Journal of Korea Multimedia Society Vol. 23, No. 2, February 2020(pp. 247-254)
https://doi.org/10.9717/kmms.2020.23.2.247

SDN 환경에서의 데이터 생성 형태를 고려한 효율적인 부하분산 기법

윤지영†, 권태욱††

An Efficient Load Balancing Technique Considering Forms of Data Generation in SDNs

Jiyoung Yoon†, Taewook Kwon††

ABSTRACT

The recent Internet environment is characterized by the explosion of certain types of data, as the data that people want is affected by certain issues. In this paper, we propose a load balancing technique that considers the data generation forms. The concept of this technique is to prioritize some type of data when it suddenly explodes. This is a technique to build an add-on middle box on a switch to monitor packets and give priority to a queue for load balancing. This technique worked when certain types of data exploded. SDN(Software Defined Networking) has the advantage of efficiently managing a number of network equipment. However, load balancing in the SDN environment has not been studied much. Applying the proposed load balancing technique in the SDN environment can save time and budget and easily implement our policies. When the proposed load balancing technique is applied to the SDN environment, it has been found that the techniques we want can be easily applied to the network systems, and that efficient data processing is possible when certain types of data explosion.

Key words: SDN(Software Defined Networking), Load Balancing, Priority, Latency, Packet Scheduling

1. 서 론

오늘날 IT 인프라가 기하급수적으로 증가하게 되면서 네트워킹 백본들은 네트워크 장치에서 생성된 엄청난 양의 데이터를 처리해야만 하게 되었으며, 이를 해결하기 위해 다양한 패킷 전송 방법과 부하분산(Load Balancing) 기법이 개발되고 적용되어왔다. 특히 최근에는 SNS(Social Network Service) 사용의 급격한 증가로 과거와는 비교할 수 없을 정도로 데이터양이 폭주하게 되었는데, 여기서 우리가 주목해야 할 점은 사용자들이 원하고 찾는 데이터가 특정 이슈에 영향을 받아 편중될 수도 있다는 것이다. 정치·사회·경제적인 사안들에 대해 대중들은 높은 관심을 보이게 되는데, 이러한 이슈가 발생하게 되면 이에 대한 정보요구가 급속히 증가하게 된다. 이를 위해 본 논문에서는 SDN을 기반으로 하여 데이터 생성 형태를 고려한 효율적인 부하분산 기법을 연구하였다. SDN이란 기존의 라우터나 스위치와 같이 소프트웨어 기반의 컨트롤 영역 (Control Plane)을 하드웨어 기반의 데이터 영역(Data Plane)과 분리하는 개념으로써 컨트롤 영역을 중앙 SDN 컨트롤러를 통해 중앙집중화하여 데이터 영역의 네트워크 장비

※ Corresponding Author : Jiyoung Yoon, Address: (33021) Hwangsanbeol-ro 1098, Yangchonmyeon, Nonsan, Korea, TEL : +82-10-5347-6916, E-mail : laputa86_jy@naver.com
Receipt date : Aug. 14, 2019, Revision date : Jan. 7, 2020
Approval date : Jan. 20, 2020

† Dept. of Computer Eng., Graduate School, Korea National Defence University
†† Dept. of Computer Eng., School of National Defence Science, Korea National Defence University (E-mail : kwontw9042@kndu.ac.kr)

를 제어하는 기술이다[1]. 이 두 영역의 분리를 통해 네트워크 관리자는 네트워크 정책의 변경이 필요할 때 체계 내의 많은 장비들을 일일이 수정할 필요 없이 중앙에서 손쉽게 원하는 정책을 주입하고 관리할 수 있게 되었다. 본 논문에서 SDN 환경 하 제안하는 부하분산 기법을 적용하는 것은 이러한 SDN의 특성이자 장점을 활용하기 위함이다. 기존 네트워크 체계에서의 부하분산은 매우 전통적인 연구 주제로써 과거부터 수많은 연구가 이루어져 온 것과는 대조적으로, SDN이라는 획기적인 개념의 새로운 네트워크 체계가 등장한 것은 불과 10년도 되지 않았고, 지금까지는 주로 기존 네트워크 체계에서의 SDN 환경 구축 등 개괄적인 내용 위주로 연구가 이루어져 왔을 뿐 부하분산 분야는 현재까지 활발한 연구가 진행되지 않은 것이 사실이다[2]. 따라서 본 논문에서는 컨트롤 영역과 데이터 영역을 분리하여 네트워크 운영자가 원하는 정책을 손쉽게 적용할 수 있는 SDN의 장점을 활용하는 한편, 이슈 발생에 따라 특정 데이터가 폭증하는 최근 인터넷 환경의 특수성을 고려하는 새로운 부하분산 기법을 제안하는데 목적이 있다. SDN 환경에서는 기존의 네트워크 장비의 교체 없이 우리가 원하는 부하분산 기법을 소프트웨어를 통해 적용하여 이전보다 효율적인 네트워크를 구현할 수 있다. 다만, SDN의 Openflow에서는 서비스별 우선순위를 지정할 수 없는 것처럼 QoS 처리가 어려운 제한사항이[3] 있어 이를 손쉽게 극복하기 위해 Middle Box를 구축한 후 스위치와 연결(Add-on)하여 우리가 원하는 부하분산 정책을 쉽고 빠르게 적용하는 방안을 제안한다. 본 논문에서는 Middle Box를 구축하여 입력되는 다양한 데이터의 유형을 분석한 후 데이터가 일정 기준 이상 폭증될 때 이를 우선처리하는 기법을 제안한다. 이를 평시 일반적인 네트워크 환경으로 가정한 무작위(Random) 데이터와 데이터가 폭증하는 상황을 가정하기 위한 군집(Burst) 데이터 등 두 가지 유형의 데이터를 전송했을 때 각각의 결과를 비교 분석 후 그 효과를 검증하였다. 2장에서는 SDN과 부하분산 기법에 대하여 알아본 후, 3장에서는 제안하는 데이터 생성 형태를 고려한 부하분산 기법에 대하여 설명한다. 4장에서 실험 방법 설명 및 실험 결과를 분석한 후 5장에서 결론을 맺는다.

2. 관련 연구

2.1 SDN

SDN은 컨트롤 영역과 데이터 영역을 분리하여 중앙집중적 통제 시스템에서 네트워크 장비의 트래픽 처리 방식과 기능을 관리할 수 있도록 설계되고 구축된 프레임워크로써 전통적인 방식의 네트워크는 네트워크 장비의 제어기능이 하드웨어에 있었지만, SDN에서는 컨트롤 영역을 하드웨어에서 분리, 소프트웨어를 통해 논리적 또는 가상적인 실체로서 네트워크를 제어할 수 있다[4]. 이처럼 컨트롤 영역과 데이터 영역을 분리하기 위해서는 데이터 영역은 하드웨어에 남겨두고 컨트롤 영역의 기능을 빼내어 컨트롤러에서 그 임무를 수행하게 하는데 사용자는 컨트롤러 위에 사용 환경에 적합하고 자신이 원하는 애플리케이션을 설치하여 사용하는 방식이다.

SDN에서 사용자는 응용계층(Application Layer)을 통해 컨트롤 계층(Control Layer)에 정책을 전달하고 컨트롤 계층(Control Layer)은 전달받은 정책을 Flow로 변경하여 기존 체계의 데이터 영역에 해당하는 인프라계층(Infrastructure Layer)에 전달하는 방식으로 사용자의 정책이 실제 네트워크에 적용되는 구조이다[5]. 인프라계층은 실제 네트워크 장비인 스위치나 라우터 등을 포함하는데 Flow Table에 정의된 규칙에 따라 스위치로 전달하거나 패킷을 처리한다. 컨트롤러는 Southbound 전용 프로토콜을 이용하여 인프라계층의 스위치들과 통신하는데 그 중 가장 대표적인 것이 Openflow이다. Openflow란 SDN만을 위한 공개 표준 프로토콜로서 가장 많이 사용되고 있는데, 컨트롤러가 Openflow를 통해 스위치에게 Flow를 내리고 스위치는 이를 기반으로 Flow Table을 생성한다[6]. 이후 들어오는 패킷들은 바로 이 Flow Table에 따라 패킷을 전송하는 것이다. 바로 이러한 SDN의 특성으로 인해 사용자들이 새로운 프로토콜을 연구하거나 스위치나 라우터 관련 새로운 기술적 시도가 있을 시에 실제 네트워크상에서 테스트를 할 수 없는 물리적 제약을 해결할 수 있게 되었다.

2.2 부하분산 기법

부하분산이란 트래픽이 많을 때 여러 서버가 분산처리하여 서버의 부하율 증가, 부하량, 등을 고려하

여 적절히 분산처리하여 해결해 주는 서비스를 말한다[7]. 부하분산의 몇 가지 대표적인 기법을 소개한다.

2.2.1 라운드로빈 스케쥴링

순차 방식으로 사용자의 요구를 차례대로 각 서버에 균등하게 분배하는 방식으로 일반적인 다른 기법에 비해 빠른 특성이 있다. 서버 커넥션 수나 응답시간과는 무관하게 그룹 내의 모든 서버를 동일하게 처리하여 일반적인 구성에 있어 다른 기법에 비해 간단하고 빠른 장점이 있는 반면, 요청이 폭주할 경우 실제 서버 사이에 동적인 부하 불균형 상태가 생길 수 있다[8].

2.2.2 최소 접속 스케쥴링

접속이 가장 적은 서버로 요청을 직접 연결하는 방식으로서 비슷한 성능의 서버들로 구성된 네트워크에서 가장 효과적이다. 이와 같이 성능이 비슷한 서버들로 구성되어있는 경우에는 아주 큰 부하가 특정 서버로만 집중되지 않기 때문에 부하가 큰 경우에도 효과적으로 처리할 수 있다.

2.2.3 응답시간 방식

응답이 가장 빠른 서버에 이용자를 연결하는 방법으로써, 서버와 통신을 하면서 그 응답시간에 대한 학습을 통하여 응답시간이 빠른 쪽으로 많이 보내주고, 응답이 느린 쪽으로는 적게 보내는 방식이지만, 특정 서버에 접속 부하가 걸릴 수 있다는 점에서 비효율적일 수 있다[9].

2.3 SDN에서의 부하분산 기법

본 논문에서 다루고자 하는 부하분산 기법이 실제 네트워크에서 작동하려면 네트워크 내에 있는 모든 장비에 동일한 정책을 적용해야 한다. 기존의 스위치나 라우터와 같은 네트워크 장비에도 제조업체에서 주입한 간단하고 일반적인 방식의 부하분산 기법이 적용되어 있지만, 이를 사용자가 임의로 변경하거나 수정할 수는 없다. 또한, 이 기법들은 본 연구의 배경이 되는 특정 이슈 발생에 따른 폭증 데이터를 효율적으로 처리할 수 없다. 반면, SDN 로드밸런서는 프로그래밍이 가능하여 자신만의 부하분산 전략을 설계할 수 있으며, 이를 위해 별도의 장비가 필요하지도 않다. 이렇듯 네트워크 운영자 중심의 중앙집중적 네트워크 관리가 가능한 SDN의 장점을 활용한다면 우리가 제안하는 부하분산 기법을 소프트웨어 프로그래밍을 통해 네트워크 내의 모든 장비에 손쉽게 적용할 수 있을 것으로 기대된다.

3. 데이터 생성 형태를 고려한 부하분산 기법

3.1 Middle Box(로드밸런서)

본 논문에서 제안하는 부하분산 기법은 특정 유형의 데이터가 폭증하는 네트워크 환경에서 입력되는 데이터의 유형을 분류하고 그중 폭증하는 데이터를 우선 처리하는 개념으로써 데이터 생성 형태를 고려한 부하분산 기법이라고 구체화할 수 있다. 이를 구현하기 위해 스위치에 연결(Add-on) 하여 로드밸런서의 역할을 하는 Middle Box를 구축하고 이를 SDN 컨트롤러와 연결한다.

스위치에 Middle Box를 연결하고 이를 SDN 컨트롤러와 연동하여 컨트롤러를 통해서 부하분산 정책 설정을 가능하게 한다. Middle Box는 데이터가 폭증하게 되면 이를 프로그래밍한 정책에 따라 처리하며, 이러한 정책은 컨트롤러와 연동되어 컨트롤러에서 자동으로 제어하게 된다.

3.2 Middle Box의 세부 동작

Middle Box는 Fig. 1과 같이 입력되는 큐와 분류기(Classifier), 우선 처리와 정상처리 등 두 가지 유형으로 동작하는 큐로 구분된다. 우선 패킷이 순서대로 들어오게 되면 분류기는 두 가지 정책으로 처리하는데, 하나는 데이터 생성 형태를 고려하여 폭증하는 유형의 데이터를 우선 처리하는 정책이고, 다른 하나는 대조군으로서 일반적인 라운드로빈 방식으로 패킷을 분류하여 큐로 넘기는 정책이다. 큐는 총 5개로

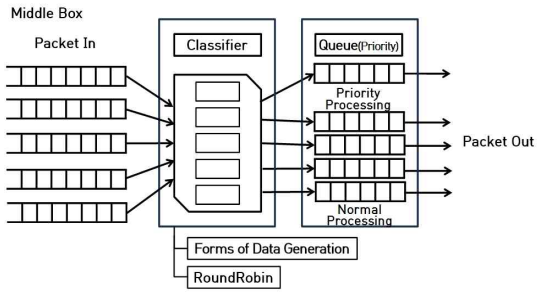

Fig. 1. Middle box processing.

설계하였으며, 본 연구에서 제안하는 정책을 실행하는 경우에는 1개의 우선 처리 큐와 4개의 정상처리 큐로 구성하였고, 라운드로빈 방식에서는 5개 모두 정상처리하는 큐로 구성하였다.

첫 번째 정책인 데이터 생성 형태를 고려한 부하분산 방식은 분류기에서 입력되어 들어오는 데이터를 모니터링하다가 동일한 형태의 데이터가 군집 되어 들어오게 되면 이를 폭증하는 것으로 판단하고 제안하는 기법으로 동작한다. 본 연구의 실험 환경에서는 입력되는 데이터 중 동일한 패킷이 5개 이상 군집되어 들어오는 경우를 폭증 데이터로 판단한다. 5개 이상의 데이터가 군집되어 입력되면 분류기는 그 폭증 데이터의 시작 패킷부터 20개의 패킷을 큐로 보내는데, 이때 전술한 두 가지 유형의 큐 중 우선처리 큐로 보낸다. 패킷을 넘겨받은 우선 처리 큐는 이를 다른 4개의 정상처리 큐에 우선하여 패킷을 내보내게 된다. 20개의 패킷을 넘긴 분류기는 이후 다시 정상작동하다가 또다시 5개 이상의 동일한 유형의 패킷이 들어오는 경우 위의 과정을 반복한다. 그 외 폭증 데이터가 아닌 일반적인 정상 데이터는 분류기에서 나머지 4개의 정상처리 큐로 보내게 되며, 이 4개의 정상처리 큐는 패킷이 들어오는 순서대로 FIFO 방식으로 패킷을 내보내는 방식으로 데이터를 처리한다. 이와 같은 폭증 데이터의 판단 기준이나 우선 처리 패킷의 수는 물론 큐의 개수 등은 네트워크 관리자가 트래픽 상황을 고려하여 프로그램 수정을 통하여 임의 조정이 가능하다.

두 번째 정책인 라운드로빈 방식으로 처리한 패킷은 5개의 동일한 조건의 정상처리 큐로 균등하게 보내져 역시 FIFO 방식으로 패킷을 내보낸다. 이때 큐에서는 별도의 우선 처리 큐 없이 5개 모두 정상처리 큐로 동작한다. 이상의 과정을 도식화하면 Fig. 2와 같다. 이러한 Middle Box의 구성은 들어오는 패킷의 패턴을 분석하여 특정 유형의 데이터가 폭증할 때 이를 우선 처리하여 네트워크가 원활히 운용되도록 소프트웨어 측면에서 쉽게 정책을 하달할 수 있는 장점이 있다. 또한, 군과 같이 전·평시가 구분되는 조직에서 다양한 상황에 맞춰 유연한 네트워크를 운용할 수 있다는 측면에서 그 가치가 있다고 할 수 있다. 본 논문에서는 제안하는 부하분산 기법의 효과를 검증하기 위해 일반적인 네트워크 상황과 데이터가 폭증하는 두 가지 상황을 가정하여 다양한 조건

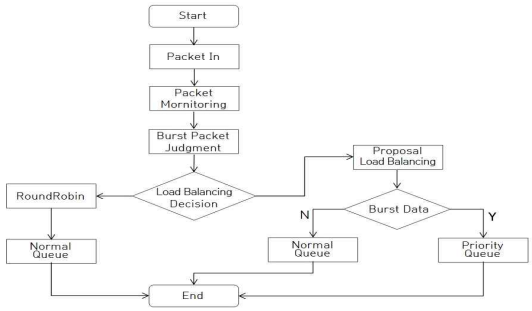

Fig. 2. Middle box flowchart.

아래 실험을 하였다.

4. 실험 결과

4.1 시스템 구현

단대단 호스트간의 폭증하는 데이터 전달의 효율적인 부하분산 기법을 확인하기 위해 패킷 생성기 역할을 하는 Client host와 분류기 및 큐의 기능을 담당하는 Middle Box, Client host가 보낸 데이터를 받아들여 출력하는 Server host를 각각의 파이썬 스크립트로 구성하였으며, 미니넷 환경에서 구동한다.

4.2 실험 환경

제안하는 부하분산 기법을 가상 네트워크 환경에 구현하여 그 성능과 효용성을 평가하였다. 실험 환경은 Intel (R) Core (TM) i5-8250U @ 1.60GHz CPU에 Windows 10 OS, 8GB RAM이며, VMware Workstation14 pro, Ubuntu 14.04.4 가상머신 환경에서 미니넷 2.2.2 에뮬레이터와 Pox 컨트롤러를 사용하였다.

4.3 실험 방법

실험은 Fig. 3과 같이 패킷 생성기 역할을 하는 h1

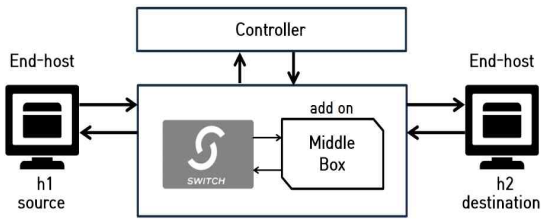

Fig. 3. Experiment design.

에서 패킷을 전송하여 Middle Box를 거쳐 데이터를 분류 및 우선 처리한 후 h2에서 패킷을 받아 출력할 때까지의 지연시간(Latency)을 측정하는 방식으로 실시한다. 실험은 크게 두 가지 유형의 데이터를 전송하여 측정하는데 일반적인 네트워크 상황을 가정한 무작위 데이터와 데이터가 폭증하는 상황을 가정한 군집 데이터이다. 이를 각각 제안하는 부하분산 기법과 일반적인 라운드로빈 방식으로 처리한 결괏값을 비교 분석한다. 또한, 전체 데이터가 아닌 폭증(군집) 데이터만 처리한 결괏값을 분리 측정하여 비교한 후 본 연구에서 제안하는 부하분산 기법이 폭증 데이터를 실제로 얼마나 효과적으로 처리했는지 확인한다.

우선 입력되는 패킷은 무작위 데이터와 군집 데이터로 구분하여 두 가지 형태의 데이터를 전송한다. 전송하는 데이터의 형태는 우선순위를 5가지 유형으로 제한하였고, 1차례 전송 시 총 10,000개의 패킷을 보내며, 이를 100차례 반복하여 최곳값과 최솟값을 제외한 지연시간의 평균값을 측정한다. 분류기에서는 두 가지의 패킷 처리 정책을 수행하는데, 첫 번째 정책은 데이터 생성 형태를 고려한 우선 처리 방식으로써 본 실험에서는 동일한 유형의 데이터가 5개 이상 연이어 전송될 시 이를 폭증하는 데이터로 간주하여 우선 처리한다. 두 번째 정책은 일반적인 라운드로빈 방식으로 패킷을 처리하는 정책이다. 큐는 총 5개의 큐로 구성하였으며, FIFO 방식으로 패킷을 처리한다.

4.4 실험 결과

본 논문에서는 총 8가지 상황을 가정하여 실험하였다. 우선 전송하는 데이터를 모두 처리했을 경우의 지연시간을 측정하는데 제안하는 부하분산 기법을 적용하여 처리한 것과 일반적인 라운드로빈 방식으로 처리한 결과를 비교한다. 이때 전송되는 데이터를 폭증하는 데이터(실험1)와 일반적인 데이터(실험2) 두 가지 유형으로 구분하여 실험한다. 다음은 제안하는 기법이 실제로 폭증 데이터를 처리하는 것에 효과적인지 확인하기 위해서 폭증 데이터만 처리한 지연시간을 측정한다. 이때 역시 두 가지 정책을 각각 적용하여 결과를 비교 분석하며, 입력되는 데이터도 폭증 데이터(실험3)와 일반 데이터(실험4)를 구분하여 처리 결과를 확인한다.

4.4.1 전송된 데이터 전체 지연시간 측정(실험1 및 실험2 참조)

우선 폭증 데이터와 일반 데이터의 전체 지연시간을 측정한 결과이다. Fig. 4는 폭증 데이터가 발생한 상황에서 제안하는 정책과 일반 라운드로빈 방식을 적용하여 측정한 결과를 보여주며, Fig. 5는 일반적인 상황을 가정한 무작위 데이터를 전송했을 때의 결과를 보여준다.

측정 결과, 폭증하는 데이터를 전송하는 실험 1의 경우 Fig. 4와 같이 제안하는 기법으로 처리했을 때는 지연시간이 최저 0.3660초에서 최대 0.4921초의 분포를 보이며 평균 0.4045초의 지연시간이 측정되었고, 라운드로빈 방식의 경우에는 0.4896초에서 0.5925초의 분포를 보이며 평균 0.5407초의 지연시간이 측정됨에 따라 본 연구에서 제안하는 방식이 라운드로빈 방식 대비 약 25.2% 적은 지연시간이 발생하는 것을 확인할 수 있었다. 이는 폭증해서 밀려 들어오는 데이터를 분리하여 우선 처리함은 물론, 큐에서도 우선 처리 큐를 운영하여 폭증 데이터의 쏠림과 지연 현상을 방지하는 한편, 그 외 기타 유형의 정상적인 데이터가 큐에서 정체되는 현상을 예방한 결과임을 알 수 있다.

실험 2 (Fig. 5)의 경우에는 무작위 데이터를 전송

Fig. 4. Burst data latency measurement result.

Fig. 5. Random data latency measurement result.

하여 실험하였는데, 여기서도 역시 우리가 제안하는 기법이 라운드로빈 방식보다 빠른 처리속도를 보였다. 제안하는 기법은 평균 0.4195초, 라운드로빈 방식은 평균 0.5327초의 지연시간이 측정됨에 따라 약 21.2%의 지연시간 감소 효과가 있음을 알 수 있는데, 이는 무작위 데이터 중에서도 군집되어 몰려 들어오는 데이터가 일부 발생할 수 있으므로 역시 이러한 군집 데이터를 우선 처리하여 큐에서의 정체 현상을 차단하고 기타 정상적인 데이터의 흐름을 원활히 하였기 때문에 이와 같은 지연시간 감소 효과가 나타났다고 할 수 있다. 또한, 앞서 실험 1의 그래프 Fig. 4에 비해 지연시간의 편차가 비교적 안정적임을 볼 수 있는데, 입력되어 들어오는 데이터 중 군집 데이터 비율이 실험 1에 비해 적기 때문에 나온 결과이다.

4.4.2 폭증 데이터 지연시간 선별 측정(실험3 및 실험 4 참조)

제안하는 부하분산 기법이 폭증 데이터 처리에 효과가 있는지 명확히 확인하기 위해 폭증 데이터를 처리한 결과만 선별하여 지연시간을 측정하였다. 이번 실험에서도 제안하는 기법과 라운드로빈 방식으로 처리한 결과를 비교하였으며, 폭증 데이터를 전송한 결과는 Fig. 6(실험3)과 같이 나타나며, 일반적인 네트워크 상황을 가정하여 측정한 결과는 Fig. 7(실험4)과 같이 나타난다.

측정 결과, 실험 3과 실험 4 모두 전체 데이터의 지연시간을 측정했던 실험 1과 실험 2와 비교하여 제안하는 부하분산 기법의 효과가 크게 나타났다. 폭증 데이터를 전송한 실험 3에서는 제안하는 기법의 경우 평균 0.0887초의 지연시간이 측정되었으며, 라운드로빈 방식은 평균 0.1501초가 측정되어 약 40.9%의 지연시간 감소 효과가 있었다. 무작위 데이터를 입력한 실험 4에서도 지연시간이 각각 0.0769초와 0.1121초가 측정되면서 역시 제안하는 기법이 31.3%의 지연시간 단축 효과가 있음을 확인하였다. 제시한 그래프를 보면 앞서 실험했던 전체 지연시간의 결과와 비교해보았을 때 우리가 제시한 부하분산 기법과 라운드로빈 기법 간의 편차가 큰 것을 확인할 수 있는데, 이는 정상적으로 입력되는 데이터의 처리 결과를 제거하고 폭증하는 데이터 처리만 선별한 결과이기 때문에 폭증 데이터 처리에 있어 본 기법의 실질적인 효과를 보여주는 결과임을 알 수 있다. 이상 4종의 실험 결과를 정리하면 Table 1과 같다.

전체 데이터의 결과를 분석해 보면, 전체 데이터를 전송한 지연시간의 경우 우리가 제안하는 기법이 라운드로빈 방식과 비교했을 때 약 23.2% 더 빨리 처리되었으며, 폭증 데이터만 처리한 결과를 보면 약 36%의 지연시간 감소가 있었음을 확인할 수 있다.

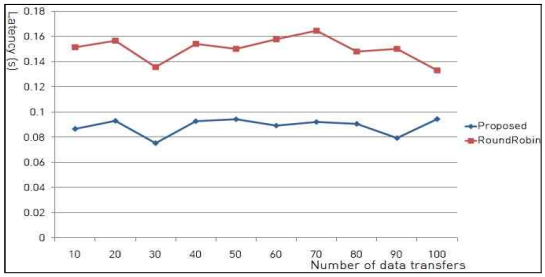

Fig. 6. Burst data latency measurement result.

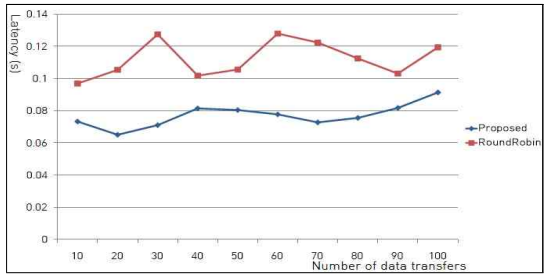

Fig. 7. Random data latency measurement result.

Table 1. Overall measurement result

Load Balancing Method	avg. Latency of overall data (s)		avg. Latency of Priority Data (s)	
	Burst Data (expt 1)	Random Data (expt 2)	Burst Data (expt 3)	Random Data (expt 4)
Proposed	0.4045982	0.4195674	0.0887058	0.0769921
RoundRobin	0.5407142	0.5327434	0.1501101	0.1121644
Ltency reduction	-0.1361160	-0.1131760	-0.0614043	-0.0351724
	25.2%	21.2%	40.9%	31.3%

이는 폭증 데이터 처리에 있어 우리가 제안하는 기법이 효과가 있음을 증명한다. 또한, 무작위 데이터 처리 부분에서도 약 26.3%의 지연시간 감소 효과가 있어 평상시 정상적인 네트워크 환경에서도 우리가 제안하는 기법이 효용 가치가 있다고 할 수 있다.

5. 결 론

본 논문에서는 우리의 인터넷 환경에서 어떤 이슈가 발생했을 때 네트워크 내 특정 데이터가 폭증한다는 가정 아래 폭증하는 데이터를 식별하여 이를 우선 처리하는 부하분산 기법을 제안하였다. 실험 결과를 통해 폭증 데이터의 처리 결과를 살펴보면 우리가 제안하는 기법이 본 연구의 배경과 목적으로 삼았던 데이터가 폭증하여 몰려 들어오는 상황에서 효율적인 부하분산을 통해 효과가 있었음을 확인할 수 있으며, 일반적인 네트워크 환경에서도 역시 지연시간이 감소하는 것을 식별할 수 있었다. 또한, 전체 데이터의 전송 결과를 살펴보더라도 본 논문에서 제안하는 부하분산 기법을 적용하였을 때 유의미한 효과가 있음이 실험을 통해 증명되었다. 이를 통해 네트워크 운영 시 본 연구에서 제안하는 기법을 적용한다면 데이터가 폭증하는 예외적인 상황에서는 물론 평시 정상적인 네트워크 환경에서도 그 효용성이 있다고 할 수 있다.

이러한 부하분산 정책 운용의 기본 배경은 SDN 환경하에서의 구축을 전제로 하고 있다. 본 연구의 실험에서는 네트워크 체계를 단순화하여 비교적 단순한 형태의 네트워크를 시뮬레이션했지만, 이를 실제 환경에 적용한다면 네트워크 운용자는 수많은 네트워크 장비들을 제어해야만 하는 상황에 직면하게 된다. SDN에서는 컨트롤 영역과 데이터 영역의 분리를 통해 컨트롤러와 연동하여 중앙집중적인 네트워크 정책 설정이 가능하기 때문에 기존 네트워크 체계와 비교하여 관리자 측면에서 매우 유리하였고, 우리의 부하분산 정책을 이전보다 빠르고 손쉽게 적용할 수 있는 배경이 되었다. 또한, 그동안 활발한 연구가 이루어지지 않았던 SDN 체계에서의 부하분산 기법을 최근 인터넷 트래픽 환경의 특성에 맞추어 새로운 개념의 기법을 제시하였다는 것에 그 의미가 있다고 할 수 있다.

이를 군 조직에 적용한다면 불시 상황이나 훈련, 전시와 같은 상황에서 특정 데이터에 사용자들의 정보요구가 집중될 때 효과적일 것이며, 네트워크 운용에 유연성을 부여하면서 보다 효율적이고 원활한 전산망을 구축할 수 있을 것으로 보인다. 또한, 본 논문에서 제시한 것과 같이 Middle Box를 구축하여 운용한다면 기존 장비와의 손쉬운 연동은 물론 신규 네트워크 장비를 구축하기 위한 예산과 시간 소요를 최소화할 수 있을 것으로 판단된다. 향후 에는 군 전산망 역시 민간 기업처럼 SDN 도입이 예상되는 가운데 부하분산을 위한 Middle Box와 같은 소프트웨어 컨트롤 개념을 선제적으로 구축하여 운용한다면 SDN으로의 전환에서도 큰 장점이 있을 것으로 기대된다.

REFERENCE

[1] S.H. Baek, I.S. Jang, D.E. Seo, and J.H. Lee, "A New Paradigm for Future Networks," *The Journal of Korean Institute of Communication Sciences,* Vol. 32, No. 7, pp. 82-92, 2015.

[2] H.H. Lim, K.T. Kim, B.J. Lee, and H.Y. Youn "Feature Selection Method for the Classification of Traffic in SDN," *The Journal of Korean Institute of Communications and Information Science*, Vol. 44, No. 1, pp. 106-116, 2019.

[3] Open Networking Foundation White Paper, https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (accessed August 7, 2019).

[4] Y.J. Choi, "Design and Implementation of The EIGRP based on the SDN," *Journal of Korea Multimedia Society*, Vol. 22, No. 2, pp. 178-185, 2019.

[5] SDN Structure, https://www.opennetworking.org/sdn-definition (accessed August 1, 2019).

[6] N.G. Kang, *Unauthorized SW Blocking Method in SDN Environment*, Master's Thesis of Korea National Defence University, 2018.

[7] J.S. Bae and B.H. Lee, "A Virtual Machine Allocation Scheme based on CPU Utilization in Cloud Computing," *Journal of Korea Institute of Information and Communication Engineering*, Vol. 15, No. 3, pp. 567-575, 2011.

[8] M.K. Ji, *The Method of Run Time Prediction Simulation-based Load Balancing*, Master's Thesis of Sungkyunkwan University of Technology, 2015.

[9] S. Kim, *Design and Implementation of Dynamic Load Balancing Model for Distributed Streaming Server*, Master's Thesis of Chonnam National University of Computer Engineering, 2002.

윤 지 영

2007년 가톨릭대학교 행정학 졸업
2007년 공군 소위 임관
2020년 국방대학교 컴퓨터공학 전공 석사

권 태 욱

1986년 육군사관학교 컴퓨터공학과 졸업
1995년 미국 해군대학원 컴퓨터공학과 석사
2001년 연세대학교 컴퓨터공학과 박사