Journal of KIIT. Vol. 18, No. 2, pp. 137-145, Feb. 28, 2020. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2020.18.2.137

PF_Ring을 이용한 대용량 트래픽 처리 시스템

이달원*, 고대식**, 김동환***, 최상용****

High-Capacity Traffic Processing System Using PF_Ring

Dalwon Lee*, Dae-Sik Ko**, Donghwan Kim***, and Sang-Yong Choi****

본 연구는 중소기업청 산학연협력기술개발사업 (S2668439)의 지원으로 수행되었음.

요 약

사물인터넷, 빅데이터 기술의 발전은 다양한 장비들이 상호 연결되어 작용하는 것을 가능하게 하였고, 이로 인하여 시스템 사이의 정보전달을 위한 트래픽 양 또한 대용량화 되어 가고 있다. 이러한 상황에서 시스템과 네트워크의 운영상태, 장애상태, 보안위협 등을 효과적으로 관제하기 위해서는 대용량 트래픽의 실시간 처리가 매우 중요하다. 본 논문에서는 5Gbps의 대용량 트래픽에서 의미 있는 데이터를 실시간으로 추출할 수 있는 PF_Ring을 이용한 시스템을 제안하였다. 제안된 시스템은 리눅스 시스템 커널에 PF_Ring을 설치하고, Collector Handler를 이용하여 PF_Ring을 주기적으로 실행하여 트래픽을 수집하고 Packet_preprocessor가 읽어 병렬처리 방식으로 동작하도록 설계하였다. 실험결과, 수집 패킷은 5Gbps를 처리하는데 손실률은 0%이었고 전처리 시간은 약 47초이기 때문에 제안된 시스템을 통하여 다음 전처리 주기인 1분 이전에 전체적인 트래픽을 완전하게 처리하고 있음이 확인되었다.

Abstract

The advances in IoT (Internet of Things) and Big Data technologies have enabled various devices to work together, also causing the amount of traffic between systems to increase significantly. In these situations, real-time processing of large traffic is critical in order to effectively control the operational status, failure status, and security threat of the system and network. In this paper, a system using PF_Ring is proposed which is capable of extracting meaningful data from high-capacity traffic of 5Gbps in real-time. The proposed system installed PF_Ring in the Linux system kernel, periodically executed PF_Ring using the Collector Handler to collect traffic and read the data through Packet_preprocessor to operate in parallel. The experiment results have confirmed that the proposed system is able to fully handle the entire traffic within one minute, which is the next pre-processing cycle, with the loss rate of 0% in processing the 5Gbps collected packets and pre-processing time of approximately 47 seconds.

Keywords
network, packet collector, PF Ring, traffic, high-capacity traffic

* 마인드서프 주식회사 인공지능연구소 소장
- ORCID: https://orcid.org/0000-0003-0612-8809
** 목원대학교 전자공학과 교수
- ORCID: https://orcid.org/0000-0002-6232-476X
*** 마인드서프 주식회사 인공지능연구소 연구원
- ORCID: https://orcid.org/0000-0002-9668-6531
**** 영남이공대학교 사이버보안과 조교수(교신저자)
- ORCID: http://orcid.org/0000-0001-5152-3897

· Received: Nov. 01, 2019, Revised: Nov. 28, 2019, Accepted: Dec. 01, 2019
· Corresponding Author: Sang-Yong Choi
Dept. of Cyber Security, Yeungnam University College, Daegu, Republic of Korea
Tel.: +82-53-650-9713, Email: spikechoi@ync.ac.kr

Ⅰ. 서 론

ICT 기술의 발전은 방대한 IoT(Internet of Things, 사물인터넷) 장치를 연결할 수 있게 하였다. 하지만 이러한 연결로 인해 분석해야 할 데이터의 크기가 기하급수적으로 증가하고, 데이터에 대한 실시간 처리는 더욱 중요해 지고 있다. 이는 네트워크 트래픽 측면에서는 더욱 민감해진다. 수 없이 연결된 다양한 디바이스 간 통신과 데이터 전송을 위해 전송되는 네트워크 트래픽의 양이 더욱 증가하고 있으므로 손실 없이 트래픽을 처리하고 트래픽 내의 이상 징후를 찾기 위한 연구가 계속되고 있다[1][2].

분석 기술 중 하나로, 트래픽에 포함된 사이버 위협을 탐지하기 위한 IDS(Intrusion Detection System, 침입탐지시스템) 기술이 등장하였으며, IDS에서 무엇보다도 중요한 것은 손실 없이 실시간으로 트래픽을 처리하는 것이다. 특히 IDS에서는 단순한 트래픽의 실시간 처리보다 패킷 내에 포함된 의미 있는 정보를 얼마나 잘 추출할 수 있는지가 중요하다고 할 수 있다[3].

본 논문에서는 IDS를 포함한 다양한 시스템에서 활용할 수 있도록 네트워크 트래픽을 빠르게 전처리할 수 있는 기술을 연구하였다. 이를 위해 기존 네트워크 트래픽 수집 및 처리 기술, 침입차단시스템 등 보안시스템에서 필요한 트래픽 내의 정보 등을 분석하였고, 기존 Libpcap의 성능을 향상시킨 PF_Ring을 이용하여 손실 없이 5Gbps의 트래픽을 실시간으로 처리할 수 있는 시스템을 제안하였다.

Ⅱ. 관련 연구

2.1 패킷 기반 수집기술

2.1.1 Libpcap, Winpcap

패킷은 네트워크에서 데이터를 전송하는 기본 단위이다. 패킷 교환 네트워크에서는 정보를 패킷으로 나누어 인코딩한 후 목적지로 전송한다. 전송에 사용되는 패킷은 프로토콜의 종류에 따라 다른 형태를 가지고 있으며, 일반적으로 헤더와 페이로드로 구분되어 있다. 패킷헤더는 패킷의 출발지와 목적지를 표시하며 네트워크 내에서 패킷이 전송되는데 사용하는 정보를 가지고 있다. 페이로드에는 실제 전달한 정보가 포함된다. 따라서 일반적으로 트래픽을 수집한다는 것은 패킷을 수집한다는 것과 유사한 의미로 사용된다. 패킷 캡처는 네트워크 관리를 위한 여러 가지 정보를 수집하는 전통적인 방법이며, 일반적으로 유닉스, 리눅스 계열 시스템에서 이 패킷 캡처의 기본이 되는 라이브러리가 Libpcap이다. 윈도우 계열 시스템에서는 Winpcap 라이브러리가 같은 역할을 한다[4]-[6].

패킷을 수집할 때 사용하는 라이브러리인 Libpcap 함수는 패킷 흐름의 모든 내용을 수집할 수 있는 기능을 제공한다. 하지만 대용량의 네트워크 트래픽을 추리하기 위해 표본추출의 방법을 활용한다. 이는 표본추출의 조건에 따라 대용량의 트래픽을 지원하지만, 표본추출 방법 자체가 패킷 일부를 수집하여 전체로 확장하는 개념이기 때문에 전체 트래픽을 전수검사 해야 하는 환경에서는 적합하지 않을 수 있다.

2.1.2 PF_Ring

PF_Ring은 네트워크 패킷을 고속으로 수집하고 처리하기 위한 기술로 개발되었으며, Linux 커널에서 동작한다. 사용하기 위해서 커널 패치가 필요하지 않으며, 10Gbit 이상의 네트워크 트래픽을 수집할 수 있는 것으로 알려져 있다. 또한, Libpcap을 지원하며, 콘텐츠 필터링과 구문분석을 지원하는 기술이다[7]. PF_Ring의 아키텍처는 그림 1과 같이 커널 내에서 네트워크 어댑터로부터 직접 소켓을 이용하여 패킷을 수집하는 구조이다[8].

2.2 플로우 기반 트래픽 수집기술

네트워크 정보를 수집하는 다른 기술로 플로우 기반 수집기술이 있다. 일반적으로 플로우는 일정 기간 특정 라우터를 통과하는 같은 특성을 가진 패킷의 집합이다.

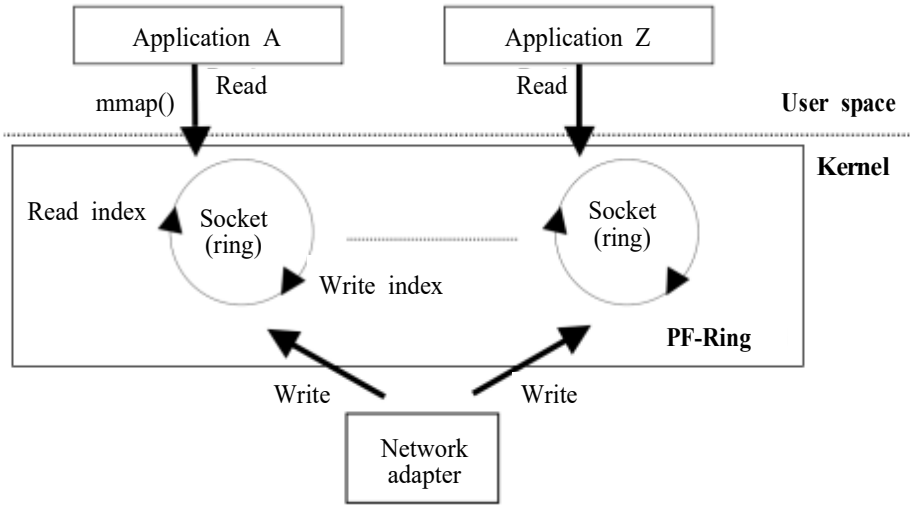


그림 1. PF_Ring 아키텍처
Fig. 1. Architecture of PF_Ring

따라서 해당 라우터를 통과하지 않는 트래픽에 대해서는 수집할 수 없으며, 모든 트래픽을 완전하게 수집하기 위해서는 네트워크 내의 모든 라우터로부터 트래픽을 수집해야 한다. 플로우 기반 수집 구조에서 가장 일반적으로 수집하는 정보는 출발지 주소와 포트, 목적지 주소와 포트, 프로토콜과 같은 5-Tuple 정보를 수집한다. 대표적인 플로우 기반 수집기술은 Netflow, SFlow와 같은 기술로 구현되어 있다[9][10]. 하지만 플로우 기반 수집기술은 전체 트래픽의 내용 중 해도 정보를 기반으로 수집하기 때문에, 패킷수집기술로 활용하기에는 적합하지 않다.

2.3 로그 기반 트래픽 수집기술

로그 기반 수집기술은 이벤트 로그와 메시지 로그 등과 같은 로그 형태로 기록을 남기고 이를 수집하는 기술이다. 로그는 일반적으로 이벤트 로그와 메시지 로그로 구분된다. 이벤트 로그는 사용자의 추적, 이벤트 상태를 기록하는 로그이다. 일반적으로 이벤트 로그는 서비스를 가동하면 파일이 생성된다. 이벤트 로그는 일반적으로 자료를 수집하는데 사용될 수 있다. 이벤트 로그의 종류로는 운영체제 로그, 웹 로그, 각종 장비의 로그 등으로 구성되며 표준 형식이 존재하지 않고 장비별로 형태가 상이하다. 메시지 로그는 채팅, 인스턴스 메시지 등을 포함한 사용자가 발생시키는 메시지 형태의 로그이다. 이는 대부분 프라이버시 이슈로 인해 암호화한다. 수집기가 탐색기에서 실시간 로그를 조사하거나 저장소에서 파일을 수집한 경우 필터 모듈은 로그의 내용에 대한 구문분석을 통해 필수적으로 필요한 정보를 추출한다. 로그 파서는 여러 가지 다양한 로그 파일에 대한 간단한 로그 선별 도구가 될 수 있다[11].

2.4 수집기술 분석

앞서 살펴본 바와 같이 수집기술은 패킷 기반, 플로우 기반, 로그 기반 수집기술로 분류할 수 있다. 하지만 본 논문에서는 패킷 내에 포함된 위협정보를 추출하기 위해 단순 헤더 정보가 아니라 해더 정보를 포함한 트래픽의 출발지와 목적지 간의 관계분석 정보를 포함하기 때문에 플로우 기반, 로그 기반 수집기술은 적절하지 않다. 플로우 기반 수집기술은 전체 트래픽을 수집하지 않고, 로그 기반 수집기술은 전체 로그는 볼 수 있지만, 전체 로그에 트래픽 정보를 기록하는 것 자체가 힘들기 때문이다.

따라서 본 논문에서 활용 가능한 수집기술은 패킷 기반 수집기술이다. 패킷 기반 대표적인 수집기술의 두 가지(Libpcap, PF_Ring)에 관해 관련 연구를 분석한 결과 수집 범위, 수집성능, 구문분석성능, 장점, 단점 등에 대해 표 1과 같이 분석할 수 있었다.

표 1. 트래픽 수집기술 분석
Table 1. Traffic collection technology analysis

Item	Libpcap	PF_Ring
Collection range	5-tuple, payload	5-tuple, payload
Collection performance	Slowness	Fast
Parsing performance	Fast	Fast
Advantage	Available for both Windows and Linux	10Gbps traffic collectable
Weakness	Drop in kernel for bulk traffic	Act on Linux kernels only

III. PF_Ring을 이용한 대용량 트래픽 처리 시스템

3.1 기존의 대용량 트래픽 처리 시스템

현재 사용되고 있는 대용량 트래픽 처리가 가능한 시스템은 패킷분석기의 형태와 침입탐지(방지) 시스템의 형태가 있다. 패킷분석기는 네트워크 트래픽을 단순히 수집하여 5-tuple을 기준으로 파싱 후 요약정보를 저장하거나 RAW 데이터를 저장하는 방식이다. 침입탐지(방지)시스템은 트래픽을 수집하여 위협정보를 메모리상에서 분석하여 위협정보를 탐지하고 탐지된 로그를 저장하는 형태이다. 이와 같은 시스템은 대용량 트래픽 처리를 위해 네트워크 전용 프로세스 등 전용 하드웨어 네트워크 장치를 사용한다. 하지만 이와 같은 하드웨어 환경은 대용량 데이터 수집과 패킷 내에 포함된 페이로드 정보를 분석하여 위협정보를 탐지하는 것은 보장하지만 본 연구에서 제안하는 바와 같이 전체 트래픽에 대해 수집된 트래픽의 관계정보를 분석하여 추출하는 등 복합적인 작업에는 한계가 있다. 또한, 공개 IDS인 Suricata에 PF_Ring을 적용하여 트래픽 처리 능력을 향상시키고 있으나 이 또한 IDS의 특성상 수집된 패킷 정보를 메모리에서 분석하여 전체 데이터 중 위협정보만을 추출하고 있다. 따라서 전체 패킷 정보를 분석하여 실시간으로 전체 트래픽을 저장하고 처리하는 데는 한계가 있다[12]-[14].

3.2 PF_Ring을 이용한 트래픽 처리 시스템

PF_Ring 모듈의 경우 이론적으로는 10Gbps의 트래픽을 수집할 수 있으나, 본 논문에서 제안하는 시스템은 트래픽 수집뿐만 아니라 수집된 트래픽을 후위 시스템에서 활용 가능한 형태로 파싱할 수 있어야 한다. 또한, 정보를 추출하고 정보 간의 관계를 분석하는 전체 과정이 실시간으로 이루어져야 한다. 이러한 기준에 따라 제안하는 시스템은 5Gbps 이상의 트래픽을 처리할 수 있도록 설계하였다.

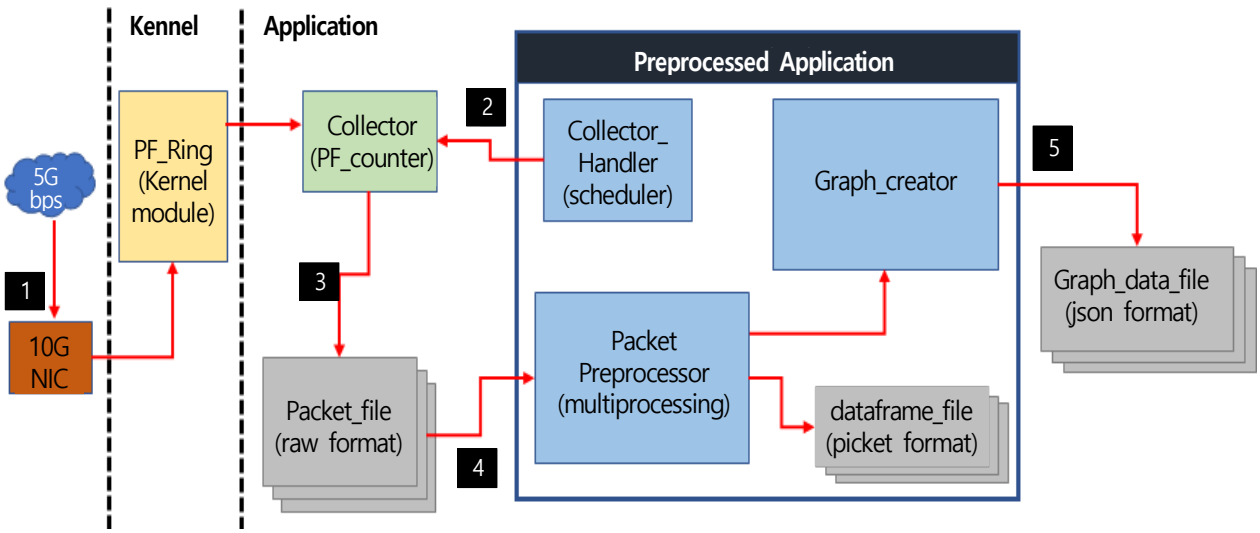


그림 2. 제안된 트래픽 처리 시스템 아키텍처
Fig. 2. Architecture of proposed traffic processing system

본 논문에서 제안하는 트래픽 처리 시스템은 리눅스 시스템 커널에 PF_Ring을 설치하고, Collector Handler를 이용하여 PF_Ring을 주기적으로 실행하여 수집을 시작한다. 수집기가 수집하여 저장한 패킷 파일을 Packet_preprocessor가 읽어 병렬처리 방식으로 패킷을 전처리한다. 처리를 완료한 데이터는 Pickle 포맷으로 압축하여 백업하고 같은 데이터를 후위 모듈인 시각화 모듈이 활용할 수 있도록 Graph_Creator로 전송한다. Graph_Creator는 입력된 데이터를 JSON 형태의 그래프 데이터로 생성한다. 본 논문에서 제안한 시스템은 이와 같은 전체의 과정이 다음 처리 주기 시간이 도달하기 이전에 실시간으로 이루어지며, 전체 패킷에서 사용자가 원하는 데이터를 실시간으로 처리하여 생성할 수 있다. 제안하는 시스템의 전체 아키텍처는 그림 2와 같다.

본 논문에서 제안한 트래픽 처리 시스템의 트래픽 처리 과정은 수집부터 그래프 데이터 파일 생성까지 총 4단계로 이루어진다. 이는 그림 2에 표현된 (2)~(5)까지의 과정이다. 그림 2에 표현된 (1)의 과정은 후위 과정을 설명하기 위해 네트워크 장비 등으로부터 트래픽이 유입되는 과정이므로 설명은 생략한다. (2)~(5)까지의 과정의 처리단계는 다음과 같다.

(2) Collector_Handler : 커널에 연결된 PF_Ring을 주기적으로 실행하여 패킷을 수집을 시작하거나 PF_Ring의 상태를 감시하여 재기동하는 역할을 하는 Collector를 관리한다.

(3) Packet_File : PF_Ring이 정상적으로 구동되고 패킷 수집이 시작되면 Collector는 PF_Ring으로부터 전달된 패킷 데이터를 파일 시스템에 파일(Packet_File)로 적재한다. 적재는 매번 collector가 시작할 때 시작되며 collector가 재시작하면 새로운 파일로 적재한다.

(4) Packet_PreProcessor : 전처리기는 기본적으로 멀티 프로세싱이 가능하도록 설계하였다. 즉, 패킷 파일 적재가 시작되면 패킷 전처리기는 적재된 파일을 읽기 시작한다. 이때, 파일을 읽는 도중 새로운 파일이 생성되면 새로운 프로세스를 생성하여 기존의 작업에 영향을 주지 않고 다시 읽도록 한다.

(5) Graph_Creator : 전처리기로부터 파싱이 완료된 데이터를 입력값으로 하여 후위 시스템에서 사용 가능한 데이터를 JSON 포맷의 파일로 생성한다.

Ⅳ. 실험 및 평가

4.1 실험시스템 구성

본 연구에서 처리성능 실험을 위한 실험시스템은 그림 3과 같다. 먼저 패킷 생성기와 수집/처리 시스템의 네트워크 인터페이스는 10Gbps 인터페이스를 사용하였다. 패킷 생성기에는 5개의 터미널을 이용하여 각 터미널당 1Gbps씩 총 5Gbps(TCP 4Gbps, UDP 1Gbps)의 트래픽을 발생시켰으며, 이를 스위치를 이용하여 수집처리시스템으로 전송하였다.

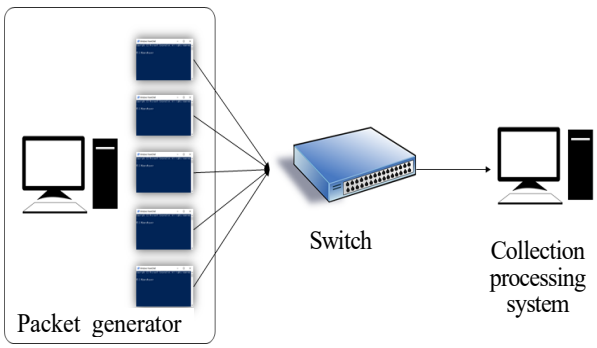


그림 3. 실험시스템의 구성
Fig. 3. Configuration of the experimental system

표 2. 실험환경 구성
Table 2. Test environment

	Category	Spec
S/W	Traffic generator	Ubuntu 18.04 generator S/W : ipert 2.0.10
	Collector/ processing system	OS:Ubuntu 18.04 WAS: Apache Tomcat 8.0.53 JAVA: OpenJDK 1.8.0_131 Etc: Python 3.6.7, pf_ring 7.5.0, iperf 2.0.10
H/W	Traffic generator	CPU: Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz RAM: 8GB SSD: 200GB NIC: 10Gbps
	Collector/ processing system	CPU: Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz, RAM: 64GB SSD: 250GB NIC: 10Gbps

표 2는 실험시스템의 실험조건이다. 실험을 위한 시스템은 총 2대이며, 운영체제는 Ubuntu 18.04버전을 사용하였다. 또한, 실험을 위한 트래픽 생성은 트래픽 테스트 도구인 ipert2 버전을 활용하였다.

4.2 실험결과 분석

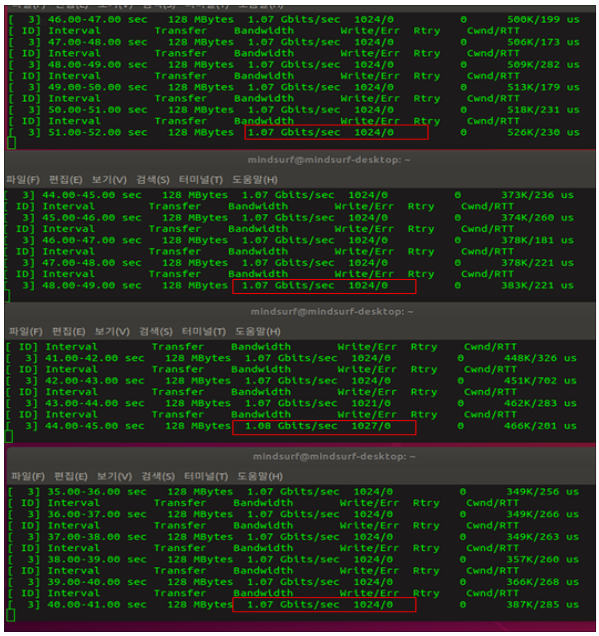
본 연구에서 실험의 범위는 처리성능을 위주로 진행하였다. 먼저 PF_Ring 커널 모듈을 사용하였을 때와 사용하지 않을 때 패킷처리 성능에 대해 간단한 테스트를 통해 PF_Ring을 사용했을 때 패킷 Drop이 발생하지 않는 것을 확인하였다. PF_Ring 커널 모듈을 사용했을 때와 하지 않았을 때의 실험결과는 표 3과 같다. 표에서 보이는 것과 같이 PF_Ring을 사용하지 않았을 때는 커널 모듈에서 패킷 손실이 발생하며 PF_Ring을 사용했을 때에는 패킷 손실이 발생하지 않는다.

표 3. PF_Ring 커널 모듈 사용 전후 실험결과
Table 3. Test result before and after using PF_Ring kernel module

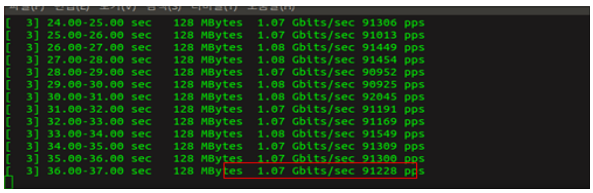
	·Non PF-Ring	PF_Ring
1st time	· 57,762 packets captured · 100,857 packets received by filter · 43,071 packets dropped by kernel	· 101,305 packets captured · 101,305 packets received by filter · 0 packets dropped by kernel
2nd time	· 8,759,813 packets captured · 18,396,160 packets received by filter · 876,390 packets dropped by kernel	· 11,039,123 packets captured · 11,039,123 packets received by filter · 0 packets dropped by kernel

다음으로 PF_Ring을 적용하여 설계한 본 논문에서 제안한 시스템에 대한 실험을 진행하였다. 트래픽 생성기에의 터미널 화면을 보면 그림 4와 같이 트래픽이 터미널당 1.07Gbps씩 5Gbps(TCP 4Gbps, UDP 1Gbps)가 정상적으로 생성되고 있다.

그림 5는 제안된 수집 처리 시스템의 성능, 즉 수집속도가 5Gbps 이상인지 확인한 결과이다. 그림 5에서 5Gbps 이상의 수집속도를 보이며, 수집된 데이터가 파일로 정상적으로 기록되고 있음을 확인할 수 있다. 이후 전처리기에서 처리 상태를 확인하였다.



(a) TCP 트래픽 생성
(a) TCP traffic generate



(b) UDP 트래픽 생성
(b) UDP traffic generate

그림 4. TCP, UDP 트래픽 생성
Fig. 4. TCP, UDP traffic generate

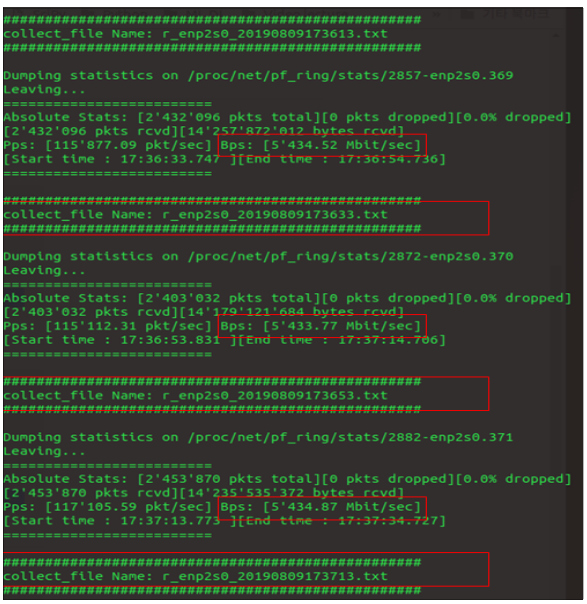


그림 5. 수집시스템 수집속도 확인
Fig. 5. Check collection system collection speed

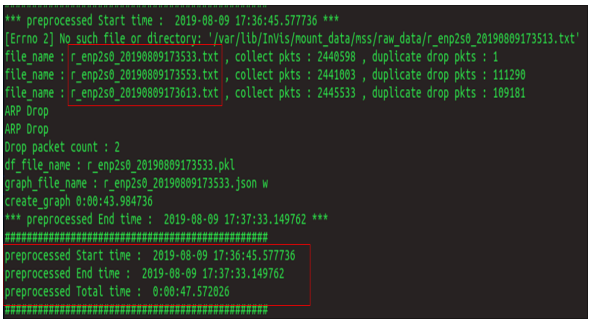


그림 6. 전처리기 처리시간 확인
Fig. 6. Check Preprocessor processing time

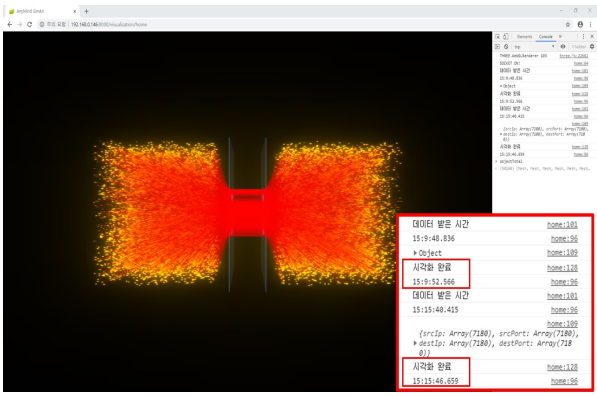


그림 7. 전처리된 데이터 시각화
Fig. 7. Preprocessed data visualization

전처리기의 경우 1분 단위로 수집된 데이터를 처리하도록 설계되어 있다. 즉, 1분간 수집된 데이터를 한 번에 처리하고 이후 1분이 지나면 다시 새로운 데이터를 처리하는 구조이다. 따라서 하나의 파일을 1분 이내에 처리할 수 있다면 지연 없이 모든 트래픽을 처리할 수 있다. 그림 6에서 보이는 것과 같이 전처리기는 약 47초에 데이터 처리를 완료하고 있다.

결론적으로 제안된 시스템을 사용하여 5Gbps에 대한 트래픽 처리를 테스트한 결과, 그림 5와 같이 수집 패킷은 약 120kpps / 5Gbps를 처리하는데 손실률은 0%를 보이며, 전처리 시간은 약 47초로 이는 다음 전처리 주기인 1분 이전에 전체적인 트래픽을 완전하게 처리하고 있음이 확인되었다.

또한, 전처리가 완료된 데이터는 그림 7과 같이 노드와 릴레이션이 생성되어 화면에 표출할 수 있었다. 실험에 사용한 노드는 패킷의 출발지 주소와 목적지 주소이며, 릴레이션은 프로토콜에 따른 연결 구조이다.

V. 결론 및 향후 과제

본 논문에서는 대용량 네트워크의 상태를 실시간으로 감시하기 위한 기초가 되는 기술인 트래픽 수집 및 전처리 시스템을 설계하고 구현하였다. 대용량 트래픽 수집을 위해 PF_Ring을 활용하여 커널 수준에서 5Gbps 이상의 트래픽을 실시간으로 수집하고, 수집된 데이터를 후위 시스템에서 활용이 가능한 형태로 전처리까지 하는 전체의 과정이 실시간으로 손실 없이 실시간으로 이루어지도록 구현하였다. 실험결과에서, 수집 패킷은 약 120kpps / 5Gbps를 처리하는데 손실률은 0%이었고 전처리 시간은 약 47초이기 때문에 제안된 시스템을 통하여 다음 전처리 주기인 1분 이전에 전체적인 트래픽을 완전하게 처리하고 있음이 확인되었다.

본 연구의 결과는 대용량 네트워크의 정보를 수집하여 시각화, 침입 탐지, 성능관리 등 다양한 용도로 활용할 수 있는 기반 기술이 될 수 있을 것으로 기대한다. 향후, 본 연구의 결과를 활용하여 실제 침입 탐지, 성능관리 등의 활용이 가능한 형태로 표현하는 방법과 기술에 관한 연구를 지속해서 수행할 예정이다.

References

[1] M. A. Qadeer, A. Iqbal, M. Zahid, and M. R. Siddiqui, "Network traffic analysis and intrusion detection using packet sniffer", Second International Conference on Communication Software and Networks, Singapore, pp. 313-317, Feb. 2010.

[2] D. Ficara, S. Giordano, F. Oppedisano, G. Procissi and F. Vitucci, "A cooperative PC/Network-Processor architecture for multi gigabit traffic analysis", 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks, Venice, Italy, pp. 123-128, Feb. 2008.

[3] Intrusion Detection Systems: A Modern Investigation - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/DS-within-an-organizational-network-Goals-and-Functions-of-IDS-Many-studies-have-shown_fig2_269464724 [accessed: Jul 15, 2019]

[4] C. Morariu and B. Stiller, "DiCAP: distributed packet capturing architecture for high-speed network links", 33rd IEEE Conference on Local Computer Networks (LCN), Montreal, Que, Canada, pp. 168-175, Oct. 2008.

[5] X. An and X. Liu, "Packet capture and protocol analysis based on Winpcap", 2006 International Conference on Robots & Intelligent System (ICRIS), Zhangjiajie, China, pp. 272-275, Aug. 2016.

[6] G. Antichi, S. Giordano, D. J. Miller, and A. W. Moore, "Enabling open-source high speed network monitoring on NetFPGA", 2012 IEEE Network Operations and Management Symposium, Maui, HI, USA, pp. 1029-1035, Apr. 2012.

[7] ntop, https://www.ntop.org/products/packet-capture/pf_ring [accessed: Sep 14, 2019]

[8] Deri, Luca. "Improving passive packet capture: Beyond device polling", Proceedings of International System Administration and Network Engineering, Amsterdam, Netherlands, 12pages, Sep. 2004.

[9] R. Hofstede, V. Bartoš, A. Sperotto, and A. Pras, "Towards real-time intrusion detection for NetFlow and IPFIX", Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), Zurich, Switzerland, pp. 227-234, Oct. 2013.

[10] L. Elsen, F. Kohn, C. Decker, and R. Wattenhofer, "goProbe: a scalable distributed network monitoring solution", 2015 IEEE International Conference on Peer-to-peer Computing (P2P), Boston, MA, USA, pp. 1-10, Sep. 2015.

[11] D. Yan, R. Feng, J. Huang, and F. Yang, "Host scurity event track for complex network environments based on the analysis of log", 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, Hangzhou, China, pp. 807-811, Nov. 2012.

[12] Intrusion Detection: A Survey - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/4-Basic-architecture-of-intrusion-detection-system-IDS_fig2_226650646 [accessed: Sep 30, 2019]

[13] How to accelerate Suricata, Bro, Snort with PF_RING FT, Available from: https://www.ntop.org/pf_ring/how-to-accelerate-suricata-bro-snort-with-pf_ring-ft/ [accessed: Sep 30, 2019]

[14] Michael Kallitsis, Stilian A. Stoev, Shrijita Bhattacharya, and George Michailidis, "AMON: An open source architecture for online monitoring, statistical analysis, and forensics of multi-gigabit streams", IEEE Journal on Selected Areas in Communications, Vol. 34, No. 6, pp. 1834-1848, Jun. 2016.

저자소개

이 달 원 (Dal-Won Lee)

2005년 2월 : 배재대학교 대학원 컴퓨터공학과(공학박사)
2018년 2월 ~ 현재 : 마인드서프 주식회사 인공지능연구소 소장
관심분야 : 네트워크보안, 인공지능

고 대 식 (Dae-Sik Ko)

1982년 2월 : 경희대학교 전자공학과 졸업(공학사)
1991년 2월 : 경희대학교 전자공학과 졸업(공학박사)
1995년 : UCSB Post-Doc
2011년 ~ 2012년 : 한국정보기술학회 회장
1989년 ~ 현재 : 목원대학교 전자공학과 교수
관심분야 : IoT, 융합 IT, 클라우드 컴퓨팅

김 동 환 (Dong-Hwan Kim)

2004년 2월 : 한국폴리텍2대학 컴퓨터정보과(전문학사)
2017년 4월 ~ 현재 : 마인드서프 주식회사 인공지능연구소 연구원
관심분야 : 네트워크, 소프트웨어 개발

최 상 용 (Sang-Yong Choi)

2000년 2월 : 한남대학교 수학과(이학사)
2003년 2월 : 한남대학교 컴퓨터공학과(공학석사)
2014년 2월 : 전남대학교 정보보안협동과정(이학박사)
2019년 3월 ~ 현재 : 영남이공대학교 사이버보안과 조교수
관심분야 : 네트워크 보안, 웹보안, 악성코드