IO Visor Project, "eXpress data path." [Online]. Available at https://www.iovisor.org/technology/xdp.

홍 지 범 (Jibum Hong)

2017 한양대학교 ERICA 컴퓨터공학과 학사
2017 ~ 현재 포항공과대학교 컴퓨터공학과 통합과정
<관심분야> SDN, 네트워크 관리

정 세 연 (Seyeon Jeong)

2015 경북대학교 컴퓨터학부 학사
2018 포항공과대학교 컴퓨터공학과 석사
2018 ~ 현재 포항공과대학교 컴퓨터공학과 박사과정
<관심분야> SDN, OpenFlow, 네트워크 관리

유 재 형 (Jae-Hyung Yoo)

1983 연세대학교 전자공학과 학사
1985 연세대학교 전자공학과 석사
1999 연세대학교 컴퓨터공학과 박사
1986 ~ 2012 KT 네트워크 연구소
2012 ~ 2013 KAIST 전기 및 전자공학과 연구부 교수
2013 ~ 2016 포항공과대학교 컴퓨터공학과 연구부 교수
2016 ~ 2018 정보통신기술진흥센터 네트워크 CP
2018 ~ 현재 포항공과대학교 컴퓨터공학과 연구부 교수
<관심분야> 네트워크 관리 및 보안, SDN, OpenFlow

홍 원 기 (James Won-Ki Hong)

1995 ~ 현재 포항공과대학교 컴퓨터공학과 교수
2007 ~ 2011 포항공과대학교 정보통신대학원장
2007 ~ 2010 포항공과대학교 정보통신연구소 연구소장
2008 ~ 2010 포항공과대학교 컴퓨터공학과 주임교수
2008 ~ 2012 포항공과대학교 정보전자융합공학부장
2012 ~ 2014 KT 종합기술원 원장
<관심분야> 네트워크 트래픽 모니터링, 네트워크 및 시스템 관리, SDN/NFV, IoT, 무크기반 교육