네트워크 트래픽 발생량 분석을 활용한 연결 토폴로지 탐색

박지태, 구영훈, 백의준, 김명섭
고려대학교

{pjj5846, gyh0808, pb1069, tmskim}@korea.ac.kr

A Connection Topology Discovery using Generated Network Traffic Analysis

Jee-Tae Park, Young-Hoon Goo, Ui-Jun Baek, Myung Sup Kim
Korea Univ.

요 약

최근 과학 기술의 발전으로 인한 네트워크 환경을 발전으로 다양한 응용과 대용량의 트래픽이 발생하고 있다. 이러한 추세에 따라 네트워크 트래픽 과부하, 외부 악성 행위와 같은 네트워크 장애 현상이 빈번하게 발생하고 있다. 네트워크 장애 현상을 신속하게 탐지하고, 원인 분석 및 예방하기 위해서 네트워크 장비들의 연결 토폴로지에 대한 직관적인 정보가 필요하다. 하지만 기존의 네트워크 토폴로지 탐색하는 방법은 관리자가 수동적으로 관리 하기 때문에 많은 시간과 노력이 들며, 제한된 장비 만을 탐색 하거나, 연결을 탐색하더라도 연결 포트 정보는 탐색하지 못한다는 문제점이 있다 따라서 본 논문에서는 연결이 정의 된 네트워크 장비를 대상으로 장비의 네트워크 트래픽 발생량 분석을 통해 연결 포트를 탐색하는 방법을 제안한다. 제안하는 방법을 검증하기 위해서 실제 네트워크 장비에 대해 실험을 진행하였으며, 실험 결과의 검증을 통해 본 논문의 타당성을 증명한다.

Ⅰ. 서 론 [1]

오늘 날의 인터넷은 네트워크 고속화 및 과학 기술의 급격한 발전으로 다양한 응용과 대용량의 트래픽이 발생하고 있다. 이에 따라 네트워크 트래픽 과부하, 네트워크 악성 행위와 같은 다양한 네트워크 장애 현상이 발생하고 있다. 네트워크 장애 현상에 신속하게 대응하기 위해서 네트워크 관리 시스템(Network Management System)을 구축하여 여러 가지 네트워크 결함에 대한 분석이 필요하다[1,2]. 현재 네트워크 관리 시스템은 네트워크 자원 관리, 서버 관리, 장애 현상 원인 분석 및 예방의 기능 제공하며, 이러한 기능들을 효율적으로 관리하기 위해서 관리자는 네트워크 연결 토폴로지에 대해 알아야한다[1].

관리자가 네트워크 장비들의 연결 구성을 수동으로 관리 할 경우, 많은 시간과 노력, 비용이 든다. 따라서 네트워크 토폴로지 탐색을 위해 여러 가지 방법론 연구되어 왔으며, 가장 널리 알려진 방법이 Traceroute 이다[2].

Traceroute 를 활용한 방법은 쉽고 편리하다는 장점이 있지만, 목적지 중복 탐색 및 대규모의 네트워크에서 사용하기 어렵다는 문제점이 있다. Traceroute 이외의 기존 방법들 역시 제한된 계층의 장비만을 탐색하거나, 연결 포트 정보를 탐색 할 수 없거나, 시간 및 리소스의 낭비가 심하다는 문제점이 존재한다.

따라서 본 논문에서는 이러한 문제점을 해결하기 위해서 SNMP 를 활용하여 네트워크 장비 MIB 정보를 저장하고, 저장 된 정보를 활용하여 트래픽 발생량 분석 및 네트워크 토폴로지를 탐색하는 방법을 제안한다.

이 논문은 2018 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구 사업이며(NRF-2018R1D1A1B07045742), 2018 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00539-002, 블록체인의 트랜잭션 모니터링 및 분석 기술개발)

본 논문의 구성은 다음과 같다. 1 장 서론에서 네트워크 토폴로지 탐색 연구 필요성과 기존 연구의 문제점을 설명하고, 2 장 관련 연구에서는 기존에 존재하는 여러 가지 네트워크 장비 탐색 방법론에 대해 언급하고, 각 방법론에 대한 장단점을 분석하였다. 3 장 본론에서 제안하는 토폴로지 탐색 알고리즘 설명하고, 4 장에서 실제 네트워크 장비에 대한 실험을 진행한다. 마지막으로 5 장에서 결론 및 향후 연구를 언급하고 마친다.

Ⅱ. 관련 연구

대부분의 네트워크 관리 시스템에서는 네트워크 자동 탐색 기능을 활용하여 네트워크 장비들의 연결 구성과 연결 변화에 대해 신속하게 모니터링하고 있다. 하지만 네트워크 자동 탐색 기능은 대부분 3 계층까지의 장비들만 탐색하기 때문에 2 계층의 네트워크 장비에 대한 관리 방안이 필요하다.

최근에 이러한 문제를 해결하기 위해 여러 방법론이 연구되어 왔으며, 가장 잘 알려진 방법은 Traceroute 를 활용한 방법이다.

먼저 Traceroute 를 활용한 방법은 출발지에서 목적지까지의 경로를 식별하기 위해 패킷을 보낸다. 이 때, 패킷 헤더의 TTL(Time-To-Live) 값을 조정하여 경로 상에 있는 네트워크 장비를 식별하고, 해당 경로에 대한 정보를 수집한다. 하지만 네트워크 장비 수가 많은 기업용 혹은 대규모의 네트워크를 대상으로 적용할 경우, 시간이 오래 걸린다는 문제점이 있다[2]. 또한, 출발지와 도착지가 다르더라도, 이전에 탐색한 중복되는 경로를 탐색하기 때문에, 비효율적이며, 네트워크 부하가 많이 걸린다[1,5].

따라서 최근에는 LLDP 및 BDDP 와 같은 2 계층 네트워크 장비 탐색 방안이 연구되고 있다[3]. LLDP 는 IEEE 802.1AB 에서 표준으로 정의 된 프로토콜로,

LLDP 를 활용하여 연결 된 2 계층의 장비 연결 정보를 알 수 있다. 하지만 LLDP 는 또한, 장비 간 연결된 포트 정보는 직접 연결 된 장비 간의 연결 정보만 알 수 있으며, 오래 된 네트워크를 대상으로 적용 할 수 없다는 단점이 있다. BDDP 는 LLDP 와 마찬가지로 2 계층 장비의 연결 정보를 편리하게 알 수 있다. 그리고 오픈 소스로 구현되어 있기 때문에 쉽게 사용 할 수 있다는 장점이 있다. 하지만 BDDP 는 브로드캐스트 형태로 패킷을 전송하기 때문에 시간과 리소스를 많이 소모하며, LLDP 와 마찬가지로 오래된 네트워크를 대상으로 사용하기 어렵다는 단점이 있다[3].

이러한 문제점을 해결하기 위해서 SNMP 기반의 네트워크 장비 탐색 방법이 연구되고 있다[4]. SNMP 를 활용하면 서버에서 대상 에이전트의 여러 정보를 쉽게 알 수 있다. SNMP 기반의 네트워크 장비 탐색 방법은 이러한 정보를 취합하여 네트워크 토폴로지를 형성한다. SNMP 를 활용한 방법은 다른 방법에 비해 쉽고, 편리하며, 여러 계층의 장비들에 대해 적용 가능하기 때문에 범용성이 높다[4]. 하지만 기존의 SNMP 기반 탐색 방법은 BRIDGE MIB 정보에 대한 의존도가 매우 크다. BRIDGE MIB 정보는 BRIDGE TABLE 로 구성되어 있으며, 부정확하게 구성 될 가능성이 있으며, 테이블 갱신 간격이 길어질수록 2 계층 장비를 탐색하기 어렵다는 문제점이 존재한다.

따라서 본 논문에서는 SNMP 기반 네트워크 장비 탐색의 장점을 유지하면서 문제점을 해결하기 위해 새로운 탐색 알고리즘을 제안한다. 제안하는 알고리즘은 BRDIGE MIB 정보를 사용하지 않기 때문에 존재하는 여러 문제점을 해결 할 수 있다.

Ⅲ.. 본론

제안하는 방법은 연결이 정의 된 장비 들을 대상으로 연결 포트를 탐색하는 방법이다. 연결 포트를 탐색하기 위해서 연결 된 장비 간에 발생하는 트래픽 양의 비교한다

이 때, SNMP MIB 정보로 ifInOuctet, ifOutOuctet 을 사용하며, 해당 장비에서 각 포트 별 발생하는 트래픽 양을 알 수 있다. ifOutOuctet 에 대한 예시는 그림 1 에 나타나있다. 그림 1 에서 대부분의 인덱스는 연결 되어있지 않기 때문에 트래픽이 발생하지 않으며, 14, 21, 28 인덱스에서 트래픽이 발생하고 있다. 이 때, 발생한 트래픽 양을 비교하여 같으면 인덱스와 맵핑 된 포트를 찾는다.

```
[                    ]# snmpwalk -v 2c -c                    1.3.6.1.2.1.2.2.1.16
IF-MIB::ifOutOctets.1 = Counter32: 0
IF-MIB::ifOutOctets.2 = Counter32: 0
IF-MIB::ifOutOctets.3 = Counter32: 0
IF-MIB::ifOutOctets.4 = Counter32: 0
IF-MIB::ifOutOctets.5 = Counter32: 0
IF-MIB::ifOutOctets.6 = Counter32: 0
IF-MIB::ifOutOctets.7 = Counter32: 0
IF-MIB::ifOutOctets.8 = Counter32: 0
IF-MIB::ifOutOctets.9 = Counter32: 0
IF-MIB::ifOutOctets.10 = Counter32: 0
IF-MIB::ifOutOctets.11 = Counter32: 0
IF-MIB::ifOutOctets.12 = Counter32: 0
IF-MIB::ifOutOctets.13 = Counter32: 0
IF-MIB::ifOutOctets.14 = Counter32: 70109012
IF-MIB::ifOutOctets.15 = Counter32: 0
IF-MIB::ifOutOctets.16 = Counter32: 0
IF-MIB::ifOutOctets.17 = Counter32: 0
IF-MIB::ifOutOctets.18 = Counter32: 0
IF-MIB::ifOutOctets.19 = Counter32: 0
IF-MIB::ifOutOctets.20 = Counter32: 0
IF-MIB::ifOutOctets.21 = Counter32: 188594904
IF-MIB::ifOutOctets.22 = Counter32: 0
IF-MIB::ifOutOctets.23 = Counter32: 0
IF-MIB::ifOutOctets.24 = Counter32: 0
IF-MIB::ifOutOctets.25 = Counter32: 0
IF-MIB::ifOutOctets.26 = Counter32: 0
IF-MIB::ifOutOctets.27 = Counter32: 0
IF-MIB::ifOutOctets.28 = Counter32: 1103156
```

Figure 1. ifOutOuctet MIB 정보 예시

하지만 MIB 정보를 저장하는 시간이 다를 때는 한 시점에서 발생하는 트래픽 양이 다를 수 있기 때문에 정확한 연결 정보가 나오지 않는다. 따라서 각 장비에서 10 초 간 발생한 트래픽 양을 비교하여 연결 토폴로지를 탐색한다.

제안하는 방법의 알고리즘은 그림 2 에 나타나있다. 먼저 입력으로 연결 된 장비의 IP 리스트가 들어온다. 입력으로 들어온 연결 된 장비의 IP 에 대해 쌍을 A, B 라고 정할 때, A 의 ifInOuctet, ifInOuctet(10 sec)과 B 의 ifOutOuctet, ifOutOuctet(10 sec)을 을 구한다. 이 때, ifInOuctet(10 sec)는 ifInOuctet 가 저장 되고 10 초 후에 저장 된 ifInOuctet 로, 두 값의 차이를 구하면 해당 인덱스에서 10 초 동안 발생한 트래픽 양을 알 수 있다. 연결 된 두 장비에서 연결 인덱스를 찾으면, ifName MIB 정보를 활용하여 연결 된 포트를 찾을 수 있다.

Algorithm 1 : Device Port Connection Discovery
Input : List of Device IP, MIB Information of Devices, Connected Information
Output : Connection Information
Notation – D : List of Device IP / In : Interface Information
Diff : Difference of ifInOuctet (10 sec) and ifInOuctet or ifOutOuctet (10 sec) and ifOutOuctet / In : Interface Information of the Device

1. Get the IP List of the Devices
2. Set the Connected Device Pair $[D_i, D_j]$
3. **for** i=0 **to** All Devices
4. **for** j=i+1 **to** All Devices
5. Load ifInOuctet of D_i and ifInOuctet (10 sec) of D_i
6. Load ifOutOuctet of D_j and ifOutOuctet (10 sec) of D_j
7. Get the difference of ifInOuctet (10 sec) and ifInOuctet of D_i
8. Get the difference of ifOutOuctet (10 sec) and ifOutOuctet of D_j
9. **if** $Diff_i$ == $Diff_j$
10. Get the Ouctet Index of $[D_i, D_j]$
11. Get the Interface Information of $[D_i, D_j]$
12. Store the In_i, In_j and D_i, D_j

Figure 2. 장비 포트 연결 탐색 알고리즘

최종 결과로 연결 된 장비 타입과, IP 정보, 연결 포트를 알 수 있다. 제안하는 방법에서는 연결 정보를 csv 파일로 저장하여, 한 행에 연결 정보를 저장하였다. 도출한 csv 파일은 데이터베이스를 활용하여 네트워크 관리 시스템에서 연결 토폴로지 UI 정보로 제공 될 수 있다.

Ⅳ. 실험

제안하는 방법을 검증하기 위해 실제 네트워크 장비를 대상으로 실험을 진행하였다. 실험에 사용한 네트워크 장비는 L2 스위치 2 대, L3 스위치 2 대이며, 실험 환경은 그림 3 에 나타나있다.

여러 계층 장비 간의 연결을 확인 하기 위해 그림 3 과 같이 구성하였으며, 호스트에서 제안하는 시스템을 실행한다. 이 때, 설치 된 시스템은 Linux CentOS 7 에서 쉘 스크립트와 C/C++로 구현되었다.

제안하는 방법을 통해 그림 3 의 L2-L2, L2-L3, L3-L3 연결에 대해 정확하게 연결 포트를 탐색할 수 있었다. 탐색 된 연결 및 연결 포트 정보는 그림 4 에 나타나있다. 2 장 본론에서 언급한대로, 저장 된 csv 파일은 한 행에서 연결 된 두 장비의 IP, 포트 정보가 저장된다.

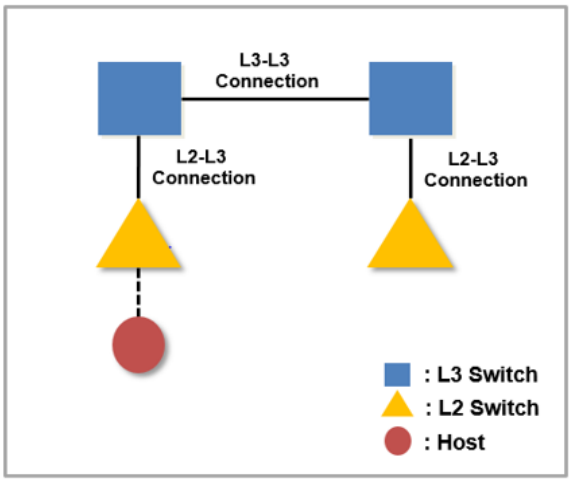

Figure 3. 실험 네트워크 구성도

L3Switch0	Fa0/1	192.168.30.	L3Switch1	Fa0/1	192.168.30.
L2Swtich0	Fa0/2	192.168.20.	L3Switch1	Fa0/1	192.168.20.
L2Switch0	Fa0/1	192.168.20.	L2Switch1	Fa0/7	192.168.20.

Figure 4. 연결 포트 탐색 결과

하나의 토폴로지만으로 제안하는 방법을 검증하기 어렵기 때문에, 네 대의 네트워크 장비로 구성할 수 있는 모든 토폴로지에 대해 실험을 진행하였다. 구성 할 수 있는 토폴로지 수는 4 가지 이며, 각각의 토폴로지에 대해 동일한 실험을 진행하였다. 각 토폴로지에 대한 실험 결과도 마찬가지로 그림 4 와 같이 잘 나오는 것을 확인 할 수 있었다.

V. 결론 및 향후 연구

본 논문에서는 네트워크 토폴로지 탐색 분야에서 기존에 존재하는 여러 방법들을 분석하고 각 방법들의 문제점에 대해서 언급하였다. 그리고 문제점을 해결하기 위해서 SNMP 기반으로 네트워크 장비의 발생 트래픽 양을 비교하고, 이를 활용한 네트워크 토폴로지 탐색 방법을 제안하였다.

3 장 본론에서 제안하는 방법의 상세 알고리즘 제시하고, 4 장 실험에서는 제안하는 알고리즘을 실제 네트워크 장비에 적용하였다. 본 논문에서 제안하는 방법은 기존 방법의 여러 가지 문제점을 해결 가능 할 뿐만 아니라, 알고리즘이 비교적 쉽고 간편하기 때문에 수행 시간도 상당히 줄일 수 있었다.

하지만 본 논문에서 적용한 네트워크는 네 대의 스위치로, 소규모의 네트워크이기 때문에 타당성을 검증하기에 충분하지 않다. 그리고 측정 시간의 차이로 인하여 계산된 수치가 차이가 날 경우, 부정확한 토폴로지가 도출 될 수 있다.

따라서 향후 연구로 여러 네트워크에 적용하여 알고리즘을 보완할 예정이며, 관련 연구에서 언급한 LLDP, BDDP 등의 여러 방법들을 함께 연구하여, 제안하는 방법의 한계점을 해결 할 예정이다.

그리고 결과로 나온 csv 파일의 네트워크 장비 연결 구성을 데이터베이스에 저장하고, 오픈 소스 기반 네트워크 모니터링 플랫폼을 활용하여 각 장비의 연결 구성 시각화에 대한 연구를 진행 할 예정이다.

.

참 고 문 헌

[1] J.-S. Kim, H.-R Oh, “Efficient Method of Collecting Network Traces for Generating Network Topology”, Journal of KIISE, Vol. 45, No. 12, pp. 1319-1328, Dec. 2018.

[2] B. Donnet, T. Friedman. "Internet Topology Discovery: a Survey." IEEE Communications Surveys & Tutorials. Vol. 9, No. 4, pp, 56-69, 2017

[3] Ochoa Aday, Leonardo, Cristina Cervelló Pastor, and Adriana Fernández Fernández. "Current trends of topology discovery in OpenFlow-based software defined networks." (2015).

[4] Pandey, Suman, et al. "Ip network topology discovery using snmp." 2009 International Conference on Information Networking. IEEE, 2009.

[5] Lowekamp, Bruce, David O'Hallaron, and Thomas Gross. "Topology discovery for large ethernet networks." ACM SIGCOMM Computer Communication Review 31.4 (2001): 237-248.