Journal of KIIT. Vol. 17, No. 8, pp. 59-66, Aug. 31, 2019. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2019.17.8.59

Suricata를 이용한 대용량 네트워크 트래픽 수집성능분석

최상용*, 천은영**, 고대식***

Analysis of the Network Traffic Collection Performance Using Suricata

Sang-Yong Choi*, Eun-Young Cheon**, and Dae-Sik Ko***

본 연구는 중소기업청 산학연협력기술개발사업 (S2668439)의 지원으로 수행되었음

요 약

최근 ICT 기술의 발전으로 네트워크와 네트워크에 연결된 시스템이 복잡해지고 있으며, 이에 대한 효율적인 모니터링이 요구되고 있다. 네트워크 상황에 대한 효율적인 모니터링 방법의 하나로 시각화 기술이 등장하고 있다. 하지만 대규모, 고속 네트워크의 상황을 표현하기 위해서는 시각화를 위한 데이터를 실시간으로 수집하고 추출하는 것이 매우 중요하다. 본 논문에서는 고속, 대용량 네트워크에서 네트워크 트래픽을 효율적으로 수집하고 추출하여 시각화를 위한 노드와 노드 사이의 관계를 효과적으로 생성할 수 있는 수집시스템을 제안한다. 제안한 시스템은 전체 네트워크의 모든 패킷을 손실 없이 처리하기 위해 비동기 병렬처리 방식을 사용하고 있으며, 실험을 통해 기존 패킷을 수집하는 기술보다 우수함을 검증한다. 실험결과, 제안한 수집 엔진이 CPU 사용량은 Tcpdump에 비하여 상대적으로 높은 사용률을 보이나, 약 0.5%의 패킷 손실률을 보이는 것을 확인할 수 있었다.

Abstract

The effective monitoring is needed as recent advances in ICT technology are complicating systems connected to networks and networks. One of the effective monitoring methods for network situations is the emergence of visualization technology. However, in order to express the situation of a large, high-speed network, it is very important to collect and extract data in real time to express it. In this paper, we propose a collection system that can efficiently collect and extract network traffic from high-speed and high-capacity networks to effectively generate nodes and relations for visualization. The proposed system uses asynchronous parallel processing to handle all packets in the entire network without loss, and tests confirm that it is superior to the technology of collecting existing packets. Experimental results show that the CPU utilization is relatively higher than that of Tcpdump, but the packet loss rate is about 0.5%.

Keywords

network, visualization, monitoring, packet collector

* 영남이공대학교 사이버보안과
- ORCID: http://orcid.org/0000-0001-5152-3897
** 충남대학교 컴퓨터공학과
- ORCID: http://orcid.org/0000-0002-2396-5505
*** 목원대학교 전자공학과(교신저자)
- ORCID: https://orcid.org/0000-0002-6232-476X

· Received: May 29, 2019, Revised: Jul. 15, 2019, Accepted: Jul. 18, 2019
· Corresponding Author: Dae-Sik Ko
Dept. of Electronic Engineering, Mokwon University, 88 Doanbuk-ro, Seo-gu, Daejeon, Korea.
Tel.: +82-42-829-7652, Email: kdsmok@gmail.com

Ⅰ. 서 론

최근 ICT 기술의 발전은 정보시스템의 로그와 이벤트뿐만 아니라 네트워크 트래픽 또한 대용량화 되고 있다. 네트워크 구성이 복잡해지고, 네트워크를 구성하는 시스템이 대용량화됨에 따라 이를 체계적이고 직관적으로 확인하기 위한 시각화 기술이 요구되고 있다. 시각화 기술은 그래프, 도표 등으로 표시하는 2차원 형태의 시각화와 3D 객체를 이용하여 시각적 효과를 높이는 3D 시각화 기술로 나눠질 수 있다. 과거 단순한 로그와 이벤트를 통계기반으로 분석하기 위해 활용되었던 기술이 2D 시각화라고 한다면 최근 복잡하고 다양한 형태의 데이터를 효과적으로 표출하고 이로부터 직관을 얻어내기 위한 노력으로 3D 시각화 기술이 요구되고 있다.

하지만 기본적으로 시각화라는 기술을 구현하기 위해서는 대용량의 데이터를 실시간으로 수집하고 처리하는 기술이 요구된다. 이 기술은 단지 트래픽의 유통을 위한 기술이 아니라 트래픽의 특성을 분석하고, 트래픽을 구성하는 객체의 관계를 분석하는 등 단순 처리의 과정이 아니므로 빠른 처리는 무엇보다도 중요하다[1][2].

본 논문에서는 네트워크 트래픽 시각화를 위한 수집, 분석, 모델링 등 여러 가지 단계 중 전처리를 포함한 트래픽 수집 단계에서 대용량의 트래픽을 효과적으로 수집하기 위한 기술을 연구분석하였다. 이를 위해 시각화를 위한 관계분석을 위한 요구사항을 살펴보고 이를 실현하기 위한 보다 빠른 트래픽 수집기술을 제안하였다. 이를 위하여, 트래픽 수집을 위한 다양한 기술과 시각화에 사용되는 그래프 데이터베이스에 대해 살펴보고 이와 같은 기술이 한계점과 개선점을 분석하였으며 개선된 트래픽 수집기술의 아키텍처를 제안하였다.

Ⅱ. 관련 연구

2.1 시각화에 활용되는 그래프 데이터베이스

그래프 데이터베이스는 노드(정점)와 에지(간선)의 집합으로 이루어진다. 이때 노드 간의 연결을 표현하는 에지는 단순히 연결 관계를 표현하는 것이 아니라 노드와 노드가 어떠한 관계를 맺고 있는지를 보여주는 개념이다[2].

또한, 노드와 에지로 표현되는 객체는 속성을 보유하고 있으며, 이와 같은 속성이 해당 객체를 그래프 데이터베이스에서 서로를 구분해주는 특성이 된다. 이와 같은 그래프 데이터베이스의 구조는 그림 1과 같이 표현할 수 있다.

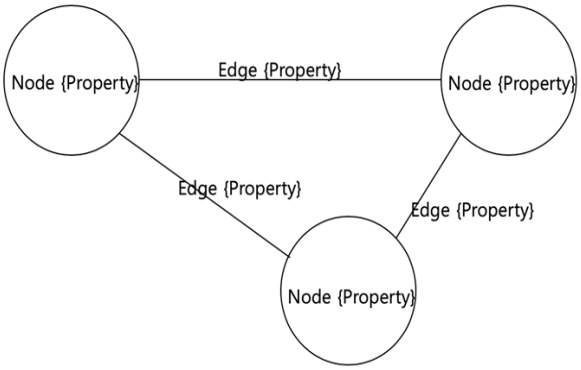

그림 1. 그래프 데이터베이스 구조
Fig. 1. Graph database structure

그래프 데이터베이스는 이와 같은 구조를 제공함으로 기존의 RDBMS보다 자유로운 정보의 표현이 가능하며 비정형 데이터 또한 처리가 가능한 장점이 있다. 이러한 특성으로 인해 그래프 데이터베이스는 다양한 시나리오를 모델링 할 수 있으며 본 논문에서 제안하고자 하는 데이터 모델링 엔진에 적합하다고 판단된다. 2019년 3월 기준 그래프 데이터베이스의 활용 순위를 살펴보면 Neo4j, Microsoft Azure Cosmos DB, Orient DB 순으로 각 점유율은 그림 2와 같이 51%, 27%, 6%의 순으로 확인된다[3].

.

include secondary database models — 32 systems in ranking, May 2019

Rank May 2019	Rank Apr 2019	Rank May 2018	DBMS	Database Model	Score May 2019	Score Apr 2019	Score May 2018
1.	1.	1.	Neo4j	Graph	51.03	+1.54	+10.45
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	27.59	+1.32	+10.06
3.	3.	3.	OrientDB	Multi-model	6.37	+0.18	+1.12
4.	4.	4.	ArangoDB	Multi-model	4.79	+0.50	+1.09
5.	5.	5.	Virtuoso	Multi-model	3.32	+0.01	+1.53

그림 2. 그래프 데이터베이스 엔진 순위
Fig. 2. DB-engines ranking of graph DBMS[3]

2.2 패킷 기반 트래픽 수집기술

패킷은 TCP/IP 프로토콜을 기반으로 하는 네트워크에서 데이터를 전송하는 단위이다. 패킷 교환 네트워크에서는 효과적인 정보를 패킷으로 나누어 인코딩의 과정을 거쳐 전송한다. 전송에 사용되는 패킷은 프로토콜의 종류에 따라 다양한 포맷을 가지고 있다. 패킷은 일반적으로 패킷 헤더와 페이로드의 두 부분으로 구성되며 헤더는 패킷이 네트워크에서 전송되도록 안내하고 패킷의 소스 정보를 표시하는 역할을 한다. 따라서 일반적으로 트래픽을 수집한다는 것은 패킷을 수집한다는 것과 같은 의미로 사용된다. 패킷 캡처는 네트워크 관리에서 정보 수집을 위한 전통적이고 가장 많이 사용되는 방법이다[4]-[8].

패킷을 수집할 때 사용하는 라이브러리인 libpcap 함수는 패킷 흐름의 모든 내용을 수집할 수 있는 기능을 제공한다. 하지만 최근 모바일기기의 확산 및 초고속 네트워크의 등장으로 고속 대규모 네트워크에서 다양한 트래픽을 효과적으로 수집하기에 한계점을 가지고 있다. 이를 해결하는 방법으로 표본추출과 계층화된 표본추출 기법이 제안되었다. 표본추출 메커니즘은 모든 트래픽에서 무작위로 패킷을 추출하는 반면, 계층화된 표본추출 메커니즘은 패킷을 분류하고 그룹에 따라 일부 패킷을 버리는 방법을 사용한다. 이와 같은 방법은 고속, 대용량 네트워크에서 빠르게 패킷을 수집하지만, 패킷 일부를 수집하여 전체로 확장하는 방법이기 때문에 트래픽 간의 관계를 완전하게 표시하기에는 한계가 있다.

2.3 플로우 기반 트래픽 수집기술

플로우 기반 수집은 네트워크 자료수집을 위한 또 다른 방법이다. 플로우 기반 수집기술에서 플로우는 일정 기간 특정 관측 지점을 통과하는 같은 특성을 가진 패킷의 집합으로 정의할 수 있다. 따라서, 전체 네트워크 트래픽에 대한 보다 완전한 수집을 위해서는 네트워크 내의 모든 네트워크 시스템으로부터 플로우를 수집해야 한다. 하지만 통상적으로 네트워크 인입구간 백본 스위치는 모든 사이버 위협과 공격에 관한 중요한 데이터를 얻을 수 있으므로 가장 효과적으로 감시하고 제어할 수 있는 노드로 활용된다. 플로우 기반 수집의 가장 일반적인 메커니즘은 출발지 주소, 목적지 주소, 출발지, 포트, 목적지 포트, 프로토콜 유형을 포함한 5-tuple 자료를 수집하는 것이다. 이와 같은 플로우 기반 트래픽 수집기술은 Netflow와 같은 기술로 구현되어 있다[9][10]. 그러나 흐름은 단지 5-tuple 흐름으로만 정의되는 것은 아니다. 수신 및 송신 필터링 메커니즘은 흐름 자료를 수집하고 감시하는 또 다른 메커니즘이다. 5-tuple 흐름과는 다르게 유입 및 송신 필터링 메커니즘은 전체 패킷 헤더에 초점을 맞춘다. 하지만, 전체 트래픽 중 5-tuple정보를 수집하는 것만으로도 네트워크의 각 노드에 대한 연결 관계 정보를 분석하는 것이 가능하므로 시각화를 위해서는 반드시 페이로드를 포함한 전체 트래픽을 수집할 필요는 없다.

2.4 로그 기반 트래픽 수집기술

로그는 이벤트 로그와 메시지 로그로 구성될 수 있다. 이벤트 로그는 필요한 경우 사용자 추적, 이벤트 상태 및 진단 실패를 기록하는 로그이다. 서비스를 가동하면 로그 파일이 생성된다. 사용자 프라이버시 문제로 인해, 인터넷 릴레이 채팅, 인스턴트 메시징 등을 포함한 메시지 로그는 일반적으로 서비스 제공자들에 의해 암호화된다. 이벤트 로그는 일반적으로 자료수집에 사용된다. 로그의 출처에 따르면 운영 체제 로그, 웹 로그, 장비 로그로 구성된다. 로그 파일에는 표준 형식이 없다. 그리고 각 클래스의 로그 파일 유형을 계산하는 것은 너무 어렵다. 이와 같은 로그 검출은 자료수집에 적용할 수 있는 또 다른 메커니즘이다. 그러나 로그 파일은 일반적으로 큰 메모리를 차지하며 정보 밀도가 낮으며 복잡한 파일 형식으로 되어있다. 따라서 로그 파일을 수동으로 처리하고 각 독립 로그에서 유효한 기능을 추출하는 것은 단순한 작업이 아니다. 이러한 문제를 해결하기 위해 자동 및 적응 솔루션을 제안한 연구가 있다.

수집기가 탐지기에서 실시간 로그를 조사하거나 저장소에서 파일을 획득한 경우 필터링 구성요소는

로그의 내용을 구문 분석하여 필수적인 기능을 추출한다. LogParser는 여러 로그 파일의 일반적인 로그 분석 작업을 수행하기 위한 간단한 로그 선별 도구가 될 수 있다[11].

하지만, 본 논문에서 다루고자 하는 시각화는 방대한 네트워크에 대한 현재의 전체상황을 실시간으로 시각화하는 것이기 때문에 로그 기반 트래픽 수집기술은 적절하지 않다. 로그 기반 수집기술은 트래픽을 있는 그대로 수집하지 않고 각 시스템 또는 소프트웨어에서 정의한 형식을 사용하기 때문에 얼마나 자세한 정보가 수집되는지, 얼마나 완전하게 수집되는지 등에 대해 보증할 수 없기 때문이다.

표 1은 패킷 기반 트래픽 수집기술과 플로우 기반 수집기술의 특징을 비교한 것이다.

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

Item	Full packet	Flow based
Collection scope	5-tuple, payload	5-tuple
Collection performance	Slow	Fast
Parsing performance	Slow	Fast
Strengths	Collect as much traffic as need	Fast processing speed
Weaknesses	Slow processing speed vast storage space required	Additional information collection method required

III. 고속 대용량 트래픽 수집을 위한 수집시스템

3.1 시각화에 필요한 기술적, 정보적 요소

네트워크 상태에 대한 시각화를 위한 요소로는 기술적인 요소와 정보적인 요소가 있다. 정보적인 요소는 시각화 화면에 표출되는 정보를 말하는 것으로 네트워크의 상태를 얼마나 자세하고 의미 있게 표출해 줄 수 있는 것인가를 결정하는 요소이다. 기술적인 요소는 이러한 정보를 얼마나 효과적으로 추출할 수 있는가에 대한 성능적인 측면이라 할 수 있다.

먼저, 네트워크 상태를 시각화하는 데 필요한 기술적인 요소는 실시간성이다. 즉 실시간으로 송수신되는 트래픽 정보를 지연 없이 처리하여 화면에 트래픽 상태의 변화를 표시해 줄 수 있어야 한다. 두 번째로 정보적인 요소는 화면에 표출하는 노드에 대한 정보와 각 노드의 상태정보에 따른 변화를 표시하기 위한 속성정보로 구분할 수 있다. 먼저 노드에 대한 정보로는 플로우 기반 수집기술에서 언급한 5-tuple 정보가 모두 필요하며, 에지를 구성하기 위한 표 2와 같은 정보가 필요하다.

표 2에서 두 개의 카테고리는 노드와 에지이다. 각 노드의 경우에는 패킷 내에 포함된 5-tuple 정보 중 출발지 주소, 목적지 주소, 출발지 포트, 목적지 포드 등이 포함될 수 있다.

관계의 경우에는 단일 패킷 내에 포함된 정보뿐만 아니라, 일정하게 정해진 시간 동안 같은 출발지와 목적지 간에 주고받은 패킷의 양, 패킷 수, 연결 수 등이 필요하고 이를 정의하기 위해 속성으로 각 관계의 출발지 노드와 목적지 노드가 필요하다. 이처럼 관계를 정의함으로 인해 특정한 두 시스템 간 트래픽의 유통 상황을 알 수 있으며, 이를 그룹화하여 특정 관심 있는 그룹에 대한 상태정보를 실시간으로 알 수 있다.

표 2. 시각화를 위한 속성정보
Table 2. Characteristics for visualization

Label	Object	Attribute
Note	IP address	Object ID, IP address
	Port number	Object ID, Port number
	Group	Name Included object ID
Relation (Edge)	Inbound	bps, pps, cps for each between IP and Protocol, Start object, End object
	Outbound	bps, pps, cps for each between IP and protocol, Start object, End object

이러한 정보를 생성하기 위해서는 수집되는 패킷을 단순히 파싱하는 것뿐만 아니라 파싱된 원본 패킷에 포함된 요약정보를 추출하여 연사하는 과정이 실시간으로 이루어져야 한다.

3.2 Suricata를 이용한 데이터 수집기 설계

본 논문에서 제안하는 방식은 실시간 자료수집과 처리를 위해 실시간으로 트래픽을 수집하는 수집 모듈과 수집된 데이터를 파싱하고 연산하는 전처리기, 그리고 전처리가 완료된 정보를 시각화를 위한 데이터 모델로 생성하는 모델러(Modeler)로 구성하였다. 또한, 더욱 빠른 처리를 위해 전처리기와 모델러는 메모리상에서 동작하도록 설계하였다.

자료수집 모듈은 상시적 미러링 상태로 대기하고, 패킷이 수집되면 그 즉시 수집된 패킷을 로컬 시스템에 파일로 기록한다. 파일로 기록된 정보는 병렬로 구성된 전처리기에서 읽어 노드 정보와 통계정보를 추출한다. 이때 병렬 프로세스는 비동기 방식으로 파일 생성이 완료될 때마다 생성되며, 파일에 대한 전처리가 완료되면 종료된다. 병렬처리로 전처리기가 빠르게 데이터 모델 생성을 위한 정보를 생산한 후 모델러로 전송하면 모델러에서는 시각화를 위한 표 2에 포함된 노드 정보와 관계 정보를 생성하여 그래프 데이터베이스로 저장한다. 이와 같은 구조를 그림으로 나타내면 그림 3과 같다.

그림 3은 대용량 트래픽 데이터 수집 아키텍처이다. 여기서 수집 엔진은 패킷수집기(Collector)와 전처리기(Pre-processor) 그리고 모델러로 구성된다.

먼저 패킷 수집기에 관해 본 연구에서는 알려진 오픈소스 기반 수집 기술인 Suricata 엔진을 일부 수정하여 사용하였다. 관련 연구에서 언급한 바와 같이 트래픽 수집기술은 패킷 기반, 플로우 기반, 로그 기반 기술이 존재하나 본 연구에서 달성하고자 하는 목적인 전체 네트워크의 종합적인 상황 모니터링 관점에서 플로우 기반 수집기술과 로그 기반 수집기술은 자료수집에 한계가 존재하여 본 연구에서는 패킷 기반 수집기술을 활용하였다. 알려진 패킷 기반 수집기술은 Tcpdump(tshark) Suricata 등과 같은 오픈소스 소프트웨어 기반 기술이 존재한다 [12]-[14]. Tcpdump는 Packet 데이터 자체를 있는 그대로 처리하기 때문에 유연한 전처리가 가능하다. 하지만 전처리를 위해 수집한 RAW 데이터를 전처리기가 읽을 수 있는 데이터로 변환해야 하는 추가적인 과정과 싱글 프로세서로 버퍼 공간의 제한으로 인해 대용량 네트워크에서 자료수집 시 패킷 유실의 가능성이 존재한다. 반면 Suricata는 출력 데이터 형태가 JSON 포맷으로 출력할 수 있어, 더욱 빠른 전처리가 가능하고, 멀티프로세스를 지원한다. 따라서 멀티프로세스 처리를 위해 CPU 사용량이 많아지지만 Tcpdump 대비하여 패킷 유실에 대한 제한으로부터 자유롭다. 이러한 이유로 본 연구에서 기본적인 수집모듈로는 병렬처리가 가능한 Suricata 엔진을 활용하였다. 수집모듈에서 수집된 데이터는 일정 시간 또는 설정된 일정 RAW데이터 크기마다 수집한 패킷을 파일로 생성하도록 하였으며, 파일이 생성되면 이를 인지하여 전처리기가 실행되도록 설계 하였다.

기본적으로 Suricata는 패킷을 수집하는 라이브러리인 libpcap을 기반으로 여러 가지 기능들이 묶여 있다. 본 연구에서는 Suricata의 여러가지 모듈 중 패킷 캡쳐를 담당하는 libpcap 라이브러리를 리눅스 커널에서 제공하는 패킷 캡쳐 라이브러리인 pf_ring 라이브러리로 교체하여 Suricata를 리빌딩 하였다.

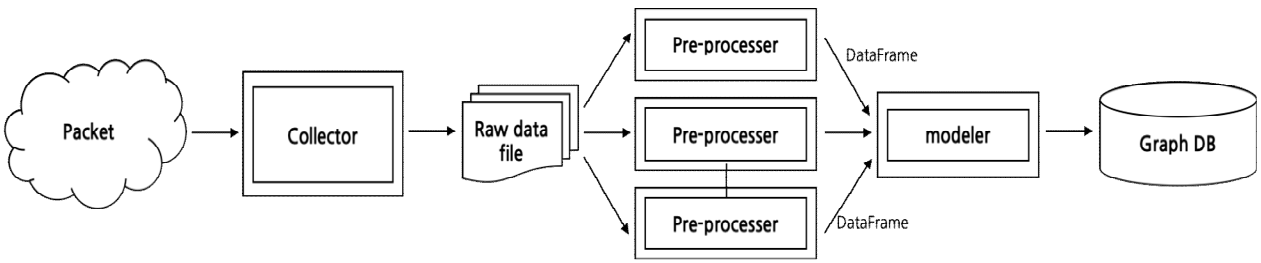

그림 3. 수집시스템 아키텍처
Fig. 3. Collection system architecture

전처리기는 파이썬을 활용하여 개발하였으며, 패킷 수집기에서 전달된 JSON 형식 데이터를 이후 모델러에서 모델링이 가능한 형태로 처리하기 위해 Pandas 포맷으로 출력한다[15].

Pandas는 파이썬에서 사용하는 데이터 분석 라이브러리로 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며, 보다 안정적으로 대용량의 데이터를 처리하는 도구이다. 전처리기는 패킷 수집기에서 수집하는 패킷 데이터를 일정한 시간 단위로 나누어 분할된 데이터가 생성될 때마다 비동기적으로 프로세스가 생성되도록 하는 비동기 병렬처리 방식을 사용하였다. 이때, 패킷 데이터를 나누기 위한 시간의 단위는 네트워크의 상태에 따라 동적으로 설정할 수 있다.

모델러는 파이썬을 기반으로 개발하였으며, 전처리기에서 파싱이 완료된 데이터를 입력으로 받아, 표 2에 표시된 관계 정보를 연산한 후 그래프 데이터베이스에 입력하도록 개발하였다.

IV. 실험 및 평가

본 연구에서는 제안된 수정 Suricata를 이용하는 대용량 데이터 수집기법의 성능을 실험을 통하여 확인하였다. 수집 엔진의 3가지 구성요소 중 모델러의 경우 파싱이 완료된 데이터를 분석하여 그래프 데이터베이스에 입력해주는 역할이기 때문에 실제 수집 엔진의 성능에 큰 영향을 주는 부분은 아니며, 실험의 범위는 패킷 수집기, 전처리기의 처리 성능이다. 이를 위한 실험환경은 표 3과 그림 4와 같다.

실험은 제안하는 데이터 수집기의 패킷의 처리율 즉 패킷손실율과 리소스 사용량을 중점적으로 확인하여 제안한 수집 엔진이 대량의 트래픽을 지연 없이 처리할 수 있는지 확인하였다.

패킷을 생성하고 생성한 패킷에 대해 수집 엔진의 수집기에서 수집하여 전처리기로 전달하는 실제 데이터는 그림 5와 같이 데이터가 전달된다.

표 3. 실험 시스템 사양
Table 3. System spec. of the experimental system

System	Performance(Spec)
Packet Generator	CPU: 3.5Ghz 4Core 8Thread Memory: 32GB Interface: 10Gbps 1Port
Collector	CPU: 3.7Ghz 6Core 6Thread Memory: 64GB Interface: 10Gbps 1Port

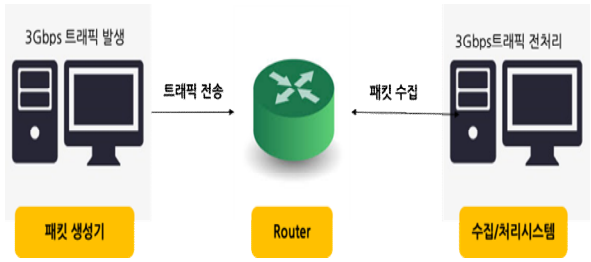

그림 4. 실험환경 구성
Fig. 4. Experimental environment

{"timestamp":"2019-03-13T14:27:38.610767+0900","flow_id":992625272967631,"in_iface":"enp3s0","event_type":"alert","src_ip":"192.168.0.196","src_port":62701,"dest_ip":"239.255.255.250","dest_port":1900,"proto":"UDP","alert":{"action":"allowed","gid":1,"signature_id":1000000,"rev":0,"signature":"test1","category":"","severity":3},"app_proto":"failed","flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":216,"bytes_toclient":0,"start":"2019-03-13T14:27:38.610767+0900"},"payload":"TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSE9TVDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1BTjogInNzZHA6ZGlzY292ZXIiDQpNWDogMQ0KU1Q6IHVybjpkaWFsLW11bHRpc2NyZWVuLW9yZzpzZXJ2aWNlOmRpYWw6MQ0KVVNFUi1BR0VOVDogR29vZ2xlIENocm9tZS83MC4wLjM2MjYuMTIxIFdpbmRvd3MNCg0K","payload_printable":"M-SEARCH * HTTP\/1.1\r\nHOST: 239.255.255.250:1900\r\nMAN: \"ssdp:discover\"\r\nMX: 1\r\nST: urn:dial-multiscreen-org:service:dial:1\r\nUSER-AGENT: Google Chrome\/72.0.3626.121 Windows\r\n\r\n","stream":0,"packet":"AQBef\/\/6ANhhM7P9CABFAADKOSMAAAERzpnAqADE7\/\/\/+vTtB2wAtlYOTS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSE9TVDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1BTjogInNzZHA6ZGlzY292ZXIiDQpNWDogMQ0KU1Q6IHVybjpkaWFsLW11bHRpc2NyZWVuLW9yZzpzZXJ2aWNlOmRpYWw6MQ0KVVNFUi1BR0VOVDogR29vZ2xlIENocm9tZS83Mi4wLjM2MjYuMTIxIFdpbmRvd3MNCg0K","packet_info":{"linktype":1}}
{"timestamp":"2019-03-13T14:27:40.036481+0900","flow_id":1193552433155713,"in_iface":"enp3s0","event_type":"alert","src_ip":"172.217.31.174","src_port":443,"dest_ip":"192.168.0.179","dest_port":52849,"proto":"UDP","alert":{"action":"allowed","gid":1,"signature_id":1000000,"rev":0,"signature":"test1","category":"","severity":3},"app_proto":"failed","flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":85,"bytes_toclient":0,"start":"2019-03-13T14:27:40.036481+0900"},"payload":"ENFZdIsi51PfELZ1e\/oYljX0\/8P3j5PdbZunfwjp4T+qU5+QcHhaRSRNDw==","payload_printable":"..Yt.\".S...u{...5.....m......?.S..pxZE$M.","stream":0,"packet":"ANhhM7PsoKsb3FXXCABFgABHAABAADkRs0Os2R+uwKgAswG7znEAM5JMENFZdIsi51PfELZ1e\/oYljX0\/8P3j5PdbZunfwjp4T+qU5+QcHhaRSRNDw==","packet_info":{"linktype":1}}
{"timestamp":"2019-03-13T14:27:41.777673+0900","flow_id":1288492685319625,"in_iface":"enp3s0","event_type":"alert","src_ip":"192.168.0.179","src_port":50934,"dest_ip":"184.25.17.222","dest_port":443,"proto":"TCP","alert":{"action":"allowed","gid":1,"signature_id":1000000,"rev":0,"signature":"test1","category":"","severity":3},"flow":{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":66,"bytes_toclient":0,"start":"2019-03-13T14:27:41.777673+0900"},"payload":"","payload_printable":"","stream":0,"packet":"oKsb3FXXANhhM7PsCABFAAA0+39AAEAGs\/HAqACzuBkR3sb2AbsiWaeAzFBYiYAQAPW1SwAAAQEICnPNgdj4PI\/h","packet_info":{"linktype":1}}

그림 5. 수집기에서 전처리기로 송신하는 데이터
Fig. 5. Data sent from collector to pre-processor

본 연구에서는 대용량 트래픽 데이터 수집기로 널리 사용되고 있는 Tcpdump와 제안한 수집 엔진을 동일한 조건으로 실험에서 비교분석하였다. 실험은 3Gbps데이터를 60초 동안 발생시키고 이를 Tcpcump와 제안한 수집시스템에의 처리에 따른 자원 사용량과 손실율을 분석하였다. 실험은 동일한 환경에서 총 30회 이상 실행되었으며, 결과는 표 4와 같다. 표 4에서 발생 패킷은 동일한 조건에서 동일한 패킷을 발생시켜 미세한 차이는 있으나 전체적으로 동일한 수준으로 표시하였으며, 매 실험 시 CPU사용량의 평균과 매 실험 시 확인된 패킷 손실율의 평균을 나타낸다. 실험결과, 제안한 수집 엔진이 CPU 사용량은 Tcpdump에 대비 상대적으로 높은 사용률을 보이나, 약 0.5%의 패킷 손실률을 보이는 Tcpdump 대비 제안한 엔진은 손실 없이 처리하고 있음을 확인할 수 있다.

표 4. 시험결과
Table 4. Test result

Category	Tcpdump	Proposed system
Generated packet	12,000 pkt, 3,000Mbps, 850kbps	
CPU usage	70% (average)	180% (average) 30% for core
Drop rate	0.5% (108 kpkt)	0%

V. 결 론

본 논문에서는 대용량 네트워크의 상태를 실시간으로 감시하기 위한 기술로 최근 주목받고 있는 시각화를 위한 네트워크 트래픽 자료 수집을 위한 수집 엔진의 성능을 분석하였다. 이를 위해 먼저 네트워크 시각화를 위한 수집조건과 수집해야 하는 요소들을 정리하고 분석하였으며, 이를 효과적으로 수집하고 처리할 수 있도록 수집기와 전처리기를 분리하고 전처리기를 비동기 방식으로 병렬처리할 수 있도록 설계하였다. 본 논문에서는 데이터 수집을 위한 오픈소스인 Suricata엔진의 일부 수정하여 대용량 데이터 수집 시 패킷손실율을 최소화할 수 있도록 하였다. 실험결과, 제안한 수집 엔진이 CPU 사용량은 Tcpdump에 대비 상대적으로 높은 사용률을 보이나, 약 0.5%의 패킷 손실률을 보이는 Tcpdump 대비 제안한 엔진은 손실 없이 처리하고 있음을 확인할 수 있다.

본 연구의 결과는 향후 대용량 네트워크의 다양한 정보를 수집하여 분석하는 위협탐지시스템, 성능관리시스템 등에 활용이 가능할 것으로 기대된다.

References

[1] C. Y. Jeong, S. G. Sohn, B. H. Chang, and J. C. Na, "An Efficient Method for Analyzing Network Security Situation Using Visualization", Journal of the Korea Institute of Information Security & Cryptology, Vol. 19, No. 3, pp. 107-117, Jun. 2009.

[2] B. S. Min, "Visualization of BlockChain Data Structure based on Graph Database", Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 480-482, Jan. 2018.

[3] DB-Engines Ranking of Graph DBMS, https://db-engines.com/en/ranking/graph+dbms. [accessed: May 05, 2019]

[4] 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.

[5] 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

[6] 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

[7] X. An and X. Liu, "Packet capture and protocol analysis based on Winpcap", 2006 International

Conference on Robots & Intelligent System (ICRIS), Beijing, China, pp. 272-275, Oct. 2006

[8] 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.

[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), Cambridge, 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, Oct. 2012

[12] Manpage of TCPDUMP, https://www.tcpdump.org/manpages/tcpdump.1.html. [accessed: Apr. 04, 2019]

[13] tshark - Dump and analyze network traffic, https://www.wireshark.org/docs/man-pages/tshark.html. [accessed: Apr. 04, 2019]

[14] Suricata - Open Source IDS/IPS/NSM engine, https://suricata-ids.org [accessed: Apr. 04, 2019]

[15] padas, https://pandas.pydata.org/. [accessed: Apr. 04, 2019]

저자소개

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

천 은 영 (Eun-Young Cheon)

2002년 2월 : 한남대학교 컴퓨터공학과(공학사)
2004년 2월 : 한남대학교 컴퓨터공학과(공학석사)
2013년 8월 : 충남대학교 컴퓨터공학과 (박사수료)
2009년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 강사
관심분야 : 소프트웨어아키텍처, MDA, SPLE

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