Regular paper | Journal of the KIECS. pp. 1081-1086, vol. 18, no. 6, Dec. 31. 2023, t. 122, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.6.1081

# SDN 환경에서 실시간 데이터 유입형태를 고려한 효율적인 부하분산 기법 연구

김주성* · 권태욱**

# A Study on the Efficient Load Balancing Method Considering Real-time Data Entry form in SDN Environment

Ju-Seong Kim* · Tae-Wook Kwon**

요 약

현대 네트워크의 급속한 성장과 복잡성 증가는 전통적인 네트워크 아키텍처의 한계를 부각시켰다. 이러한 과제에 대응한 SDN(Software-Defined Network)의 등장은 기존의 네트워크 환경을 변화시켰다. SDN은 제어부와 데이터부를 분리하고 중앙 집중식 컨트롤러를 사용하여 네트워크 동작을 조정한다. 하지만 이러한 구조도 최근 수많은 IoT(Internet of Things) 기기의 급속한 확산으로 엄청난 양의 트래픽이 발생하게 되었고 이는 네트워크의 전송 속도를 느리게 할 뿐 아니라 QoS(Quality of Service)를 보장하기 어렵게 만들었다. 이에 본 논문에서는 어느 특정 IP에서 다량의 데이터가 유입되는 경우 즉, 서버 과부화 및 데이터 손실이 발생하게 되어 전체적인 네트워크 지연이 발생할 시 기존의 데이터처리 스케줄링 기법인 RR(Round-Robin) 방식에서 해당 IP와 임의의 서버(처리기)를 Mapping 하는 방식으로 전환하여 데이터를 부하분산하는 기법을 제안하고자 한다.

ABSTRACT

The rapid growth and increasing complexity of modern networks have highlighted the limitations of traditional network architectures. The emergence of SDN (Software-Defined Network) in response to these challenges has changed the existing network environment. The SDN separates the control unit and the data unit, and adjusts the network operation using a centralized controller. However, this structure has also recently caused a huge amount of traffic due to the rapid spread of numerous Internet of Things (IoT) devices, which has not only slowed the transmission speed of the network but also made it difficult to ensure quality of service (QoS). Therefore, this paper proposes a method of load distribution by switching the IP and any server (processor) from the existing data processing scheduling technique, RR (Round-Robin), to mapping when a large amount of data flows in from a specific IP, that is, server overload and data loss.

키워드

SDN(: Software-Defined Network), Load-balancing, RR(Round-Robin), Mapping
소프트웨어 정의 네트워크, 부하 부산, 라운드 로빈, 매핑

* 국방대학교 대학원생(kimjusung@army.mil)
** 교신저자 : 국방교 컴퓨터공학과
· 접 수 일 : 2023. 09. 14
· 수정완료일 : 2023. 10. 28
· 게재확정일 : 2023. 12. 27
· Received : Sep. 14, 2023, Revised : Oct. 28, 2023, Accepted : Dec. 27, 2023
· Corresponding Author : Ju-Seong Kim
Dept. Korea National Defense University,
Email : jusung_kim@naver.com

## Ⅰ. 서 론

최근 네트워크 환경은 소프트웨어 정의 네트워크(SDN)의 등장으로 상당한 패러다임 변화를 경험했다. SDN의 등장은 빠르게 발전하는 현대 애플리케이션 및 서비스의 요구사항에 보조를 맞출 수 있는 보다 프로그래밍이 가능하고 유연하며 확장 가능한 네트워크의 필요성에서 비롯되었다. 이러한 SDN은 제어부를 데이터 평면에서 분리하여 중앙 집중식 네트워크 관리 및 프로그래밍을 가능하게 함으로써 기존 네트워크 아키텍처에 혁신을 가져왔다[1]. 흔히 '폐쇄형' 또는 '단일형' 아키텍처라고 불리는 전통적인 네트워크 아키텍처는 네트워크 장치 내에서 긴밀하게 결합된 제어 및 데이터 평면에 의존했다. 스위치 및 라우터와 같은 네트워크 장치는 패킷 전달과 패킷 라우팅, 트래픽 관리에 대한 모든 결정을 담당했다. 따라서, 유연성, 확장성 및 변화하는 네트워크 요구사항에 적응할 수 있는 능력이 부족했다. 이후 기존 네트워크 아키텍처의 보완으로 제시된 SDN의 등장은 네트워크 인프라에 상당한 이점을 가져왔다. 중앙 집중식 제어 및 관리가 가능하여 네트워크 관리자가 중앙 지점에서 네트워크 정책 및 구성을 정의할 수 있고 네트워크를 전체적으로 파악하고 네트워크 관리를 단순화하며 동적이고 프로그래밍 가능한 네트워크 동장을 가능하게 한다. 하지만, 이러한 SDN에도 과다한 트래픽 발생에 따른 부하분산 관련 연구는 현재까지도 활발히 진행 중이다[2-4]. 이러한 부하분산 문제를 해결하려면 효율적인 알고리즘, 지능형 트래픽 모니터링 매커니즘 등이 지속적인 연구 및 개발 노력이 필요하다.

## Ⅱ. 관련 연구

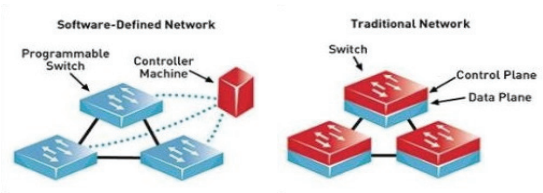
### 2.1 SDN

SDN은 기존 네트워크 환경의 한계에 따른 네트워크 관리 구조에 대한 재검토가 요구되면서 2006년 스탠포드 대학에서 처음으로 제안되었다. 과거에는 네트워크 장비의 제어기능이 H/W에 있었지만 SDN에서는 제어 영역을 H/W에서 분리, S/W를 통해 논리적 또는 가상적 실체로서 네트워크를 관리하고 제어한다. 이는 신속하고 용이하게 가상 네트워크 환경 구축이 가능하고 통합 인터페이스로 네트워크 환경관리가 가능하게 한다. [5]

또한, SDN 기술이 도입되면서 네트워크 장비의 H/W 영역과 S/W 영역이 분리되어 소프트웨어로 구현된 SDN 컨트롤러가 제어기능을 전담하게 되고, 컨트롤러와 스위치 사이에는 오픈소스 기반의 Open Flow를 가지고 통신하게 되면서 기존의 네트워크 체계에 비해 뛰어난 유연성과 확장성을 확보하게 된다[6-7].

표 1. 기존 네트워크 환경과 SDN환경 비교(1)
Table 1. Comparing existing Network to SDN(1)

| Sortation | Legacy System | SDN |
|---|---|---|
| Perspective | HW-Centric | SW-Centric |
| Configuration | HW Vendor | User |
| Openness | Closed | Open |
| Compatibility | Independent | Standard |
| Efficity | Inefficient | Reasonable |
| Acceptance | Vendor needs | Requested |
| Fairness | Monopoly | Fair Competition |




(a) Legacy Network (b) SDN environment

그림 1. 기존 네트워크 환경과 SDN환경 비교(2)
Fig. 1 Comparing existing Network to SDN(2)

### 2.2 SDN에서 부하분산 연구 동향

Client가 특정 이슈에 영향을 받아 편중된 데이터를 요구할 때, Middle Box를 활용하여 데이터의 유형을 분류하고, 그 중 폭증하는 데이터를 우선 처리함으로써 부하를 분산하는 개념을 제시하였다[8-9].

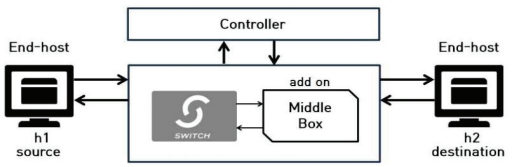



그림 2. Middle Box를 활용한 부하분산
Fig. 2 Load balancing using Middle Box

네트워크 계층으로 유입되는 데이터를 효과적으로 SDN 기술로 부하분산하는 방안을 제시하였는데, 컨트롤러의 정책적용 유연성을 이용하여 사용자로부터 유입된 데이터를 유형별로 분류하고 이를 통해 서버의 응답시간을 기준으로 데이터를 처리하면서 효율성을 높였다.

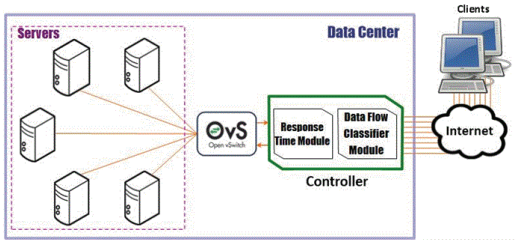



그림 3. 서버의 응답시간 모듈을 적용한 부하분산
Fig. 3 Load balancing with server response time module

SDN 컨트롤러를 이용한 패킷 Flow를 평가하고 특정 유형의 패킷 딜레이, 송·수신률이 전체적인 딜레이와 송·수신률과 괴리되지 않도록 부하분산하는 방안을 제시하였다[10]. 확정된 유입형태를 이용하던 기존 연구와 달리 실제 인터넷 환경과 유사한 트래픽을 가지고 실시간 QoS를 측정하여 부하를 분산하는 방법을 사용하였다.

그 외 서버의 여러 상태를 확인하여 그에 맞는 부하분산 기법을 적용하는 동적 부하분산 방식에 대한 연구 또한 활발하게 이루어지고 있다.

## III. 제안하는 부하분산 기법

### 3.1 실시간 데이터 유입형태에 따른 매핑 기법

기존의 데이터처리 스케줄링 기법인 라운드로빈(RR) 방식은 다수의 클라이언트(IP)에서 데이터 유입 시 데이터의 형태 등에 상관없이 유입되는 순서에 맞춰 서버(처리기)가 데이터를 처리하는 방식이다. 반면, 어느 특정 IP에서 다량의 데이터가 유입되는 경우 서버 과부화 및 데이터 손실이 발생하게 되어 전체적인 네트워크 지연이 발생 되게 된다.

이에, 특정 IP의 폭증 데이터 여부 등을 판단하여 해당 IP를 임의의 서버와 1:1 매핑하는 데이터처리 스케줄링 기법을 제안한다. 이는 다수의 서버(처리기)가 있는 경우 여러 대에 과부하가 발생하여 네트워크 전체의 부하가 발생되거나 데이터 손실율이 발생되는 것보다 1대의 서버(처리기)에 임무를 매핑하여 데이터를 처리함으로써 전반적인 네트워크 부하분산과 지터(Jitter)값을 감소시킬 수 있다. 최초 네트워크 환경에서 패킷 데이터 양의 수준을 실시간으로 모니터링하고 기준치 이상의 데이터 유입시 해당 IP를 확인하고 제안 기법인 매핑 모드로 전환한다. 매핑 모드의 경우 폭증된 IP에서 직접 별도의 서버 처리기로 패킷을 전달하기 때문에 나머지 서버(처리기)에서는 기존의 RR방식으로 동작되는 형태를 유지한다. 그림 4는 데이터 유입 형태를 고려한 부하분산 과정인 위의 내용을 Flowchart로 표현한 것이다.

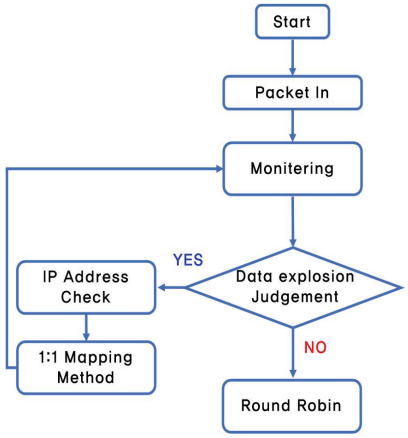



그림 4. 제안하는 기법의 Flowchart
Fig. 4 The flowchart of the proposed technique

### 3.2 동작 방식

SDN에서는 제어부가 데이터 평면과 분리되어 네트워크 관리를 중앙 집중식으로 관리하고 제어할 수 있다. 매핑 부하분산 방식에서 패킷은 소스 IP주소, 대상 IP주소 또는 전송 계층 포트와 같은 패킷 헤더의 특정 필드 또는 필드 조합에서 작동하는 해시 함수를 기반으로 여러 경로에 분산된다. 해시 함수는 해시 값을 생성하며, 이 값은 패킷 전달을 위한 경로로 사용된다.

Controller는 네트워크 상태에 대한 정보를 수신하고 데이터 폭증 판단시 트래픽을 분산하는 제안기법

인 매핑 방식을 결정한다. 들어오는 각 패킷에 해시 함수를 적용하고 전달에 적합한 경로를 결정한다. 이 결정은 해시 함수에 의해 생성된 해시 값을 기반으로 한다. 또한, 선택한 경로를 따라 스위치에 흐름 테이블 항목을 설치한다. 이러한 항목은 특정 특성을 가진 패킷에 대한 전달 동작을 정의한다. 패킷이 스위치에 도착하면 흐름 테이블 항목과 일치하여 적절한 작업을 결정한다. 항목을 기반으로 스위치는 SDN 컨트롤러의 부하 분산 결정에 따라 지정된 출력 포트로 패킷을 전달한다.

### 3.3 Controller의 동작

데이터가 들어오게 되면 실시간 모니터링을 통해 데이터 유입 형태를 판단하고 기준치 이상의 특정 IP를 식별하게 된다. 이를 통해 데이터 폭증을 판단되면 해당 IP주소를 확인하여 1:1로 매핑하는 방식으로 임의의 서버를 지정하여 데이터를 처리한다. 이후, 일정 시간 모니터링 경과 데이터 유입량이 정상화된 경우 기존의 RR 방식으로 전환하여 처리한다.

## Ⅳ. 실험 및 결과분석

### 4.1 실험 환경

제안하는 기법을 구현하기 위한 실험환경 구성으로 논리적 구성을 통한 SDN 환경을 구축하였다. PC기반으로 가상으로 네트워크 환경을 시뮬레이션 할 수 잇는 오픈소스 프로그램을 사용하였다. 표 1은 제안하는 Mininet, Ryu controller의 성능을 평가하기 위한 시스템 구축 환경이다. 그리고 8대의 IP(Client)와 SDN 역할을 해주는 Controller, Switch, 8개의 다중 서버(처리기)를 두어 네트워크를 구성하였다. 네트워크 Topology는 그림 5와 같다

표 2. 시스템 구축 환경

Table 2. System deployment environment

| Name | Operating system | Description |
|---|---|---|
| Ubuntu | Python 3.8.10 | 20.04 |
| Mininet | | 2.3.0 |
| Ryu controller | | 4.34 |

각 호스트는 서버로 일정한 사이즈의 패킷을 지속적으로 보내며 다수의 서버(처리기)에서는 RR방식을 사용한 데이터처리 방식을 보인다. 이후 임의의 호스트에서 기준치 이상의 데이터(1024kb)가 유입될 시 폭증 데이터로 판단하여 해당 IP를 임의의 서버(처리기)와 매핑을 시켜 데이터를 처리한다. 위 시나리오는 표 2와 같다.

표 3. 성능평가 시나리오

Table 3. Performance evaluation scenarios

* (R) = Random

| | End to End Device | Destination | Packet |
|---|---|---|---|
| 1 | Host - Server | All | 1 kb |
| 2 | Host(R) - Server | All | 1024 kb |
| 3 | Host(R) - Server(R) | Server(R) | 1024 kb |

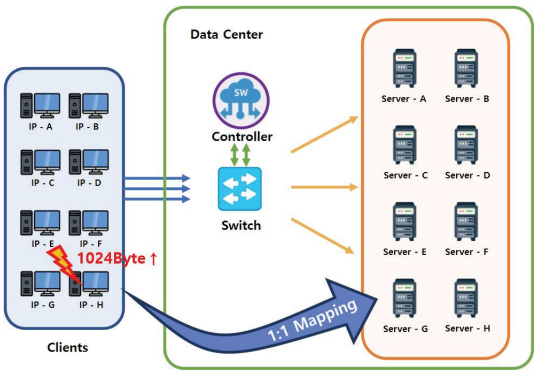



그림 5. 실험 환경 토폴로지

Fig. 5 Experimental environment topology

기본 네트워크 구성을 통해 대용량 패킷이 들어온 횟수(Large Traffic Count), 서버로 들어온 패킷의 수(Total Income Count), 해당 처리기로 들어온 패킷의 전체 사이즈(Total Packet Size), 전체 분포 중 어느 정도 처리수준(%)을 유지하는지(Total Distribution)에 대한 정보를 모니터링 한다.

이후 패킷 데이터의 양을 실시간으로 일정하게 전송하며 RR방식의 데이터 처리수준을 모니터링한다. 실시간으로 'PacketinHandler' 이벤트로 들어온 패킷 프로토콜 정보를 담은 IP는 'get_packet_size' 함수로 매개변수로 받으며 해당 IP의 패킷 사이즈를 확인하여 처리 장치에 매핑 한다.

### 4.2 실험 및 분석

기존 RR방식에서 8대의 서버(처리기)의 데이터 처리상태는 그림8과 같다. 대용량 패킷 발생 시의 데이터 처리상태는 기존 방식에서 데이터 처리율이 소폭 상승하였고 전체적인 처리율이 균등하게 분할되며 RR 방식으로 처리됨을 볼 수 있었다. 다만, 실험환경 제한에 따른 확연한 서버 간 처리율 차이는 확인이 제한되었다.

이어서 제안하는 기법이 적용된 경우 대용량 패킷이 발생한 임의의 호스트는 매핑 방식을 통해 1개 서버에서 온전히 해당 IP의 패킷을 처리함으로써 7개의 서버에서 원할한 데이터 처리상태를 확인할 수 있었다.

RR방식과 제안방식에서의 데이터 처리율(%)의 비교는 그림 6과 같다. 기존 방식에서의 각 서버의 데이터 처리율은 비슷한 수준을 보였다. 반면, 데이터 폭증 시 임의의 서버(그림에서는 서버 8)에서는 해당 폭증 데이터의 IP를 확인하여 매핑함으로써 남은 7대의 서버의 데이터 처리율을 감소시킨 것을 확인할 수 있었다.

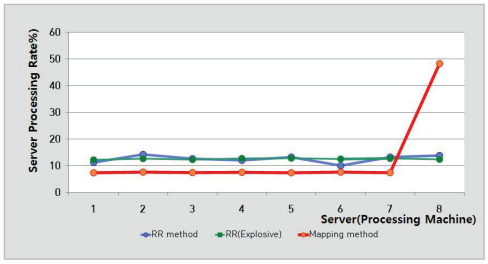



그림 6. 데이터 처리율 비교

Fig. 6 Comparison of data processing rates

또한, 데이터 패킷 처리에 대한 각 서버별 대기시간을 비교해본 결과 표 3과 같이 기존 방식보다 제안 방식의 대기시간이 최대 36.4%의 효과를 볼 수 있었으며 패킷 수와 상관없이 데이터 처리에 대한 대기시간의 감소 효과가 있었다.

표 4. 지연시간 측정결과

Table 4. Latency measurement results

| | avg. Packet quantity | | |
|---|---|---|---|
| | 200 Packet | 500 Packet | 1,000 Packet |
| RR method | 0.0006883 | 0.0005863 | 0.0004737 |
| Proposed | 0.0004371 | 0.0005121 | 0.0004481 |
| Latency reduction | 0.0002512 | 0.0000742 | 0.0000256 |
| | 36.4% | 12.6% | 5.4% |

기존 방식이나 제안 방식 모두 패킷 수별 대기시간에는 큰 차이는 없었으나, 모든 패킷 수별 기존 방식보다 제안 방식이 대기시간이 짧은 결과를 그림 7을 통해 알아볼 수 있다. 추후 연구 결과 범위를 확장한다면 매핑된 서버를 제외한 나머지 서버와의 비교를 통해 더욱 효과적인 결과값을 도출해 낼 것이라 판단된다.

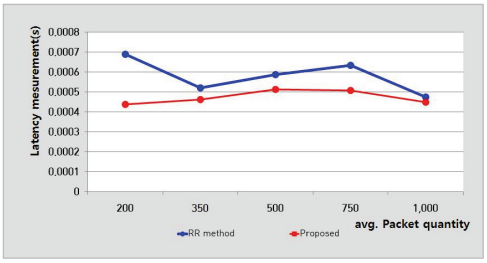



그림 7. 패킷 수에 따른 대기시간

Fig. 7 Latency by number of packets

## Ⅴ. 결론

본 논문에서는 SDN 환경에서의 실시간 데이터 유입형태에 따른 매핑 방식을 통한 부하분산을 연구했다. SDN에 있어서 중앙집중식 관리 및 제어는 네트워크 성능을 좌우하는 핵심적인 요소이다. 해당 논문에서는 선행 연구되었던 부하분산 정책들을 보완하여 데이터처리 스케쥴링 기법의 한가지 방법을 제안하였다. 향후 시뮬레이션을 통해 기존 데이터처리 방식과의 성능 비교값을 확인할 경우 전반적인 데이터처리의 수준을 가늠할 수 있는 지터(Jitter)값에 대해 연구가 세부적으로 이뤄지길 바란다. 지터는 패킷 지연(Delay)이 일정하지 않고 수시로 변하고 패킷 사이의 간격이 일정하지 않는 현상을 의미함으로 여러번 측정을 통한 그 값들의 평균을 의미한다. 즉, 여러 방안의 제안모델의 데이터 처리 수준을 측정 후 지터값을 비교함으로써 전반적인 네트워크 환경의 원할함을 분석하기 용이할 것이라 생각된다. 앞으로도 다양한 고려요소를 바탕으로 데이터 처리에 관련한 추가적인 연구가 필요할 것으로 생각이 되는 가운데 본 연구를 마친다.

## References


[1] N. McKeown,, "OpenFlow: enabling innovation in campus networks", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008, pp. 69-74.

[2] Q. He, X. Wang, and M. Huang, "OpenFlow based low overhead and high accuracy SDN measurement framework", *Transactions on Emerging Telecommunications Technologies*, vol. 29, no 2, 2018, pp. 3263.

[3] Y. C. Wang and S. Y. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks", *IEEE Transactions on Network and Service Management* vol. 15, no. 4, 2018, pp. 1422-1434.

[4] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks", *International Conference on Computer Communications and Networks,* Lahaina, HI, USA, 2011, pp. 1-6.

[5] Wenfeng Xia, "A Servey on Software-Defined Networking", *IEEE COMMUNICATION SURVEYS & TUTORIALS*, vol. 17, Issue. 1, 2015, pp. 27-51.

[6] Diego Kreutz, "Software-Defined Networking: A Comprehensive Survey", *Proceedings of the IEEE*, vol. 103, issue 1, 2014, pp. 14-76.

[7] Yunchun. Li, "MultiClassifier: A combination of DPI and ML for application-layer classification is SDN", *International Conference on Systems and Informatics*, Shanghai, China, 2014, pp. 15-17.

[8] J. Yoon and T. Kwon, "An Efficient Load Balancing Technique Considering Forms of Data Generation in SDNs," *J. of Korea Mutimedia Society*, vol. 22, no. 2, 2020, pp. 247-254

[9] J. Kim and T. Kwon, "Efficient Load Balancing Technique Considering Data Generation Form and Server Response Time in SDN", *J. of the Korea Institute of Electronic Communication Sciences*, vol. 15, no. 4, 2020, pp. 679-686.

[10] J. Yoon and T. Kwon, "Efficient Load Balancing Techniques Based on Packet Types and Real-Time QoS Evaluation in SDN", *J. of the Korea Institute of Electronic Communication Sciences*, vol. 16, no. 5, 2021, pp. 807-816.


## 저자 소개


**김주성(Ju-Seong Kim)**

2012년 육군3사관학교 토목건축공학과 졸업
2022년 ~ 국방대학교 대학원 컴퓨터공학과 재학 중

※ 관심분야 : Next Generation Networking, Soft-defined Network

**권태욱(Tae-Wook Kwon)**

1986년 육군사관학교 전자공학과 졸업(공학사)
1995년 美 해군대학원 컴퓨터공학과 졸업(공학석사)
2001년 연세대학교 대학원 컴퓨터공학과 졸업(공학박사)
2007년~현재 국방대학교 컴퓨터공학과 교수

※ 관심분야 : Next Generation Networking, Content Centric Networking, Software Defined Networking, Network Function Virtualization, U-Sensor Networking, VR, RFID