JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 9 : 1655~1665 Sep. 2017

오픈플로우 기반의 과학실험데이터센터 네트워크의 성능 향상을 위한 스케줄링 알고리즘

공정욱[1*] · 민석홍[2] · 이재용[3] · 김병철[3]

A Scheduling Algorithm for Performance Enhancement of Science Data Center Network based on OpenFlow

Jong Uk Kong[1*] · Seok Hong Min[2] · Jae Yong Lee[3] · Byung Chul Kim[3]

[1*]National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, 34141, Korea
[2]Min Data Corporation, Sejong 30088, Korea
[3]Department of Radio and Information Communications Engineering, Chungnam National University, Daejeon 34134, Korea

요 약

최근 많은 클라우드 서비스 제공자, 기업, 연구소 등에서 데이터센터를 활발히 구축하고 있다. 일반적으로 데이터센터는 부하 분산을 위해 ECMP 데이터 포워딩 기법을 사용하여 트리 토폴로지 형태로 구축된다. 본 논문에서는 트리 토폴로지와 팻트리 토폴로지를 살펴보고, 또한 MLAG와 ECMP 같은 부하 분산 기술을 알아본다. 그리고 데이터센터내의 호스트에 저장되어 있는 특정 파일을 데이터센터 외부로 효율적으로 송신할 수 있는 스케줄링 알고리즘을 제안한다. 제안된 알고리즘은 팻트리 토폴로지와 오픈플로우 프로토콜을 이용한다. 수치해석을 통해 성능 분석을 수행하며, ECMP의 성능과 비교한다. 이러한 성능 비교를 통해 평균처리율과 파일전송완료시간에 있어서 제안된 알고리즘의 성능이 우수함을 보인다.

ABSTRACT

Recently data centers are being constructed actively by many cloud service providers, enterprises, research institutes, etc. Generally, they are built on tree topology using ECMP data forwarding scheme for load balancing. In this paper, we examine data center network topologies like tree topology and fat-tree topology, and load balancing technologies like MLAG and ECMP. Then, we propose a scheduling algorithm to efficiently transmit particular files stored on the hosts in the data center to the destination node outside the data center, where fat-tree topology and OpenFlow protocol between infrastructure layer and control layer are used. We run performance analysis by numerical method, and compare the analysis results with those of ECMP. Through the performance comparison, we show the outperformance of the proposed algorithm in terms of throughput and file transfer completion time.

키워드 : 데이터센터 네트워크, 오픈플로우, 부하 분산, 스케줄링 알고리즘

Key word : Data Center Network, OpenFlow, Load Balancing, Scheduling Algorithm

Received 26 July 2017, **Revised** 28 August 2017, **Accepted** 10 September 2017

* **Corresponding Author** Jong Uk Kong(E-mail:kju@kisti.re.kr, Tel:+82-42-869-0562)
National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Daejeon, 34141, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.9.1655 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

클라우드 컴퓨팅 기술과 네트워킹 기술의 발달로 클라우드 서비스 공급자나 기업 등에서 신규 데이터센터를 구축하는 사례가 증가하고 있다. 또한 기존의 조직 내에서 운영하던 컴퓨팅 관련 시설을 클라우드 호스팅 업체로 옮기고 있는 추세이다. 데이터센터는 규모의 경제를 제공하고 있으며, 인프라를 공유할 수 있는 장점이 있다. 이러한 데이터센터는 Google, Facebook 등과 같이 자신들이 운영하는 사설 센터(private center)와 Amazon EC2, Microsoft Azure, Rackspace와 같은 공개 센터(public center)로 나눌 수 있다. Dropbox, foursquare, Pinterest 등은 Amazon EC2의 클라우드 컴퓨팅 시설을 이용하고 있다.

과학기술분야에서도 e-Science 응용을 위한 많은 과학실험데이터센터들이 있다. 유럽입자물리학연구소(Conseil Européenne pour la Recherche Nucléaire, CERN)[1]와 유럽VLBI공동연구소(Joint Institute for VLBI ERIC, JIVE)[2], 미국의 페르미연구소(Fermilab)[3], 일본의 고에너지가속기연구기구(KEK)[4] 등이 과학실험데이터센터를 보유하고 있다. 이런 데이터센터들은 자체에서 생산된 실험 데이터를 원시 형태 또는 후처리하여 분배하거나, 여러 연구기관으로부터 수집한 데이터를 후처리하여 분배하는 역할을 수행하여 전 세계적으로 실험데이터를 공동 활용하는데 기여하고 있다.

한국과학기술정보연구원(Korea Institute of Science and Technology Information, KISTI)에서도 국내 응용연구자들이 개별적으로 시스템을 구축하고 네트워크를 연결하는 등의 시간, 비용 및 운영의 어려움을 해결하고자 슈퍼컴퓨터와 국가과학기술연구망을 직접 운영중인 장점을 살려 연구자를 위해 글로벌 사이언스 실험 데이터 허브 센터(Global Science Experimental Data Hub Center, GSDC)를 운영하고 있다. GSDC는 CERN의 CMS(Compact Muon Solenoid) 및 ALICE(A Large Ion Collider Experiment) 실험 데이터, KEK의 Belle 실험데이터, Fermilab의 입자물리 실험데이터, LIGO (Laser Interferometer Gravitational Observatory)의 중력파 관측 데이터, 국내 기상기후 데이터 등 많은 원시 데이터 및 후처리 데이터를 보유하고 있다. 국내외 연구자들은 자신이 필요로 하는 데이터를 GSDC로부터 직접 가져가거나 GSDC의 컴퓨팅 시설을 활용하여 후처리된 데이터를 가져가기도 한다.

이렇게 상용 분야나 과학기술분야에서 데이터센터의 이용이나 구축이 활성화되고 있으며, 데이터센터의 응용이나 데이터의 규모가 점점 커짐에 따라 내부 스위치나 라우터를 연결하는 네트워크의 대역폭 용량도 커지고 있다. 하지만 데이터센터 네트워크는 성능을 예측하기 어려운 것으로 알려져 있다[5-7]. 갑자기 트래픽이 집중되거나, 성능의 변화가 심하거나, 특히 내부 링크의 단절과 호스트의 다운이 자주 발생하여 네트워크의 비대칭이 발생하여 성능이 열악해지는 등의 문제가 발생하고 있다.

이에 본 논문에서는 네트워크 상태에 따라 트래픽의 부하를 손쉽게 분산할 수 있는 오픈플로우(OpenFlow)[8] 스위치를 이용하여, 상용 데이터센터에 비해 소규모인 과학실험데이터센터 내부의 네트워크를 구성하고 데이터센터 네트워크의 부하 분산을 효율적으로 수행할 수 있는 스케줄링 알고리즘을 제안한다. 제안된 알고리즘에 대한 수학적 모델링과 서비스 시나리오를 제시하며, 최적화 분석 도구를 사용하여 제안된 알고리즘의 성능 분석을 수행한다. 이렇게 함으로써 기존의 데이터센터에서 널리 활용하고 있는 ECMP(Equal Cost MultiPath)[9]와의 성능 비교를 통해 제안된 알고리즘의 우수성을 제시하고자 한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 데이터센터 토폴로지, 부하 분산 및 오픈플로우 기술에 대해 살펴본다. 제 3장에서는 제안하는 스케줄링 알고리즘을 위한 시스템 모델을 기술한다. 제 4장에서는 오픈플로우 기반의 데이터센터 네트워크를 위한 스케줄링 알고리즘을 제안한다. 제 5장에서는 제안된 알고리즘의 성능 평가 결과를 제시하고, 제 6장에서 결론을 맺으며 본 논문을 마무리한다.

Ⅱ. 배 경

2.1. 데이터센터 토폴로지

그림 1은 널리 사용되고 있는 트리 구조의 데이터센터 토폴로지를 보여주고 있다. 이 구조는 단일 장애점(single point of failure)을 갖는 단점이 있다. 또한 에지 계층의 스위치에서 더 높은 계층의 스위치로 상승함에

따라 용량초과(oversubscription) 문제가 발생하고, 이에 따라 계층이 올라갈수록 스위치 포트의 집적도가 높고 고성능의 값비싼 하이엔드 제품을 사용해야 하는 문제점이 있다.

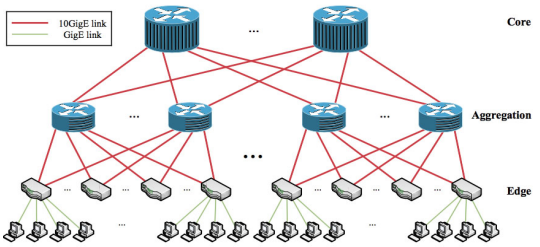


Fig. 1 General topology of hierarchical multi-rooted tree

1950년대에 Charles Clos는 전화망에서 많은 단말들에게 높은 수준의 대역폭을 제공하기 위해, 저가의 일반적인 상용 스위치를 어떻게 상호 연결하면 좋을지를 고민하여 'Clos 토폴로지'라고 불리는 하나의 네트워크 토폴로지를 제안하였다[10]. 이 Clos 토폴로지의 특별한 경우인 팻트리 토폴로지(fat-tree topology)는 상호연동 대역폭이 확장가능하고, 일반적인 저렴한 기성품(off-the-shelf) 이더넷 스위치로 대규모의 데이터센터 네트워크를 구축하여 규모의 경제를 달성하고, 이더넷과 IP 상에서 운영되는 이전 기종의 호스트들과 호환성이라는 측면에서 트리 구조의 토폴로지보다 유리한 측면이 있다. 기존 상용 제품을 사용하기 때문에 응용의 변경도 불필요하다. 또한 구조적으로 NIC(Network Interface Card)의 속도로 통신을 할 수 있는 장점이 있다.

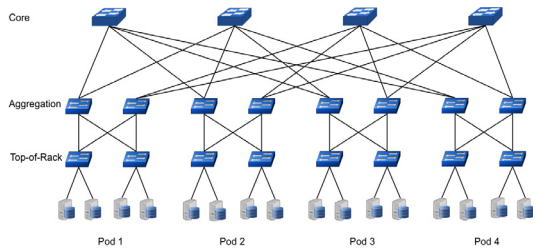


Fig. 2 3-layer topology of 4-ary fat-tree

팻트리 토폴로지는 일반적인 상용 이더넷 스위치를 상호 연결하며, 그림 2와 같은 토폴로지를 갖는다[11]. 전형적인 k-ary 팻트리 토폴로지는 $(k/2)^2$개의 코어스위치(core switch)와 k개의 포드(pod)로 구성된다. 각 포드는 $k/2$개의 집적스위치(aggregation switch), $k/2$개의 ToR(Top-of-Rack) 스위치 및 k개의 호스트들로 구성된다. 각 스위치는 계층에 상관없이 k개의 포트를 동일하게 가진다. 각 ToR 스위치는 $k/2$개의 호스트와 연결된다. 각각의 집적스위치는 $k/2$개의 ToR 스위치와 $k/2$개의 코어스위치와 연결된다. 각각의 코어스위치는 각 포드를 연결하는 k개의 포트를 갖는다. 코어스위치의 i번째 포트는 포드 i와 연결된다. 그리고 일반적으로 k개의 포트를 갖는 스위치로 구성된 팻트리 토폴로지는 $(k/2)^2k$개의 호스트를 지원한다. 이것을 정리하면 표 1과 같다.

Table. 1 Numbers of systems in k-ary fat-tree topology

System	Number
pod	k
core switch	$(k/2)^2$
aggregation switch/pod	$k/2$
ToR switch/pod	$k/2$
host	$(k/2)^2k$

이러한 팻트리 토폴로지는 트리 구조보다 계층 간 상호연결이 더 많은 단점이 있지만 몇 가지 장점을 갖는다. 상호연결이 많다 보니 데이터 전송을 위한 대체 경로를 제공할 수 있다. 대체 경로는 데이터 전송 경로 상의 링크가 단절되었을 때 유용하다. 그리고 플로우렛 스위칭(flowlet switching)[12]과 같은 다중 경로를 통한 플로우 전송도 가능하다. 데이터센터내의 모든 스위치를 저렴한 일반적인 상용제품을 사용하여 구성할 수 있다. 그리고 모든 스위치가 동일하기 때문에 유지 보수가 수월하다. Myrinet[13]이나 Infiniband[14]와 같은 전문화된 하드웨어와 특화된 통신 프로토콜을 가지는 고가의 하이엔드 제품을 사용하지 않아 구축비용을 절감할 수도 있다.

2.2. 부하 분산과 오픈플로우

트래픽을 포워딩하기 위해 가능한 모든 링크를 사용하는 것은 데이터센터 설계에서 매우 중요한 요소이다. 모든 링크의 용량을 사용할 수 있을 뿐만 아니라 경제적으로도 유리하다. 10GbE과 40GbE 링크는 고가인

데 사용하지 않는 것은 낭비이다. 신장 트리(spanning tree)는 다중 경로(multi-path)를 지원하지 않는다. 이러한 문제점을 극복하기 위해 나온 기술들 중에 MLAG (Multichassis Link Aggregation)[15]와 ECMP가 있다.

MLAG는 멀티섀시 번들(multi-chassis bundle)에 있는 모든 링크를 통해서 트래픽을 포워딩할 수 있다. 서로 다른 스위치에 있는 링크들로 번들을 형성하여 하나의 링크처럼 동작하게 하는 기술이다. 또한 2개의 스위치가 다른 2개의 스위치와 MLAG를 사용하여 연결될 수 있다. LACP(Link Aggregation Control Protocol)가 노스바운드와 사우스바운드 사이의 협상을 위해 보통 사용된다. 그림 3에서 호스트와 MLAG 가상 스위치 사이 또는 MLAG 가상 스위치 사이에서 LACP가 사용된다. 같은 MLAG 가상 스위치의 구성원들인 스위치들 사이의 이스트와 웨스트는 벤더 고유의 프로토콜이 사용된다. 즉 MLAG 가상 스위치를 구성하기 위한 기술은 표준이 아니다. 즉 이스트-웨스트 간은 표준이 아니고 노스-사우스 간은 LACP를 사용하기 때문에 타 벤더와 서로 연동할 수 있다.

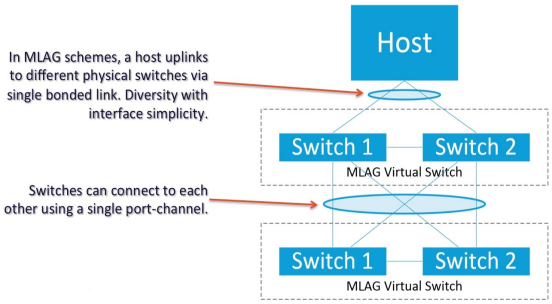


Fig. 3 MLAG virtual switch[9]

ECMP는 지난 수년 동안 데이터센터 네트워크에서 사용 추세가 상당히 증가하고 있다. 점점 증가하고 있는 데이터센터 네트워크 트래픽을 분산하는데 ECMP에 많이 의존하고 있는 실정이다. ECMP는 2계층과 3계층에서 모두 사용할 수 있다. 데이터센터내에서 3계층 ECMP 서비스를 제공하기 위해서 라우팅 프로토콜로써 OSPF(Open Shortest Path First)를 보통 선택한다. 3계층 다중경로 프로토콜은 새로운 것은 아니다. 매 패킷마다 라우팅 경로를 바꾸어 트래픽을 분산하는 측면에서는 플로우 기반의 2계층 프로토콜에 비해 유리하나 많은 TCP 구현물들과 응용들에서 패킷의 순서 역전(out-of-order)에 매우 민감하고 그 결과 패킷 순서 재계산(re-ordering)을 발생시키는 어떤 것도 심각한 문제가 될 수 있다. 따라서 3계층 ECMP보다 2계층 ECMP에 대한 관심이 더 높은 편이다[16].

ECMP는 같은 네트워크 비용(network cost)을 갖는 다중 경로들 사이로 트래픽을 분배하는 메커니즘이다. 현재 데이터센터에서 부하 분산(load balancing)을 위해 가장 많이 사용하고 있는 ECMP 포워딩은 망의 부하를 적절히 분산하지 못하는 것으로 알려져 있다[17]. ECMP는 플로우의 해시(hash) 값에 따라 랜덤하게 데이터 전송 경로를 설정하기 때문에 해시 값의 충돌로 인한 부하 불균형을 초래할 수 있다. 특히 ECMP는 가능한 각 전송 경로의 혼잡도를 고려하지 않고 동일한 네트워크 비용(equal-cost)을 갖는 경로들 중에서 하나를 임의로 선택하는 지역적인 의사결정(local decision)을 내린다.

컨트롤러를 통하여 중앙집중식 제어가 가능한 오픈플로우 네트워크 프로토콜은 데이터 평면과 제어 평면을 분리하여 컨트롤러가 중앙집중식으로 스위치를 제어할 수 있는 장점이 있다. ECMP 스위치의 수가 많은 상용 클라우드 데이터센터와 달리, GSDC와 같은 소규모 데이터센터는 제어해야할 스위치의 수가 작으므로 오픈플로우 기반의 중앙집중식 제어를 하더라도 시간 지연이나 성능 저하의 문제가 크지 않다. 또한 많은 상용 제품이 시장에 출시되어 저렴하게 구입이 가능하다.

오픈플로우 스위치 규격에 있는 오픈플로우 스위치의 포트별 카운터 통계를 기반으로 데이터센터 네트워크의 글로벌 상태를 인지할 수 있다. 또한 네트워크의 변화를 쉽게 조정할 수 있는 장점이 있다. 이런 오픈플로우의 특성을 이용하면 트래픽의 변화에 동적으로 대처할 수 있는 스케줄링이 가능하다. 현재까지 오픈플로우 기반의 데이터센터 네트워크에 대한 스케줄링 논문은 다수 있으나[11, 18-20], 데이터센터 간 부하 분산에 초점을 두거나[19, 20], 데이터 센터 내부의 부하 분산에 초점을 두고 있다[11, 18].

Ⅲ. 시스템 모델

본 논문에서는 데이터센터 구축비용과 응용의 호환성을 고려하여 일반 상용 이더넷 스위치와 라우터를 사

용하는 팻트리 토폴로지를 이용한다. 그리고 오픈플로우 네트워크의 상태 정보를 이용하여 응용 연구자가 실험데이터센터로부터 데이터를 수신하는 성능에 초점을 맞춘 중앙집중식 동적 스케줄링 접근을 시도한다.

3.1. 네트워크 토폴로지

본 논문에서는 그림 4와 같은 k-ary 팻트리 구조의 데이터센터 토폴로지를 사용한다. 여기서 각 스위치는 오픈플로우 프로토콜을 지원하고, 컨트롤러가 중앙집중식으로 데이터센터내의 모든 스위치를 제어한다. 컨트롤러는 스위치의 데이터 포워딩 경로를 추가, 삭제, 변경 등을 요구할 수 있으며, 이러한 명령을 받으면 각 스위치는 자신이 보유한 플로우 테이블을 갱신한다. 컨트롤러는 스위치의 각종 통계 자료를 요구할 수도 있다.

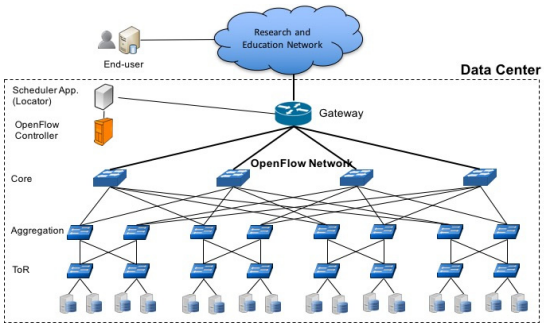


Fig. 4 OpenFlow based fat-tree topology

오픈플로우를 사용하는 소프트웨어 정의 네트워크는 그림 5와 같은 구조를 갖는다. 인프라스트럭처 계층에 해당하는 것이 데이터센터 네트워크이며, 컨트롤러에 해당하는 것이 제어 계층이다. 두 계층 사이의 통신 프로토콜이 오픈플로우이다. 제안된 시스템 모델에서는 API를 통해 제어 계층과 통신하는 응용 계층에 스케줄러 응용과 로케이터(locator) 응용을 둔다. 스케줄러 응용은 표 2에서 보여주는 오픈플로우 스위치의 포트 통계 자료를 주기적으로 모니터링 하여 링크의 잔여 용량을 자체 데이터베이스에 저장하고 그것을 바탕으로 트래픽이 경미한 경로로 데이터를 포워딩하도록 스위치에게 지시하는 기능을 갖는다. 로케이터 응용은 연구자가 전송을 요구하는 파일의 위치를 저장하는 데이터베이스를 보유한다. 보유한 데이터베이스를 기반으로

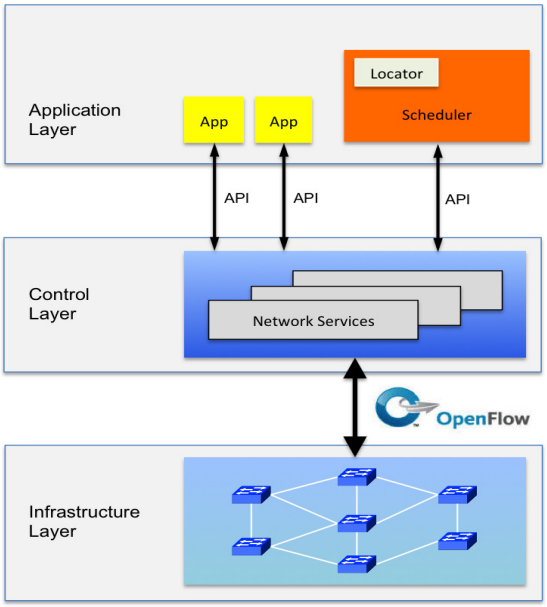


Fig. 5 OpenFlow based scheduler application and locator application

연구자가 요구하는 데이터를 가진 각 호스트에게 파일을 전송하라고 지시하는 기능을 갖는다.

트래픽을 전송할 적정 경로를 찾아내기 위해서는 오픈플로우 스위치와 연결된 링크의 잔여대역폭을 계산하는 것이 필요한데 개념적으로 그 과정을 기술한다. 그림 6은 잔여대역폭을 계산하기 위한 링크의 예시이다. 스위치 A의 포트 a_1의 tx_bytes(표 2 참고)를 기준으로 계산한다. 물론, 스위치 B의 포트 b_1의 rx_bytes를 기준으로 계산하여도 된다. 링크 (a_1,b_1)의 용량은 $C^{a_1b_1}$, 포트 통계를 호출하는 주기를 T_p, 오픈플로우 스위치 A의 포트 a_1에서 $t=t_{n-1}$일 때 tx_bytes=$N^{a_1}_{t_{n-1}}$, $t=t_n$일 때 tx_bytes=$N^{a_1}_{t_n}$, 링크 (a_1,b_1)에서 $t=t_n$일 때 점유대역폭을 $C^{a_1b_1}_{t_n}$, 잔여대역폭을 $R^{a_1b_1}_{t_n}$이라고 하면, 링크 (a_1,b_1)에서 $t=t_n$일 때, 점유대역폭과 잔여대역폭은 각각 식 (1)과 (2)와 같다.

$$C^{a_1b_1}_{t_n-t_{n-1}} = \frac{N^{a_1}_{t_n}-N^{a_1}_{t_{n-1}}}{t_n-t_{n-1}} = \frac{N^{a_1}_{t_n}-N^{a_1}_{t_{n-1}}}{T_p} \quad (1)$$

$$R^{a_1b_1}_{t_n} = C^{a_1b_1} - C^{a_1b_1}_{t_n} \quad (2)$$

Table. 2 Port statistics[8]

Parameter	Value	Parameter size (bits)
rx_packets	Number of received packets	64
tx_packets	Number of transmitted packets	64
rx_bytes	Number of received bytes	64
tx_bytes	Number of transmitted bytes	64
rx_dropped	Number of packets dropped by Rx	64
tx_dropped	Number of packets dropped by Tx	64
rx_errors	Number of receive errors	64
tx_errors	Number of transmit errors	64
rx_frame_err	Number of frame alignment errors	64
rx_over_err	Number of packets with RX overrun	64
rx_crc_err	Number of CRC errors	64
collisions	Number of collisions	64

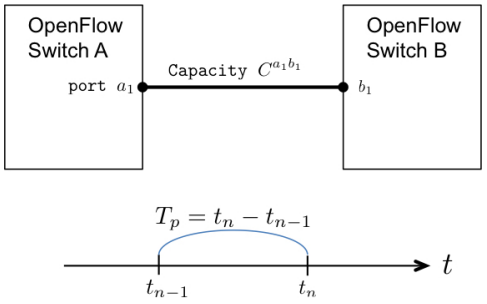


Fig. 6 Illustration of calculation of residual bandwidth

3.2. 서비스 시나리오

KISTI의 GSDC는 고에너지물리, 천문학, 대기, 해양 등 다양한 분야의 대규모 과학실험데이터를 보유하고 있다. 데이터 파일의 사이즈는 수 기가바이트(gigabyte)에서 수 테라바이트(terabyte)까지 다양하다. 일반적인 상용 데이터센터와 비교할 때 보유한 파일의 크기는 크지만, 클러스터 형태로 연결된 호스트의 수는 작은 편이다. 대규모의 원시 과학실험데이터를 요구하는 연구자와 데이터의 특성상 실시간 데이터 전송을 요구하지 않는다. 이러한 특성을 고려하면 로케이터 서버를 두어 저장한 데이터가 어느 호스트에 있는지 정보를 데이터베이스화하고 색인화 하여 요구 파일을 가진 서버를 찾아 그 경로를 계산하여 설정하고 데이터를 응용연구자에게 전송하는 시간지연의 상황은 감내할 만한 수준이다.

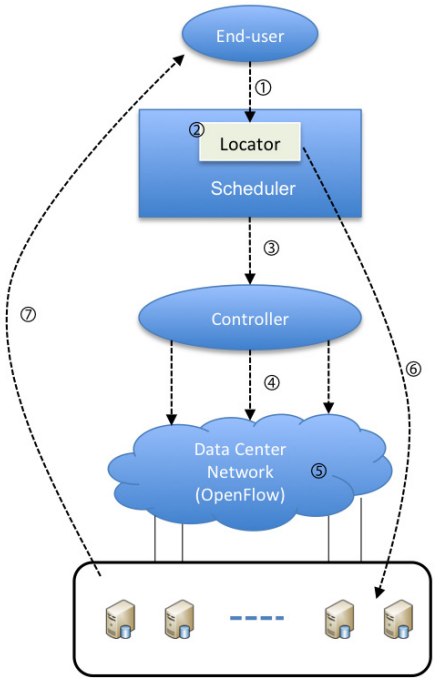


Fig. 7 Service scenario of data transfer

그림 7은 과학실험데이터를 전송하는 서비스 시나리오를 보여준다. 여기서 데이터 파일은 여러 서버에 분산되어 저장되어 있고 응용연구자에게 데이터를 전송할 때 데이터를 저장한 여러 서버가 동시에 응용연구자에게 전송하는 것을 가정한다(예, GSDC의 CMS 및 ALICE 데이터). 이것은 하나의 플로우를 서브플로우로 분할하여 전송하는 다중경로 TCP[21, 22]와 개념적으로 유사하다. 한편, 로케이터 서버는 자신과 관련 있는 도메인의 데이터가 데이터센터 내 어느 서버에 있는지에 관한 정보를 사전에 스케줄러에게 넘겨주고 또한 컨트롤러는 자신이 관장하는 오픈플로우 스위치들에게 포트 통계 정보를 요구하여 받은 후 그것을 스케줄러에게 API를 통해 넘겨준 상태이다.

데이터센터에서 응용연구자가 요구한 파일을 전송하는 절차는 다음과 같다.

① 응용연구자는 자신이 필요한 특정 데이터를 로케이터에게 전송해달라고 요구한다.
② 로케이터는 데이터가 어느 서버들에 분산되어 있는지 정보를 찾아내 스케줄러에게 넘긴다.
③ 스케줄러는 로케이터가 보내준 정보와 사전에 주기적으로 계산해놓은 가용대역폭을 기준으로 경로를 계산하여, 경로상의 스위치들이 갱신해야할 정보를 컨트롤러에게 넘겨준다.
④ 컨트롤러는 스케줄러에게 받은 정보를 기반으로 경로상의 스위치들에게 플로우 테이블을 갱신하라고 지시한다.
⑤ 스위치들은 컨트롤러로부터 받은 정보를 바탕으로 자신의 플로우 테이블을 갱신한다.
⑥ 경로가 설정된 후, 로케이터는 각 서버에게 어떤 파일을 누구에게 보내라는 명령을 내린다.
⑦ 각 데이터 서버는 로케이터의 지시대로 해당 데이터를 응용연구자에게 보낸다.

Ⅳ. 스케줄링 알고리즘의 제안

무지향성의 링크(Undirected Arc)를 갖는 데이터센터 네트워크의 토폴로지 그래프를 노드들의 집합 $\boldsymbol{N}$과 링크들의 집합 $\boldsymbol{A}$로 구성하고 스위치 간 양방향 링크를 고려하여 지향성 그래프(Directed Graph) $\boldsymbol{G}=(\boldsymbol{N},\boldsymbol{A})$로 나타낸다. 풀고자 하는 문제는 다수 개의 데이터센터 내 근원지(source) 호스트로부터 데이터센터 외부의 한 목적지(destination) 호스트로 동시에 데이터를 전송할 때 각 근원지 호스트의 플로우별로 어떻게 트래픽 경로를 설정해야 전송률을 최대화시킬 수 있는지 하는 것이다.

다중 근원지에서 출발하여 단일 목적지로 가고 근원지 호스트별로 단 하나의 플로우만 있는 상황을 가정한다(그림 8 참고). 이때 각 근원지 호스트에는 여러 개의 플로우들이 있을 수 있지만 여기서는 특정 데이터를 보낼 때 근원지 호스트당 한 개의 플로우가 있다고 가정한다. 물론 같은 근원지 호스트 내의 다른 세션에서 플로우를 보내고 있을 수도 있으며 이 트래픽은 배경 트래픽이라고 생각할 수 있다.

네트워크 토폴로지 그래프에서, $\boldsymbol{N}$은 n개의 노드들로 구성된 집합이며, $\boldsymbol{A}$는 m개의 지향성 링크(Directed Arc)들로 구성된 집합이다. 송신 노드 $i \in \boldsymbol{N}$이고, 수신 노드 $j \in \boldsymbol{N}$일 때, 인접 노드간의 링크 $(i,j) \in \boldsymbol{A}$이다. k번째 근원지 호스트를 s_k, 목적지 호스트는 하나이므로 d라고 한다. 위에서 언급한 것처럼 사용자가 요청한 데이터 파일을 가진 근원지 호스트의 총 수는 K개이며, 이 호스트들의 집합을 $\boldsymbol{S}=\{s_1,...,s_K\}$라고 정의한다. s_k의 플로우를 f_k, 각 근원지 호스트들의 플로우의 집합을 $\boldsymbol{F}=\{f_1,...,f_K\}$, s_k가 보내는 플로우의 데이터 전송률 λ_k, 전송률의 집합을 $\boldsymbol{\Lambda}=\{\lambda_1,\cdots,\lambda_K\}$라고 각각 정의한다.

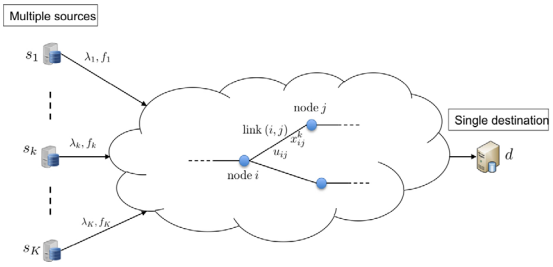


Fig. 8 With one flow per source host, multiple source hosts send flows to single destination host.

x_{ij}^k는 링크 (i,j)에서 각 플로우 f_k의 존재 여부를 나타낸다. 즉 링크 (i,j)에서 x_{ij}^k의 값이 `1'이면 $f_k$가 존재하고 `0'이면 플로우가 존재하지 않는다. 이를 통하여 링크 (i,j)에서 각 플로우별 존재 유무를 파악하면 각 플로우별 경로를 파악할 수 있다. u_{ij}는 링크 (i,j)에서 배경 트래픽을 제외한 최대 가용전송률이다.

여러 근원지로부터 하나의 목적지로 데이터를 동시에 전송하는 것은 multi-commodity flow problem[23]으로 해석할 수 있다. 앞서 기술한 서비스 시나리오를 바탕으로 전송률을 최대화하기 위한 문제를 다음과 같이 정수 선형계획법(integer linear programming)을 이용한 수식으로 표현 가능하다. 식 (3)과 같이 각 근원지 호스트의 전송률을 최대화하는 것은 외부의 목적지 호스트가 데이터센터에서 최대의 전송률로 데이터를 수신하는 것과 동일하다.

$$\text{Maximize} \sum_{k=1}^{K} \lambda_k \tag{3}$$

subject to

$$\sum_{j:(i,j)\in A} \lambda_k x_{ij}^k - \sum_{j:(i,j)\in A} \lambda_k x_{ji}^k = \lambda_k \quad i = s_k;\ \forall f_k \in F \quad (4)$$

$$\sum_{j:(i,j)\in A} \lambda_k x_{ij}^k - \sum_{j:(j,i)\in A} \lambda_k x_{ji}^k = -\lambda_k \quad i = d;\ \forall f_k = F \quad (5)$$

$$\sum_{j:(i,j)\in A} \lambda_k x_{ij}^k - \sum_{j:(j,i)\in A} \lambda_k x_{ji}^k = 0 \quad \forall i \in \{N - \{s_k, d\}\};\ \forall f_k \in F \quad (6)$$

$$\sum_{j:(i,j)\in A} x_{ij}^k = 1 \quad i = s_k;\ \forall f_k \in F \quad (7)$$

$$\sum_{j:(j,i)\in A} x_{ji}^k = 1 \quad i = d;\ \forall f_k \in F \quad (8)$$

$$\sum_{j:(i,j)\in A} x_{ij}^k = 0 \text{ or } 1 \quad \forall i \neq s_k;\ \forall f_k \in F \quad (9)$$

$$\sum_{j:(j,i)\in A} x_{ji}^k = 0 \text{ or } 1 \quad \forall i \neq d;\ \forall f_k \in F \quad (10)$$

$$x_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A;\ \forall f_k \in F \quad (11)$$

$$\lambda_k \geq 0 \quad \forall f_k \in F \quad (12)$$

$$\sum_{k=1}^{K} \lambda_k x_{ij}^k \leq u_{ij} \quad \forall (i,j) \in A \quad (13)$$

주어진 정수 선형계획법은 식 (3)과 같이 데이터센터내에서 플로우들의 최대전송률을 얻기 위한 목적함수와 식 (4) - (13)과 같이 목적함수에 대한 조건식으로 구성된다. 식 (4) - (6)은 각각 근원지 노드, 목적지 노드, 이외의 나머지 노드에서 플로우 보존(flow conservation)을 나타낸다. 식 (4)와 식 (5)는 근원지 노드에서 출발한 하나의 플로우가 중간 노드에서 손실 없이 목적지 노드에 도착하는 것을 의미하며, 식 (6)은 중간 노드에서 수신한 플로우가 손실 없이 그대로 송신된다는 것을 의미한다.

식 (7) - (10)은 각 플로우가 나뉘어져 여러 경로로 전송되지 않고 오직 하나의 경로를 선택하여 전송된다는 것을 나타낸다. 식 (7)은 근원지 노드에서 다음 노드로의 경로가 1개라는 것을 나타내고, 식 (8)은 목적지 직전의 노드들에서 목적지 노드로의 경로가 1개라는 것을 나타낸다. 식 (9)와 (10)은 중간 노드의 직전 노드와 다음 노드로의 경로가 각각 1개임을 나타낸다.

식 (13)은 특정 링크 (i,j)를 지나는 모든 플로우의 합은 해당 링크 (i,j)의 가용량인 u_{ij}를 초과할 수 없음을 나타낸다.

주어진 정수 선형계획법은 u_{ij}에 따라 플로우들의 최대 전송률을 위한 경로를 탐색한다. 게다가, 경로 탐색 이후 각 링크 (i,j)의 자원 사용 현황을 파악할 수 있다. 따라서 플로우들의 최대 전송을 위한 경로 탐색을 수행한 후 모든 링크 (i,j)의 자원 사용 현황을 파악하여 가용대역폭이 높은 순서대로 스케줄링을 하면 전송률을 향상시킬 수 있다.

Ⅴ. 제안된 알고리즘의 성능 평가

제 4장에서는 최적화 기법을 이용하여 제안된 스케줄링 알고리즘의 수학적 모델링을 제시하였다. 이번 절에서는 서비스 시나리오를 바탕으로 최적화 분석 툴을 이용하여 제안된 알고리즘의 성능을 분석한다.

네트워크 토폴로지는 그림 4와 같은 4-ary 팻트리 토폴로지를 사용하였다. 표 1의 $k=4$에 해당하는 시스템의 개수로 전체 시스템은 구성된다. 데이터센터 네트워크의 각 스위치는 오픈플로우 프로토콜을 사용하며 표 2와 같은 오픈플로우 스위치의 포트 통계 자료 기반으로 플로우 전송 경로 상의 각 링크에 대한 점유대역폭을 식 (1)과 같이 계산하여 스케줄러는 가용 대역폭 u_{ij}를 이미 알고 있다고 가정한다.

각 호스트는 임의의 배경 트래픽을 가지고 있으며, 응용연구자가 요구한 특정 데이터를 가진 호스트 수는 데이터의 종류에 따라 가변적이다. 특정 데이터를 전송하도록 로케이터로부터 지시받은 호스트는 동시에 데이터 플로우를 보낸다. 여기서 각 호스트는 요구받은 데이터를 전송하는데 단 하나의 플로우만을 가진다고 가정한다. 따라서 4개의 호스트가 특정 데이터를 분할해서 가지고 있다면 4개의 호스트가 동시에 데이터 플로우를 보내게 되고, 데이터센터 네트워크를 흘러가는 총 플로우의 수는 4개가 된다.

제안된 알고리즘과 성능을 비교하기 위하여 데이터센터 네트워크에서 가장 많이 사용하는 포워딩 방식인 ECMP를 사용하였다. ECMP는 패킷별로 포워딩을 하지 않고 플로우별로 포워딩을 하고, 특정 데이터를 전송하는 각 호스트는 하나의 플로우만을 이용해서 특정 데이터를 전송한다고 가정한다. ECMP는 데이터센터 토폴로지에 따라 전송할 수 있는 경로의 수가 팻트리

구조에 따라 가변적이다. 4-ary 팻트리 구조에서는 동일 비용을 갖는 다중 경로는 호스트별로 4개씩 존재한다. 토폴로지 상에서 각 링크의 용량은 10Mbps이며, 각 데이터 호스트는 랜덤 백그라운드 플로우가 존재한다. 이때 목적지 노드로 파일을 전송할 때, 패킷 사이즈는 1,000 바이트, TCP의 윈도우 크기는 8,000 바이트이다. 전송하고자하는 파일의 크기는 100MB이다.

위와 같은 전제하에서 사용자가 요구한 데이터를 가지고 있는 호스트의 수를 1부터 8까지 늘려가면서 시뮬레이션을 수행하였다. 이때 요구 데이터를 가지고 있지 않은 호스트는 백그라운드 트래픽만 흘린다. ECMP의 경우는 ns-2를 사용하였으며, 제안된 알고리즘은 최적화 분석 툴을 사용하여 시뮬레이션을 수행하였다.

그림 9는 목적지 노드에서 수신한 데이터의 평균 처리율(throughput)을 나타낸다. 제안된 스케줄링 알고리즘의 처리율이 ECMP 처리율의 평균보다 2배 가까이 높음을 알 수 있다. 여기서 ECMP(worst case)는 하나의 경로로 트래픽이 집중할 때의 처리율을 나타낸다.

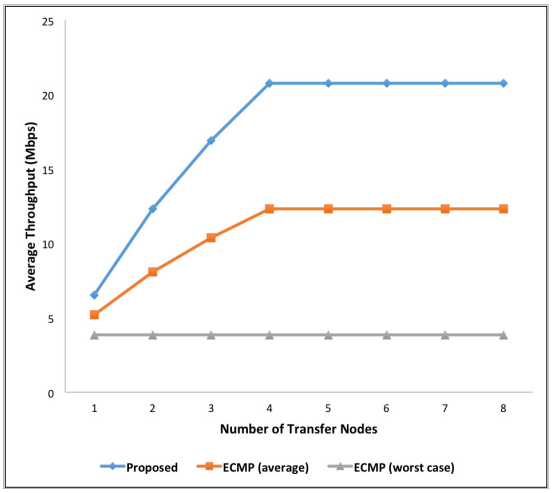


Fig. 9 Average throughputs of proposed algorithm and ECMP

그림 10은 100MB 파일을 전송할 때 걸리는 총 시간을 나타낸다. 전송 노드의 수가 증가함에 따라 총 전송해야할 데이터양이 비례하여 증가하지만, 파일 전송 시간은 ECMP보다 제안된 알고리즘에서 상대적으로 기울기가 낮고, 전송시간이 작음을 알 수 있다.

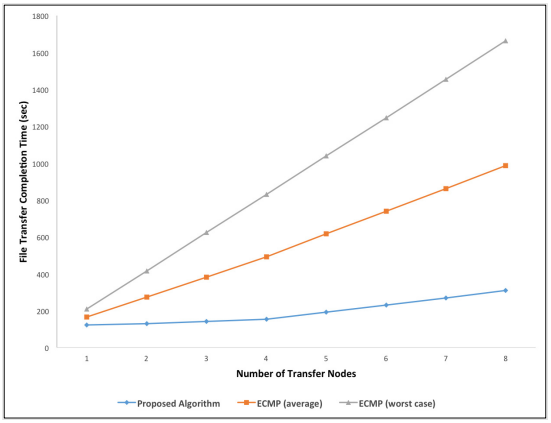


Fig. 10 File transfer completion time of proposed algorithm and ECMP

위의 시뮬레이션 결과를 볼 때, 전반적으로 제안된 알고리즘이 ECMP보다 데이터 전송 성능이 우수함을 알 수 있으며, 데이터센터 네트워크의 링크 용량을 효율적으로 사용하고 있음을 확인할 수 있다.

VI. 결 론

본 논문에서는 최근에 클라우드 서비스 제공자나 기업 등에서 활발히 구축하고 있는 데이터센터의 토폴로지와 내부 네트워크의 현재 상황을 알아보았다. 일반적인 트리 구조의 문제점에 비해 비용 효율적인 팻트리 토폴로지에 대해 알아보았다. 기존의 데이터센터에서 많이 활용하고 있는 부하 분산 기법인 ECMP의 문제점과 오픈플로우를 사용하는 데이터센터 네트워크에 관한 연구 동향도 살펴보았다.

위에서 언급한 여러 사항을 고려하여 팻트리 구조를 갖는 오픈플로우 기반의 토폴로지를 사용한 시스템 모델을 설정하고 데이터센터, 특히 과학실험데이터센터의 데이터 전송 성능을 향상시킬 수 있는 서비스 시나리오와 스케줄링 알고리즘을 제안하였다. 제안된 알고리즘은 정수 선형계획법을 이용하여 문제를 풀 수 있으며, 플로우들의 최대 전송을 위한 경로 탐색을 수행한 후 모든 링크의 자원 현황을 파악하고 플로우 사이의 간섭을 고려하여, 최적 스케줄링을 수행함으로써 전송률을 향상시킬 수 있는 장점이 있다.

ACKNOWLEDGMENTS

This research was supported by Korea Institute of Science and Technology Information(KISTI)

REFERENCES

[1] CERN [Internet]. Available: https://home.cern/.

[2] Joint Institute for VLBI ERIC [Internet]. Available: https://www.jive.nl/.

[3] Fermilab [Internet]. https://www.fnal.gov/.

[4] KEK [Internet]. https://www.kek.jp/en/.

[5] T. Benson, A. Akella, and D. A. Maltz, “Network Traffic Characteristics of Data Centers in the Wild,” in *Proceeding of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10)*, New York, NY, USA, pp. 267-280, 2010.

[6] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta, “VL2: a scalable and flexible data center network,” in *Proceeding of the ACM SIGCOMM conference on Data communication (SIGCOMM '09)*, New York, NY, USA, pp. 51-62. 2009.

[7] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken, “The nature of data center traffic: measurements & analysis,” in *Proceeding of the 9th ACM SIGCOMM conference on Internet measurement (IMC '09)*, New York, NY, USA, pp. 202-208. 2009.

[8] Open Networking Foundation, O*penFlow Switch Specification Version 1.5.1*, 2015.

[9] RFC 2992, *Analysis of an Equal-Cost Multi-Path Algorithm*, IETF, US, 2000.

[10] Charles Clos, “A Study of Non-blocking Switching Networks,” *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406-424, March 1953.

[11] Li, Yu, and Deng Pan, “OpenFlow based load balancing for Fat-Tree networks with multipath support,” in *Proceeding of the 12th IEEE International Conference on Communications (ICC'13)*, Budapest, Hungary, pp. 1-5, 2013.

[12] Shan Sinha, Srikanth Kandula, and Dina Katabi, “Harnessing TCPs Burstiness using Flowlet Switching,” in *Proceeding of the 3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, USA, 2004.

[13] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W. Su, “Myrinet: A gigabit-per-second local area network,” *IEEE micro*, vol. 15, no. 1, pp. 29-36, 1995.

[14] InfiniBand Trade Association. The InfiniBand Architecture Specification [Internet]. Available: http://www.infinibandta.org/specs/.

[15] Ethan Banks. The Ethernet Switching Landscape - Part 04 - Multichassis Link Aggregation (MLAG) [Internet]. http://ethancbanks.com/2014/03/27/the-ethernet-switching-landscape-part-04-multichassis-link-aggregation-mlag/.

[16] Marten Terpstra. ECMP: It's not all equal, or even normal [Internet]. Available: http://www.plexxi.com/2013/08/ecmp-its-not-all-equal-or-even-normal/.

[17] S. Kandula, D. Katabi, S. Sinha, and A. Berger, “Dynamic Load Balancing Without Packet Reordering,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 51-62, March 2007.

[18] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” in *NSDI*, vol. 10, 2010.

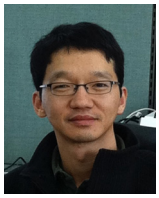
[19] Jain, Sushant, et al., “B4: Experience with a globally-deployed software defined WAN,” *ACM SIGCOMM Computer Communication Review*, vol. 43. no. 4, pp. 3-14, 2013.

[20] Hong, Chi-Yao, et al., “Achieving high utilization with software-driven WAN,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp.15-26, 2013.

[21] M. S. Kim, J. Y. Lee, and B. C. Kim, “Design of MPTCP Congestion Control based on BW measurement for Wireless Networks,” *Journal of the Korea Institute of Information and Communication Engineering*, vol. 21, no. 6, pp. 1127-1136, Jun. 2017.

[22] M. S. Kim, K. M. Han, J. Y. Lee, and B. C. Kim, “Design of Bandwidth Measurement based Scheduler for Improving MPTCP Performance in Bufferbloat Environment,” *Journal of the Korea Institute of Information and Communication Engineering*, vol. 21, no. 8, pp. 1508-1516, Aug. 2017.

[23] Ravindra, K. Ahuja, Thomas L. Magnanti, and James B. Orlin, *Network flows: theory, algorithms, and applications*, Upper Saddle River, NJ: Prentice Hall, 1993.



공정욱(Jong Uk Kong)
1993년 2월 한국과학기술원 전기전자공학부 공학사
1998년 2월 포항공과대학교 정보통신대학원 공학석사
2015년 8월 충남대학교 정보통신공학과 공학박사
1993년 ~ 2001년 ㈜데이콤 종합연구소 선임연구원
2001년 ~ 2002년 ㈜맥스웨이브 부장
2002년 ~ 현재 한국과학기술정보연구원 책임연구원
※관심분야 : Federated Authentication, 네트워크 성능분석

민석홍(Seok Hong Min)

2005년 공주대학교 전기전자정보공학과 공학석사
2010년 ~ 2015년 충남대학교 전자전파정보통신공학과 공학박사
2004년 한국전자통신연구원 BcN 시험기술팀 위촉연구원
2005년 디지피아㈜ 방송장비팀 연구원
2006년 ~ 2009년 ㈜엠티아이 전임연구원
2016년 ~ 현재 ㈜민데이터 대표이사
※관심분야 : SDN, 무선 메쉬망, MANET

이재용(Jae Yong Lee)

1988년 2월 서울대학교 전자공학과 공학사
1990년 2월 한국과학기술원 전기전자공학부 공학석사
1995년 2월 한국과학기술원 전기전자공학부 공학박사
1990년 ~1995년 디지콤 정보통신 연구소 선임연구원
1995년 ~ 현재 충남대학교 전파정보통신공학과 교수
※관심분야 : 무선 메쉬망, MANET, 네트워크 성능분석

김병철(Byung Chul Kim)

1988년 2월 서울대학교 전자공학과 공학사
1990년 2월 한국과학기술원 전기전자공학부 공학석사
1996년 2월 한국과학기술원 전기전자공학부 공학박사
1993년 ~ 1999년 삼성전자 무선네트워크 사업부 선임연구원
1999년 9월 ~ 현재 충남대학교 전파정보통신공학과 교수
※관심분야 : 유/무선 인터넷, 이동통신 네트워크, 이동성 처리