https://doi.org/10.22670/knom.2020.23.1.1

네트워크 트래픽 입출량 분석을 통한 네트워크 토폴로지 탐색 시스템 설계

박 지 태◆, 백 의 준*, 신 무 곤*, 이 민 성*, 김 명 섭°

A Design of Network Topology Discovery System based on Traffic In-out Count Analysis

Ji-Tae Park◆, Ui-Jun Baek*, Mu-Gon Shin*, Min-Seong Lee*, Myung-Sup Kim°

요 약

최근 과학 기술의 급격한 발전에 따라 네트워크 시장 및 환경이 커지는 중이며, 막대한 양의 트래픽이 발생하고 있다. 특히 5G 네트워크, 엣지 컴퓨팅의 발전으로 이러한 현상은 더욱 가속화 될 전망이다. 하지만 이러한 추세에 따라 네트워크상에서 발생하는 트래픽 과부하, 악성 행위 등의 장애 현상도 빈번하게 발생하고 있다. 네트워크 관리자는 이러한 문제점을 해결하고, 초고속 네트워크를 구현하기 위해 네트워크 관리 시스템을 구축해야 하며, 네트워크 관리 시스템을 통해서 네트워크 장비들의 연결 구성에 대해 정확하게 알아야한다. 하지만 기존 네트워크 토폴로지 탐색 방법은 관리자가 수동으로 관리하기 때문에 비효율적이며, 시간 및 비용이 많이 든다는 문제점이 있다. 따라서 본 논문에서는 네트워크 트래픽의 입출량에 따른 네트워크 토폴로지 구성 방법을 제안한다. 제안하는 방법을 실제 네트워크 장비에 적용하여 본 논문의 타당성을 검증한다.

Key Words : Network Topology Discovery, Network Management System, SNMP

ABSTRACT

With the rapid development of science and technology in recent years, the network environment are growing, and a huge amount of traffic is generated. In particular, the development of 5G networks and edge computing will accelerate this phenomenon. However, according to these trends, network malicious behaviors and traffic overloads are also frequently occurring. To solve these problems, network administrators need to build a network management system to implement a high-speed network and should know exactly about the connection topology of network devices through the network management system. However, the existing network topology discovery method is inefficient because it is passively managed by an administrator and it is a time consuming task. Therefore, we proposes a method of network topology discovery according to the amount of in and out network traffic. The proposed method is applied to a real network to verify the validity of this paper.

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구이며, (No.2018-0-00539-003,블록체인의 트랜잭션 모니터링 및 분석 기술개발) 2020년도 산업통상자원부 및 한국산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임. (No. 20008902, IT비용 최소화를 위한 5채널 탐지기술 기반 SaaS SW Management Platform(SMP) 개발)

◆ Korea University Department of Computer and Information Science, pjj5846@korea.ac.kr
° Corresponding Author : Korea University Department of Computer and Information Science, tmskim@korea.ac.kr
* Department of Computer and Information Science, Korea University, {pb1069, tm0809, min0768}@korea.ac.kr
논문번호 : KNOM2020-01-02, Received July 12, 2020; Revised July 26, 2020; Accepted August 15, 2020

Ⅰ. 서 론

최근 과학 기술의 비약적인 발전과 초고속 네트워크의 등장으로 오늘 날의 네트워크 환경은 날이 갈수록 복잡해지고 커지고 있다. 특히 다양한 응용의 출현으로 대량의 트래픽이 발생하고 있으며, 5G 네트워크, 엣지 컴퓨팅 기술 개발로 인해 이러한 추세는 더욱 가속화 될 전망이다. 다양한 응용의 출현과 대량의 트래픽의 발생으로 인해 네트워크 트래픽 과부하, 악성 행위, 네트워크 품질 저하 등의 다양한 문제가 발생하고 있다. 이러한 문제들은 은행, 주식과 같은 경제 기관이나 국가 및 기업 내 중요한 기관이 대상 일 경우 큰 피해를 일으킨다.

이러한 문제점을 해결하기 위해서 네트워크 관리자는 지속적으로 네트워크 장비에 대해 모니터링 해야 한다. 네트워크 모니터링을 통해서 외부로부터 발생하는 비정상행위 뿐만 아니라 네트워크 트래픽 과부하와 같은 장애 현상에 대해서도 빠른 대응이 가능하다. 최근에는 여러 기업 혹은 기관에서 자사 네트워크를 효율적으로 관리하기 위한 네트워크 관리 시스템(Network Management System, NMS)을 사용하고 있다. 네트워크 관리자는 NMS를 통해 네트워크 자원 관리, 네트워크 장비 관리 및 여러 성능 정책 설정을 수행한다. 그리고 지속적인 네트워크 장비 상태 모니터링을 통해 시스템 성능에 영향을 주는 이벤트에 대한 경고 및 대응 기능을 수행한다.

효율적으로 네트워크 장비를 모니터링하기 위해서 관리자는 네트워크 장비의 토폴로지에 대해 정확하게 알아야한다. 예를 들어, 특정 서브 네트워크에 트래픽 과부하가 발생 할 경우, 네트워크 장비의 연결 구성에 따라서 해당 서브 네트워크에 대해 적절한 자원 관리 정책을 설정 할 수 있다.

하지만 네트워크 장비 수가 많아지면, 모든 장비의 연결에 대해 정확하게 파악하기 어렵다. 특히 기존의 네트워크 토폴로지 탐색하는 방법은 수동적으로 수행하기 때문에 비효율적이며, 많은 시간과 비용이 든다. 그리고 수동으로 네트워크 토폴로지 탐색을 진행 할 경우 네트워크 장비의 연결을 잘못 구성 할 가능성이 높으며, 잘못 구성 할 경우 위급한 상황에서 심각한 피해를 입을 수 있다. 따라서 효율적으로 네트워크 토폴로지 탐색을 할 수 있는 방법론에 대한 연구가 진행 중이다 [1-5].

네트워크 토폴로지 탐색 방법론으로 Traceroute, Ping, ARP 프로토콜을 이용하는 방법이 있다[1-3]. 하지만 기존 연구 방법론은 제한된 적용 범위, 낮은 토폴로지 탐색 정확도 등의 문제점이 있다.

따라서 본 논문에서는 네트워크 트래픽의 입출량에 따라서 네트워크 토폴로지 탐색하는 방안을 제안한다. 트래픽의 입출량 정보를 구하기 위해서 SNMP(Single Network Management Protocol)을 사용한다. SNMP 프로토콜을 활용하면 서버에서 네트워크 장비들의 여러 정보를 간단하게 알 수 있다는 장점이 있다.

본 논문의 구성은 다음과 같다. 1장 서론에서 네트워크 관리 시스템의 필요성과 기존 방법론의 문제점에 대해 언급하고, 2장에서 관련 연구에 대해 설명한다. 3장에서 제안하는 방법론을 설명하며, 4장에서 실제 네트워크 장비에 대해 제안하는 방법론을 적용하여 검증한다. 마지막으로 5장에서 결론 및 향후 연구를 언급하고 마친다.

Ⅱ. 관련 연구

네트워크 관리를 위해 구현 된 NMS는 공통적으로 몇 가지 이슈가 있다. 첫 번째로 장비들 간의 연결 구성을 수동으로 관리하기 때문에 주기적으로 장비들의 상태를 확인해야하며, 대규모 네트워크에서 매우 비효율적이다. 두 번째로 NMS에서 악성 행위 탐지, 트래픽 과부하 경고 등 여러 기능을 제공하지만, 네트워크 장비 간 토폴로지에 대해서는 제한된 정보만 제공한다. 따라서 관리자는 ping을 통해 네트워크 장비의 상태에 대해 기본적인 점검을 진행하고 있다. 하지만 최근에는 외부 방화벽 설정으로 막아두는 경우가 있기 때문에 ping만으로 네트워크 장비 상태를 확인하기 어렵다.

이러한 이슈를 해결하기 위해 다양한 연구가 진행되어 왔다. 먼저 Traceroute를 활용한 방법은 두 호스트 사이의 경로를 탐색하는 기법으로, 패킷 헤더의 TTL(Time-To-Live) 값을 조정하여 경로를 탐색하는 방법이다[6,7]. 이를 활용하면 두 대의 네트워크 장비 사이의 경로에 존재하는 모든 네트워크 장비들을 탐색 할 수 있다. 하지만 대규모의 네트워크에서 모든 장비에 대해 Traceroute를 적용 할 경우, 많은 시간과 트래픽이 발생하며, 경로 추적 과정 중 중복 탐색 문제가 발생할 수 있다. 대규모의 네트워크에서 중복 탐색이 발생하면 매우 비효율적이며,

처리하기 위해 많은 시간과 비용이 들게 된다.

중복 탐색 현상을 해결하기 위해 더블 트리 알고리즘(Double Tree Algorithm)이 제안되었다[7,8]. 더블 트리 알고리즘은 출발지 중심 트리와 목적지 중심 트리 구조를 이용하여 경로 위의 중복 지점을 효과적으로 탐지한다. 하지만 더블 트리 알고리즘은 트래픽의 과부하와 데이터베이스 오버헤드를 발생시킬 수 있다.

SNMP를 이용하는 방법은 네트워크 장비의 MIB 정보를 수집하여 네트워크 토폴로지를 구성한다[9-11]. [9]에서는 L2, L3 계층의 장비만 탐색하며, SNMP와 ICMP를 사용한다. 네트워크 노드에 ICMP 패킷을 보내고, 제안한 방법을 통해 노드로부터 AFT(Address Forwarding Table)을 만들어서 네트워크 토폴로지를 구성한다. 이 방법은 적당한 양의 트래픽을 발생시키며, 효율적이지만 AFT를 생성하는 데 많은 시간이 걸린다는 단점이 있다. 그리고 제한된 네트워크 장비에 대해서만 적용 가능하며, 다수의 ICMP 패킷을 생성하여 보내기 때문에 트래픽 과부하가 발생 할 수 있다.

[10]에서는 네트워크 장비의 MIB 정보를 수집하여 데이터베이스에 저장하고, 저장된 MIB 정보를 활용하여 토폴로지를 구성한다. 이 방법은 L2, L3 뿐만 아니라 L4, L7 스위치까지 탐색 가능하여서 다양한 네트워크에 적용 가능하다는 장점이 있다. 또한, 트래픽 과부하가 적으며 관리자 입장에서 쉽게 사용 할 수 있다. 하지만 장비 탐색 할 때 사용하는 ipRouteTable과 연결 구성을 만들 때 사용하는 BRIDGE-MIB는 네트워크 장비 중 지원하지 않는 경우도 존재한다. 따라서 BRIDGE-MIB가 없을 경우 장비 간 연결 구성을 추론 할 수 없다.

[11]에서는 [10]과 마찬가지로 SNMP와 HTML5 기반의 TWaver를 이용하여 웹 기반 토폴로지를 구현한다. 하지만 [4]에서는 NMS 시스템에서 보여주는 UI 및 데이터베이스 구조를 어떻게 설계하였는지에 중점을 맞추고 있기 때문에 어떻게 SNMP를 이용하여 토폴로지를 구성하는지에 대해서는 언급이 되어있지 않다.

기존 연구들 중 [10]의 방법이 관리자가 사용하기에 편리하고 성능이 좋지만, [9],[10] 모두 적용 가능한 네트워크가 제한되어 있다는 단점이 있다. 따라서 본 논문에서는 [10]의 방법을 개선하여 네트워크 장비의 트래픽의 입출량을 비교하여 네트워크 토폴로지 구성 방법에 대해 제안한다.

Ⅲ. 본 론

1. 전체 시스템 구조

본 장에서는 제안하는 시스템의 구조와 세부 알고리즘을 소개한다. 전체적인 구조는 네트워크 장비 탐색, 연결 토폴로지 탐색으로 구성되어 있으며, 그림 1에 나타나 있다.

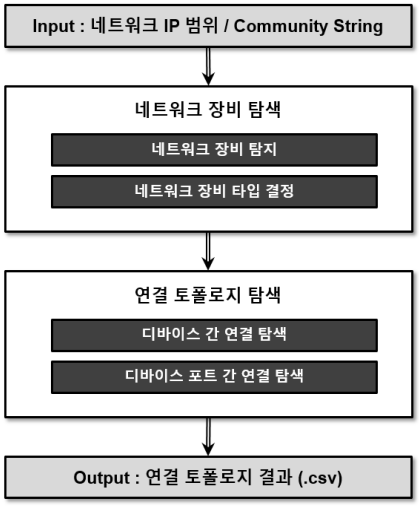

그림 1. 전체 시스템 구조
Fig. 1. Entire System Structure

1.1. 네트워크 장비 탐색

먼저 입력으로 대상 네트워크의 네트워크 IP 범위와 Community String이 들어온다. 네트워크 장비 탐색 모듈은 네트워크 장비 탐지, 장비 타입 결정의 세부 모듈로 구성되어 있다.

네트워크 장비 탐지 모듈에서 대상 네트워크 IP 범위 내의 모든 IP에 SNMP 프로토콜의 SNMP GET 메시지를 활용하여 장비 타입 및 연결 토폴로지 탐색에 필요한 MIB 정보를 저장한다. 이때, MIB 정보를 저장하는 장비는 SNMP를 지원하는 장비를 대상으로 하기 때문에 SNMP를 지원하지 않거나 유효하지 않은 IP의 경우 데이터를 저장하지 않는다. 예를 들어, 10.1.1.1부터 10.1.1.50의 IP 범위 내에 10.1.1.6, 10.1.1.10의 네트워크 장비가 있다고 가정하면, 해당 IP 범위 내에서 SNMP를 지원하는 장비인 10.1.1.6, 10.1.1.10의 MIB 정보만 저장된다.

하지만 SNMP를 지원하지 않거나 유효하지 않은 IP일 경우 특정 시간 동안 대기하다가 Timeout 메시지가 출력된다. 이때, 네트워크 IP 범위가 클 경우 시간이 상당히 많이 소모될 수 있다. 따라서 제안하는 시스템에서는 SNMP GET 메시지를 보낼 때, 병렬로 진행 할 수 있게 하여 이러한 문제점을 해결하였다. 네트워크 장비 탐색 모듈에서 저장하는 MIB 정보는 다음과 같다. 이때, ifInOctet (10 sec), ifOutOctet (10 sec)은 ifInOctet, ifOutOctet의 정보를 저장하고 10초 후의 트래픽 양을 저장한 것이다. 두 장비가 서로 연결되어있다고 하더라도, 트래픽 입출량이 반드시 똑같지 않을 수 있기 때문에, 10초간 포트에서 발생한 트래픽 량을 비교하기 위해 ifInOctet (10 sec) 정보를 저장한다. 즉, 특정 장비의 포트 별 10초간 트래픽 발생량을 비교하여 연결을 탐지한다.

표 1. 저장 되는 네트워크 장비 MIB 정보
Table 1. Stored MIB Information of Network Device

사용 MIB 정보	설명	상위 MIB 객체	모듈
sysServices	장비 시스템 정보 (OS, Hardware ..)	system	네트워크 장비 탐색
ipNetToMediaNetAddress	ARP 테이블 내 연결 장비 IP	IP	연결 토폴로지 탐색
ipNetToMediaType	ARP 테이블 내 연결 타입		
ifInOctet	장비로 들어오는 트래픽 양	interface	
ifOutOctet	장비에서 발생하는 트래픽 양		
ifInOctet (10 sec)	측정 시간으로부터 10초 후 장비로 들어오는 트래픽 양		
ifOutOctet (10 sec)	측정 시간으로부터 10초 후 장비로 들어오는 트래픽 양		
ifName	인터페이스 정보	ifMIB	

네트워크 장비 타입 결정 모듈은 [10]에서 제안하는 방법과 동일하다. 네트워크 장비 정보 저장 모듈에서 저장된 정보 중 sysServices를 활용하여 네트워크 장비의 타입을 결정한다. sysServices는 네트워크 장비의 타입을 10진수로 표현한 정수 값이다. sysServices을 7 비트의 문자열로 변환하고, 각 비트는 OSI 7 계층에 대응된다. 예를 들어 sysServices가 6일 경우, 2 진수로 0000110에 해당하기 때문에 2, 3계층을 지원하는 장비이다[10]. 네트워크 장비 정보 저장과 타입 탐색 모듈의 결과로 SNMP를 지원하는 장비의 IP, MIB 정보와 장비의 타입을 알 수 있다.

1.2. 연결 토폴로지 탐색

연결 토폴로지 탐색 모듈은 각 장비들의 연결 정보를 탐색한다. 기존 연구의 방법은 장비 타입 탐색 모듈에서 정의된 장비의 타입에 따라 연결 정보가 다르게 저장 된다. 예를 들어 2 계층 장비 간의 연결, 3계층 장비 간의 연결에서 적용하는 알고리즘과 사용하는 MIB 정보가 다르다. 이때, 3 계층 간의 연결 정보를 알기 위해서 ipRouteNextHop 정보와 2 계층의 연결 정보를 알기 위해서 BRIDGE-MIB 정보가 필요하다. 하지만 2절에서 설명한대로 ipRouteNextHop와 BRIDGE-MIB 정보가 존재하지 않는 경우가 존재하기 때문에 네트워크 장비 간의 연결 정보를 알 수 없다.

제안하는 연결 토폴로지 탐색 모듈은 크게 디바이스 간 연결 탐색, 디바이스 포트 연결 탐색의 세부 모듈로 구성되어 있다. 디바이스 간 연결 탐색 모듈에서 네트워크 장비 간 연결을 탐색하고, 탐색한 연결을 기준으로 연결된 포트를 탐색한다.

먼저 이전 모듈에서 저장 된 ipNetToMediaTable의 MIB 정보를 활용하여 네트워크 장비 간 연결을 탐색한다. ipNetToMediaTable은 ARP 테이블에 해당하는 MIB 정보로 연결된 네트워크 장비의 물리주소, IP 주소, 타입 등을 알 수 있다. 디바이스 연결 탐색의 세부 알고리즘은 2절에서 설명한다.

interface MIB Object의 포트별 트래픽 입출량을 사용한다. 여기서 사용하는 MIB 정보로 ifInOctet과 ifOutOctet가 있는데, ifInOctet는 해당 장비의 포트별 들어오는 트래픽 양이며, ifOutOctet은 해당 장비의 포트별 나가는 트래픽 양이다. 두 가지 MIB 정보를 비교하여 각 장비 별 연결 토폴로지를 얻을 수 있다. 디바이스 포트 간 연결 탐색 알고리즘은 2절에서 설명한다.

제안하는 방법의 모든 모듈을 지나면 각 장비의 연결 토폴로지 정보가 나온다. 이때, 연결 토폴로지 정보는 디바이스 간의 연결뿐만 아니라 포트 연결 정보까지 나온다. 연결 토폴로지 정보는 csv 파일로 저장되며, 장비의 IP, 타입, 연결 포트가 저장된다.

2. 연결 토폴로지 탐색 알고리즘

본 장에서는 연결 토폴로지 탐색 알고리즘에 관해 설명한다. 먼저 연결 토폴로지 탐색 알고리즘은 세부 모듈 별로 구성되어 있다.

Algorithm 1 : Device Connection Discovery
Input : List of Device IP, MIB Information of Each Devices
Output : Device Connection Information
Notation – D : List of Device IP / I : 'ipNetToMediaType' MIB Information
IP : 'ipNetToMediaNetAddress' MIB Information

```
for each pair of D [D_i, D_j]
  Check if the Connection Information Exists
  if Connection does not exist in [D_i, D_j]
    Check the 'ipNetToMediaTable MIB Information' of [D_i, D_j]
    Compare the 'ipNetToMediaType MIB Information' of [D_i, D_j]
    if I_i == "static"
      Get the Mapping IP List of I_i [IP_i]
      if I_j == "dynamic"
        Get the Mapping IP List of I_j [IP_j]
        if Corresponding IP dose exist in [IP_i, IP_j]
          define the Connection of [D_i, D_j]
          store the Connection Information of [D_i, D_j]
        else
          break
  else            // Connection Exists in D_i and D_j
    Get the another pair of D
```

그림 2. 장비 연결 탐색 알고리즘
Fig. 2. An Algorithm of Device Connection Discovery

먼저 장비 연결 탐색 알고리즘은 이전 모듈에서 저장된 장비별 MIB 정보를 활용하여 장비 간의 연결을 탐색한다. 장비 간 연결이 정의되면, 장비 포트 연결 탐색 알고리즘을 통해서 연결된 장비가 어떤 포트와 연결되어있는지 탐색한다. 장비 연결 탐색 알고리즘에서는 표1에서 ipNetToMediaTable의 하위 MIB 정보를 사용하며, 그림 2에 나타나 있다.

먼저 장비 간 연결 쌍을 A, B라고 하고, 저장된 연결 정보가 있는지 확인한다. 연결이 없을 경우 두 장비의 ipNetToMediaTable MIB 정보를 확인한다. A의 ipNetToMediaType이 static에 해당하는 IP 리스트와, B의 ipNetToMediaType이 dynamic에 해당하는 IP리스트를 구한다. 두 IP 리스트 중 공통으로 포함하는 IP가 있을 경우 두 장비는 연결되었다고 정의한다. 이 단계에서는 장비 간 모든 연결을 탐색하기 때문에, 직접 연결되어있지 않아도 연결 되어 있다고 정의한다. 예를 들어 네트워크 장비 A, B, C, D 중 A, B, C가 서로 일렬로 연결되어있고, D는 어느 장비와도 연결 되어있지 않다고 가정 할 때, 장비 연결 탐색 알고리즘을 통해 A-B, A-C, B-C의 연결이 탐색 된다. 이때, A-C는 직접 연결되어있지 않지만 정의된 연결이다. 이후, 장비 포트 연결 탐색 알고리즘에서 직접 연결된 장비만 선별한다.

```
[                    ]# 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
```

그림 3. ifOutOctet MIB 정보 예시
Fig. 3. An Example of ifOutOctet MIB Information

장비 포트 연결 탐색 알고리즘은 두 장비의 트래픽 입출량에 따라 정의된 장비 간 연결 중 직접 연결된 장비의 포트 연결 정보를 탐색한다.

예를 들어 정상적인 네트워크 내에서 두 대의 장비 A, B가 서로 10, 11번 포트로 연결되어있다면, A의 10번 포트에 들어오는 트래픽 양과 B의 11번 포트에서 나오는 트래픽양이 같다. 즉, 장비 A의 들어오는 트래픽 양과 장비 B의 나오는 트래픽 양이 같을 경우, 두 장비는 연결 되어있다고 정의한다.

또한, 나오고 들어오는 양이 같은 트래픽이 어떤 포트인지 알 수 있기 때문에 두 장비 간의 인터페이스 연결 정보도 쉽게 알 수 있다. ifOutOctet의 MIB 정보에 대한 예시는 그림 3에 나타나 있다. 그림 3에서 14, 21, 28번 인덱스에서 각각 트래픽이 발생하고 있으며, 다른 장비의 ifInOctet과 일치할 경우 인덱스와 맵핑되는 포트를 찾는다. 맵핑되는 포트를 찾을 대 표 1의 ifName MIB 정보를 활용하면 쉽게 찾을 수 있다.

하지만 네트워크 트래픽 입출량 비교를 적용하기 위해서는 네트워크 장비 정보 저장 모듈에서 ifInOctet과 ifOutOctet가 저장되는 시간이 동일해야 한다. 하지만 네트워크 장비 수가 많거나 특정 상황에 의해서 저장되는 시간이 다를 경우가 있어서, 장비 포트 탐색 알고리즘에서는 10초 간 발생한 트래픽량을 비교한다. 여기서 10초는 사용자가 따로 지정이 가능하지만, 실제 네트워크에서 만약 10초 이내 발생한 트래픽 양을 비교할 경우, 연결을 잘못

Algorithm 2 :Device Port Connection Discovery
Input : List of Device IP, MIB Information of Each Devices, Connection Information
Output : Total Connection Information (Connection / Port)
Notation – D : List of Device IP / In : Interface Information
In : ifInOctet of Device / Out : ifOutOctet of Device / Diff : Difference of IfInOctet (10 sec) and IfInOctet or IfOutOctet (10 sec) and IfOutOctet

1. Get the IP List of the Devices
2. Set the 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 ifInOctet (after 10 sec) of D_i
6. Load ifOutOuctet of D_j and ifOutOctet (after 10 sec) of D_j
7. Get the Difference of IfInOctet (10 sec) and IfInOctet of D_i // Define as $Diff_i$
8. Get the Difference of IfOutOctet (10 sec) and IfOutOctet of D_j // Define as $Diff_j$
9. if $Diff_i$ == $Diff_j$
10. Get the Octet 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

그림 4. 장비 포트 탐색 알고리즘
Fig. 4. An Algorithm of Device Port Connection Discovery

탐지하는 경우가 발생하며, 10초 이후 발생한 트래픽 양을 비교할 경우, 발생한 트래픽의 미세한 차이로 인해 연결을 탐지하지 못하는 경우가 발생한다. 따라서 제안하는 시스템에서는 10초를 기준으로 발생 트래픽 양을 비교하여 연결을 탐지한다. 장비 포트 탐색 알고리즘에 대한 설명은 그림 4에 나타나 있다.

먼저 입력으로 SNMP를 지원하는 장비의 IP, 저장된 MIB 정보가 들어간다. 연결된 장비 쌍을 A, B라고 정의하고, 두 장비에 대한 ifInOctet, ifOutOctet, ifInOctet (10 sec), ifOutOctet (10 sec)를 로드한다. 다음으로 로드 된 A의 ifInOctet, ifInOctet (10 sec)의 차이와 B의 ifOutOctet, ifOutOctet (10 sec)의 차이를 구하고, 해당 리스트를 저장한다. 저장된 차이 값들에 대해서 서로 비교를 하며, 만약 값이 같을 경우 두 장비는 각각의 ifInOctet, ifOutOctet에 해당하는 인덱스를 찾는다. 이때, 장비 연결 알고리즘에서 탐색 된 직접 연결되지 않은 장비 간 연결의 경우, 두 장비 간 트래픽 입출량이 다르므로 비교하는 과정에서 제외하게 된다. 마지막으로 각 인덱스와 맵핑되는 포트 정보를 ifName MIB 정보를 통해 찾고 해당 연결 포트 정보를 저장한다.

두 가지 세부 모듈을 거치면 최종적으로 연결 토폴로지 정보가 나온다. 연결 토폴로지 정보는 장비의 IP, 연결 포트, 타입의 정보로 구성되어 있으며, csv 파일로 저장된다. 하나의 행에 두 장비 간의 하나의 연결 정보가 저장되며, 저장된 csv 파일은 데이터베이스를 활용하여 네트워크 내 장비 간의 토폴로지를 UI로 나타낼 수 있다.

제안하는 알고리즘은 세 가지 장점이 존재한다. 먼저 기존의 SNMP 기반 방법뿐만 아니라 다른 방법론에 비해 쉽고 간편하다. 장비의 포트별 나오고 들어오는 트래픽 양만 비교하기 때문에, 사용하는 MIB 정보도 적고, 시간도 적게 걸린다. 두 번째로 장비의 타입에 상관없이 적용할 수 있다. L2 스위치, L3 스위치, 라우터 등의 네트워크 장비 타입에 상관없이 트래픽 양만 비교하면 되기 때문에 동일한 알고리즘을 여러 타입의 장비에 동일하게 적용 가능하다. 세 번째로 문제점으로 언급되었던 BRIDGE-MIB과 ipRouteNextHop의 MIB 정보가 없어도 적용 가능하다. 따라서 제안하는 방법은 기존 연구의 여러 가지 문제점을 해결 가능 할 뿐만 아니라 대부분의 네트워크 장비가 SNMP를 지원하기 때문에 더 쉽고 널리 사용 가능하다는 장점이 있다.

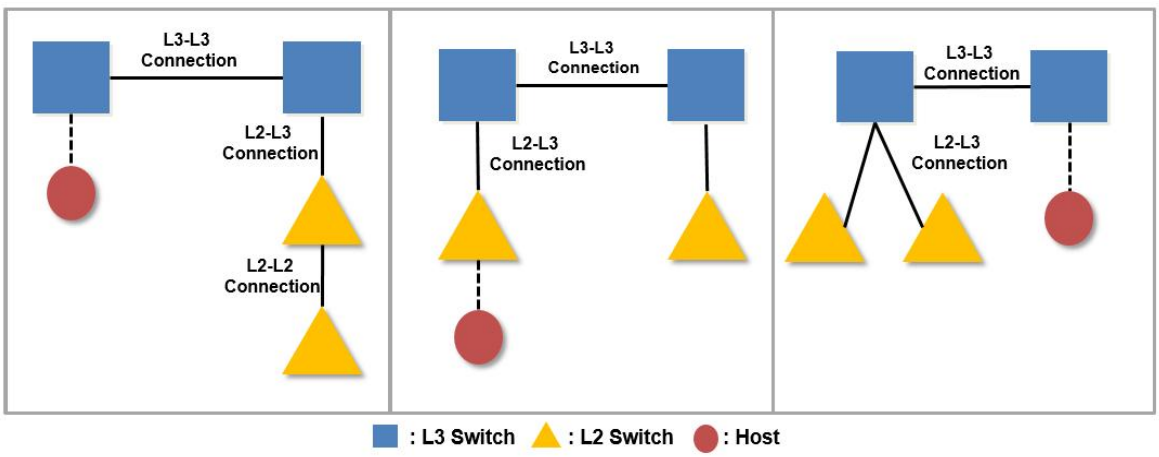

그림 5. 실험 환경 구성도
Fig. 5. A Structure of Experiment Environment

```
---------------- L2 to L2 Connection -----------------

192.168.20.10 & 192.168.20.20 Connected!
L2 Switch_1[192.168.20.10] - 14 / L2 Switch_2[192.168.20.20] - 8 / OUCTET DIFF : 0 [130851734 - 130851734]
 192.168.20.10 (Fa0/14)  192.168.20.20  (Fa0/7) is Connected

-----------------------------------------------------

---------------- L2 to L3 Connection -----------------

192.168.20.10 & 192.168.20.1 Connected!
L2 Switch[192.168.20.10] - 21 / L3 Switch[192.168.20.1] - 10016 / OUCTET DIFF : 0 [46321930 - 46321930]
 192.168.20.10 (Fa0/21)  192.168.20.1  (Fa0/16) is Connected

-----------------------------------------------------

---------------- L3 to L3 Connection -----------------

192.168.10.1 & 192.168.20.1 Connected!
L3 Switch_1[192.168.10.1] - 11 / L3 Switch_2[192.168.20.1] - 10011 / OUCTET DIFF : 0 [50198307 - 50198307]
 192.168.10.1 (Fa0/11)  192.168.20.1  (Fa0/11) is Connected
```

그림 6. 연결 토폴로지 탐색 실험 결과
Fig. 6. Experiment Result of Connection Topology Discovery

Ⅳ. 실험 및 평가

1. 실험 환경

본 연구에서 제안한 네트워크 장비의 연결 토폴로지를 탐색 방법을 검증하기 위해 실제 네트워크 장비에 대해 실험을 진행하였다. 실험 환경은 그림 5에 나타나 있다. 실험에 사용한 네트워크 장비는 L2 스위치 2대, L3 스위치 2대이다. 연결 구성은 총 3가지로 L2 스위치 간의 연결, L2, L3 스위치 간의 연결, L3 스위치 간의 연결로 구성되어 있다. 각 타입별 연결 구성을 정확하게 탐색하는지 검증하기 위해 그림 5와 같이 구성하였다. 네 대의 스위치를 활용하여 구성 가능한 연결은 세 가지가 있으며, 모든 환경에 대해 실험을 진행하였다.

그림 5에서 호스트는 제안하는 시스템이 설치되어있는 서버이며, 네트워크 장비가 아니기 때문에 연결 탐지가 되지 않는다. 하지만 호스트와 대상 네트워크의 L3 스위치와 연결되어있기 때문에 SNMP 메시지를 보내어 각 장비들의 정보 및 연결 정보를 저장 할 수 있다. 호스트는 Linux CentOS 7의 환경이며, 호스트에 설치된 시스템은 쉘 스크립트와 C/C++ 언어로 구현하였다.

2. 실험 결과

제안하는 방법을 검증하기 위해 구성한 실험 환경에서 네트워크 토폴로지 탐색 실험을 진행하였다. 실험 결과는 그림 6에 나타나 있다.

그림 6은 그림 5의 실험 환경에 맞게 네트워크를 구성하였으며, 그림 5에서 첫 번째 네트워크에 해당하는 연결 구성을 출력한 것이다. 먼저 두 장비의 입출량 트래픽을 비교하여 같을 경우 연결되었다고 탐색하고, 해당 연결 포트를 찾아서 트래픽 입출량과 함께 출력한다. 예를 들어 실험에 사용 한 두 대의 L2 스위치를 A, B라고 할 때, A의 IP는 192.168.20.10이며, B의 IP는 192.168.20.20이다. 그리고 A와 B의 10초간 트래픽 입출량은 130,851,734 바이트로 같기 때문에 연결되어있다고 정의한다. 언급한 예시는 그림 6에서 L2 to L2 Connection에 해당하며, 연결이 정의될 경우 맵핑된 인덱스를 통해 연결되어있는 포트를 찾는다. 그림 5의 모든 토폴로지에 대하여 동일한 실험을 진행하였으며, 실험 결과로 세 가지 토폴로지에 대하여 모든 연결 정보를 정확하게 도출하였다.

제안하는 방법의 타당성을 검증하기 위해서 연결 토폴로지를 탐색하는 데 걸리는 시간을 측정하였다. 탐색 실험은 두 방법론이 같게 30번 수행하였으며, 30번 수행 중 가장 오래 걸리는 시간과 평균 시간을 측정하였다. 그림 1의 연결 토폴로지 탐색 모듈 수행 시간만 측정하여 기존 연구와 제안하는 방법과 실험 시간을 비교하였으며, 결과는 표 2에 나타나 있다.

표 2. 수행 시간 및 탐색 결과 비교
Table 2. Comparison of the Performance Time and Detection Result

방법론		최대 수행 시간	평균 수행 시간	탐색 결과
IP	Network	28 sec	12 sec	모든 연결 정보

Topology Discovery Using SNMP[10]			정확하게 탐색
Proposed Method	1.2 sec	0.8 sec	모든 연결 정보 정확하게 탐색

두 가지 방법 모두 정확하게 연결 정보를 탐색하였다. 하지만 수행 시간을 비교할 경우, 기존 연구 방법은 최대 28초가 걸렸으며, 제안한 방법은 1초 내로 모든 연결 토폴로지 정보를 탐색 할 수 있었다. 만약 실험에 사용한 네트워크 장비 수가 여러 대 있으면, 두 방법론의 수행 시간 차이는 더 급격하게 차이 날 것이다. 따라서 실험을 통해 제안하는 방법의 알고리즘을 적용하면 수행 시간을 줄일 수 있다는 것을 알 수 있다.

Ⅴ. 결 론

본 논문에서는 1장 서론에서 네트워크 관리의 필요성과 NMS에 대해 소개하였으며, 효율적인 네트워크 관리를 위한 네트워크 토폴로지 탐색 필요성을 언급하였다. 네트워크 토폴로지 탐색에 관한 기존 연구를 소개하고, 기존 연구의 문제점을 언급하여 본 연구의 필요성을 제시하였다. 기존 연구의 문제점을 해결하기 위해서 네트워크 장비의 포트별 트래픽 입출량 비교를 통한 연결 토폴로지 구성 방법론을 제안하였다. 3장 본론에서 제안하는 방법의 전체 구조 및 알고리즘에 관해 설명하였고, 4장 실험 및 평가에서 실제 네트워크 장비를 대상으로 검증 실험을 진행하였다.

실험 결과로 네트워크 장비 간의 연결과 장비 인터페이스 간의 연결까지 정확하게 탐지 할 수 있었다. 또한, 연결 토폴로지 탐지 시간을 1초 이내로 줄일 수 있기 때문에 전체 시스템 동작 시간을 크게 단축시킬 수 있었다.

하지만 제안하는 방법은 몇 가지 한계점이 존재한다. 먼저, 3.2장에서 언급한 대로 저장되는 시간이 다를 경우 연결 정보를 구성할 수 없다. 현재 구현한 시스템상에서는 이러한 문제점이 보이지 않았지만, 대규모의 네트워크에서는 MIB 정보를 저장하는 시간이 차이가 날 수 있다. 또한, 실제로 구현 가능한 모든 네트워크를 대상을 실험을 진행하였지만, 실험에 사용한 네 대의 스위치만으로 제안하는 방법의 타당성을 검증하기 충분하지 않다. 그리고 [10]에서 구현되어있는 VLAN에 대해서는 연결 정보를 제공하지 못한다. 또한, SNMP를 지원하지 않는 네트워크에는 적용 할 수 없는 문제점이 있다.

그럼에도 불구하고 제안하는 방법은 기존 연구 방법에 존재하는 여러 가지 문제점을 해결하였다는 점에 타당성을 입증하였다. 네트워크 트래픽 입출량을 비교하여 수행 시간을 크게 단축시켰으며, 네트워크 대부분에서는 SNMP를 지원하고 있기 때문에, 다른 방법론에 비해 다방면으로 활용할 수 있다. 또한, ipRouteNextHop, BRIDGE-MIB의 MIB 정보 없이 네트워크 장비들의 연결 정보를 탐색 가능하여서 기존 SNMP 기반 방법론의 문제점을 해결 할 수 있다.

향후 연구로 대규모의 네트워크 혹은 여러 종류의 네트워크에서도 적용 할 수 있도록 알고리즘을 수정 보완할 예정이다. 그리고 실험에 사용한 네트워크보다 더 다양하고, 복잡한 대규모 네트워크에 대해 실험을 진행 할 예정이다. 그리고 SNMP를 지원하지 않는 네트워크에도 적용할 수 있도록 LLDP와 같은 다른 방법론을 추가로 적용하여 개선할 예정이다.

References

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

[2] M, Reza, R. Rejaie and W. Willinger. "A Survey of Techniques for Internet Topology Discovery." IEEE Communications Surveys & Tutorials 17.2, Vol. 17, No. 2, pp. 1044-1065, 2015.

[3] B. Donnet, P. Raoult, T. Friedman, M. Crovella, "Efficient Algorithms for Large-Scale Topology Discovery", Proc. of 2005 ACM SIGMETRICS, pp.327-338, 2005.

[4] J. Yin, Y. Li, Q. Wang, B. Ji and J. Wang, "SNMP-based network topology discovery algorithm and implementation," 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, Sichuan, 2012, pp. 2241-2244.

[5] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," IETF, Internet RFC-1157, May 1990.

[6] M.-H. Gunes and K. Sarac, ″Resolving IP Aliases in Building Traceroute-Based Internet Maps,″ IEEE/ACM Transactions on Networking, Vol. 17, No. 6, pp. 1738-1751, 2009.
[7] 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.
[8] J.-S. Kim, H.-R Oh, “Network Topology Discovery with Load Balancing for IoT Environment”, Journal of KIISE, Vol. 44, No. 10, pp. 1071-1080, Oct. 2017.
[9] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz “Topology Discovery in Heterogeneous IP Networks: The NetInventory System,” IEEE/ACM Transactions on Networking. Vol. 12, No. 3, pp. 401-414, June. 2004
[10] S. Pandey, M.-J. Choi, Y.-J Won, W.-K. Hong ″SNMP-based enterprise IP network topology discovery.″ International Journal of Network Management, Vol.21, No.3, pp. 169-184, May 2011.
[11] K.-C. Shim, G.-H Hwang, “Network Topology Automatic Configuration and Remote Fault Diagnosis System”, Journal of the Korea Institute of Information and Communication Engineering, Vol. 22, No. 3, pp. 548-556, Mar. 2018.

박 지 태 (Jee-Tae Park)

2017 고려대학교 컴퓨터정보학과 학사
2017년 – 현재 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

백 의 준 (Ui-Jun Baek)

2018 고려대학교 컴퓨터정보학과 학사
2018년 – 현재 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

신 무 곤 (Mu-Gon Shin)

2019년 고려대학교 컴퓨터정보학과 학사
2019년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

이 민 성 (Min-Seong Lee)

2019년 고려대학교 컴퓨터정보학과 학사

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 학사
2000년 : 포항공과대학교 전자계산학과 석사
2004년 : 포항공과대학교 전자계산 학과 박사
2006년 : Dept. of ECS, Univ of Toronto Canada
2006년~현재 : 고려대학교 컴퓨터정보학과 교수
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크