Regular paper | Journal of the KIECS. pp. 817-824, vol. 16, no. 5, Oct. 31. 2021, t. 109, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2021.16.5.817

SDN 환경에서의 서버 부하 임계치 경고를 통한 효율적인 부하분산 기법

이준영* · 권태욱**

Efficient Load Balancing Technique through Server Load Threshold Alert in SDN

Jun-Young Lee* · Tea-Wook Kwon**

요 약

기존 네트워크 체계의 한계점을 극복하기 위해 등장한 SDN(Software Defined Networking) 기술은 네트워크 장비에서의 HW와 SW의 분리를 통해 기존 체계의 경직성을 해소한다. 이러한 SDN의 특성은 하드웨어 중심의 네트워크 장비를 벗어나 폭넓은 확장성을 제공하며, 다양한 규모의 데이터센터에서의 유연한 부하분산 정책을 제공해준다. 그동안 이러한 SDN의 장점을 데이터센터에 적용한 연구들이 다수 진행되어왔으며 효과를 보여줬다. 기존 연구들에서 주되게 사용된 방식은 서버의 부하를 주기적으로 확인하여 이를 기반으로 부하분산을 수행하는 방식이었다. 이 방식에서는 서버의 수가 많고 서버 로드 확인 주기가 짧을수록 트래픽이 증가한다. 본 논문에서는 이러한 제한사항을 해소하기 위해 서버에서 특정 수준의 부하 발생 시 이를 컨트롤러로 보고하는 방식을 통해 불필요한 트래픽을 없애고 서버들의 자원을 보다 효율적으로 관리할 수 있는 새로운 부하분산 기법을 제안한다.

ABSTRACT

The SDN(Software Defined Networking) technology, which appeared to overcome the limitations of the existing network system, resolves the rigidity of the existing system through the separation of HW and SW in network equipment. These characteristics of SDN provide wide scalability beyond hardware-oriented network equipment, and provide flexible load balancing policies in data centers of various sizes. In the meantime, many studies have been conducted to apply the advantages of SDN to data centers and have shown their effectiveness. The method mainly used in previous studies was to periodically check the server load and perform load balancing based on this. In this method, the more the number of servers and the shorter the server load check cycle, the more traffic increases. In this paper, we propose a new load balancing technique that can eliminate unnecessary traffic and manage server resources more efficiently by reporting to the controller when a specific level of load occurs in the server to solve this limitation.

키워드

Data Center, Load Balancing, SDN(Software Defined Networking), Server Load Monitoring
데이터센터, 부하분산, 소프트웨어 정의 네트워크, 서버 부하 모니터링

* 국방대학교 석사과정(ljy07@mnd.go.kr)
** 교신저자 : 국방대학교 컴퓨터공학과 교수
· 접 수 일 : 2021. 07. 22
· 수정완료일 : 2021. 09. 03
· 게재확정일 : 2021. 10. 17
· Received : Jul. 22, 2021, Revised : Sep. 03, 2021, Accepted : Oct. 17, 2021
· Corresponding Author : Tea-Wook Kwon
Dept. Computer Science, Korea National Defense University
Email : ljy07@mnd.go.kr, kwontw9042@kndu.ac.kr

Ⅰ. 서 론

글로벌 콘텐츠 네트워크(Content delivery network)를 운영하는 기업 아카마이에 따르면 코로나-19의 여파로 비대면(언택트) 문화가 보편화하면서 재택근무, 인터넷쇼핑, OTT 콘텐츠 소비 등 온라인이 생활의 중심으로 자리 잡으며 온라인 트래픽이 빠르게 증가하면서 '19년 3분기 대비 '20년 3분기 트래픽 증가율은 200%로 폭증하였다[1].

트래픽의 폭증은 기존의 네트워크 체계로는 감당하기 힘든 상황이 발생하고 있다. 이를 해소하기 위해 네트워크의 지속적인 확장이 요구되고 있으나 기존 네트워크 체계는 벤더별 제공되는 하드웨어 중심의 일관된 정책 적용으로 이종 간 유연한 확장과 사용자의 요구에 유연하게 대응이 제한되는 등 시스템의 확장이 요구될수록 그 한계가 더욱 드러나게 된다. 기존 네트워크 체계의 이러한 한계점을 극복하기 위해 SDN(Software Defined Network)이 등장했다[2].

SDN은 하드웨어와 소프트웨어 기능이 분리되어 소프트웨어를 통해 논리적인 구조를 형성함으로써 기존 체계보다 사용자 요구를 충족할 수 있는 유연한 정책변화와 다양한 확장성을 제공한다. 이러한 점은 방대하고 다양한 유형의 데이터를 생성하는 4차 산업혁명의 데이터 부하분산에 적합하다[3].

데이터센터는 4차 산업혁명 시대의 관련 산업 활성화를 위한 데이터의 처리, 유통 및 저장을 수행하는 핵심 인프라로써 중요성이 증대되고 있으며, 근래의 데이터센터는 대규모의 서버 클러스터를 가상화하여 운영하기 때문에 가상화에 바탕을 둔 SDN을 활용한 부하분산이 데이터센터 전체 효율의 높이는 데 큰 역할을 할 것으로 기대된다[4].

앞선 SDN를 기반으로 한 부하분산 연구들은 효율적인 부하분산을 위해 노력하였으나, 컨트롤러-서버 간 일부 비효율적인 데이터 처리가 발생한다.

본 논문에서는 컨트롤러-서버 간 발생하는 비효율적인 데이터 처리를 최소화하기 위해 임계수준의 부하 발생 시 서버가 경보를 SDN 컨트롤러로 전달하는 방식을 활용하여 컨트롤러-서버 간의 주기적 트래픽을 줄이고 서버의 임계수준의 부하 발생 시 이를 즉각 컨트롤러로 전달하여 서버 상태를 보다 실시간으로 모니터링 할 수 있는 부하분산 기법을 제안한다.

본 논문의 구성은 다음과 같다. 제2장에서는 관련 연구 및 문제점에 대하여 알아보고, 제3장에서는 본 논문에서 제안하는 서버 임계치 경고를 통한 부하분산 방안에 관해 설명한다. 제4장에서는 제안기법을 구현하고 대조기법과 제안기법의 성능을 실험하고 결과를 분석하였다. 마지막으로 제5장에서는 결론을 소개한다.

Ⅱ. 관련 연구

2.1 SDN

SDN은 네트워크 장비에서 하드웨어와 소프트웨어 기능을 분리하는 것이 핵심이다. 하드웨어 영역은 Data Plane으로써 데이터를 전송하는 역할을 한다. 소프트웨어 영역은 Control Plane과 Application Plane으로 나뉘고, Control Plane은 네트워크에 대한 정책이 적용되는 영역으로 볼 수 있으며, Application Plane은 정책 적용을 위한 사용자 지원 영역이다. Control Plane은 SDN 컨트롤러로 구현되며, 논리적으로 중앙 집중화하여 프로그래밍할 수 있도록 고안되었고, 이를 통해 네트워크의 제어나 흐름, 혼잡 제어 등에 유연성을 제공한다[5].

기존 체계에서 각각의 네트워크 장비별로 이루어지던 Control Plane의 역할이 SDN에서는 하나의 SDN 컨트롤러를 통해 전체적인 서버 클러스터의 부하를 측정, 스위치 제어를 통해 네트워크 영역에서의 유연성을 발휘할 수 있다. 이러한 소프트웨어 기반의 SDN 특징을 통해 데이터센터의 서버 부하분산에 적용하는 많은 연구가 진행되고 있다[6].

2.2 데이터센터와 서버 클러스터

데이터센터는 조직의 방대한 정보저장을 위한 서버, 네트워크 회선 등을 제공하여 데이터를 안정적으로 통합·관리하는 인프라 시설이다. 가상화 기술 발전과 클라우드 등의 서비스 제공과 함께 빠르게 발달 중이다[7]. 데이터센터는 웹을 기반으로 하는 다양한 서비스 제공을 위해 부하분산 목적의 서버 클러스터를 구축하고 있으며, 데이터 사용량이 늘어날수록 서버는 고집적화되고 있다. 고용량의 서버가 집적된 데이터센터에서 부하분산은 매우 중요한 부분이며, SDN을 기반으로 S/W를 이용해서 서버 클러스터를

제어하고 활용하는 것은 데이터센터 효율을 높이는 데 필수적이다[3].

2.3 SDN에서의 부하분산 기법

[8]은 데이터센터로 유입되는 패킷 형태와 특정 이슈에 영향을 받아 편중된 데이터를 요구할 때, 미들박스를 활용 데이터의 유형을 분류하고, 그중 폭증하는 데이터를 서버 응답시간을 고려하여 우선 처리함으로써 부하를 분산하는 개념을 제시하였다.

미들박스를 통한 폭증 데이터 분류 세부 정책과 서버와의 응답시간을 SDN 컨트롤러를 통해 부하 분산함으로써 SDN 환경에서 그 효과가 극대화되도록 하였다. 이 기법은 외부로부터 여러 사용자의 요청을 처리하는 데이터센터에 적용하면 부하를 분산시키는 데 효과적이다. 이 방법은 SDN의 장점을 활용하여 부하를 효과적으로 분산시켰다.

[9]은 컨트롤러를 통해 웹서버 부하분산에 최적화된 부하분산 요인으로 서버의 CPU, Memory, 연결 가능 세션 수를 매개변수로 선택하였으며, 매개변수별 중요도에 따라 가중치를 부여하여 서버의 부하를 판단하였다. 컨트롤러가 서버의 부하를 정기적으로 모니터링하고 서버의 부하에 따라 3가지 계층으로 분류하여 부하 분산하는 기법을 제안하였다. 이를 통해 운용하는 서버 클러스터의 특징에 맞춰 관리자는 이를 통해 유연성 있는 부하분산 전략 기법을 제안했다.

그 외 [10]은 서버의 부하 판단을 위해 CPU idle, Free Memory를 변수로 선정하여 매개변수별 중요도에 따라 가중치를 부여하여 서버의 부하분산을 수행하였으며, [11]는 SDN 컨트롤러 내 모듈을 배치하여 서버 응답시간을 수집하고, 스위치 포트 트래픽 정보를 수집하여 서버를 선택하는 일련의 절차를 통해 부하분산을 수행하는 방안을 제안하였다.

위의 기존 연구들은 서버의 응답시간 측정을 위해 주기적으로 컨트롤러-서버 간 별도의 메시지가 필요하게 되며, 그로 인해 비효율적인 트래픽이 발생한다.

III. 제안사항

3.1 제안하는 기법의 동작방법

제안기법은 기존 연구들에서 서버의 부하측정을 위해 발생하는 비효율적인 트래픽 교환의 최소화를 목표로 하며, 이를 위해 컨트롤러와 서버의 주기적 통신이 아닌 서버의 특정 임계치 도달 시에만 서버에 의해 경고를 컨트롤러로 전송하는 기법을 제안한다.

이를 위해 서버는 특정 임계치 이상의 부하 발생 시 이를 ① 서버에서 임계치 초과에 관한 정보를 Control Packet을 통해 컨트롤러로 전달하고, ② 컨트롤러는 서버로부터 수신된 Control Packet 정보를 확인하여 이를 기반으로 등록된 서버의 상태정보를 수정한다. 이후 부하가 임계치에 도달한 서버를 스위치의 포워딩에서 제외하고 정상 서버를 대상으로 요청을 할당한다. 그림 1, 그림 2는 제안기법에 대한 시스템 설계와 알고리즘을 표현한다.

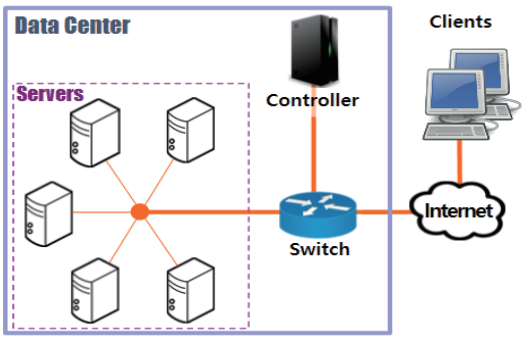

그림 1. 시스템 설계
Fig. 1 System design

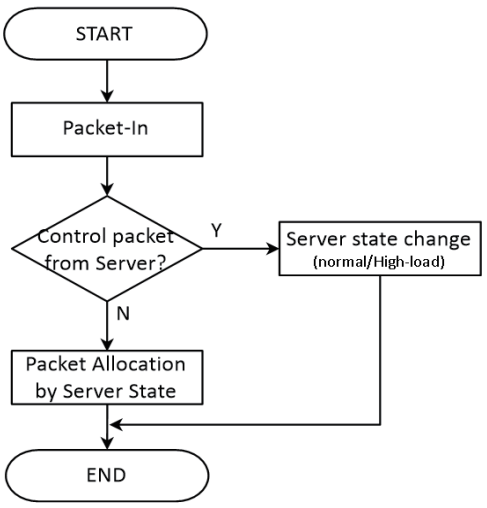

그림 2. 제안하는 기법의 알고리즘 순서도
Fig. 2 Algorithm flowchart of proposed technique

3.1.1 컨트롤러의 동작

시스템이 시작되면 컨트롤러는 각 서버로부터 기본 정보를 담고 있는 Control Packet을 수신하게 되고 이를 기반으로 서버 목록을 생성한다. 사용자들로부터 요청 패킷이 수신되면 등록된 서버들을 대상으로 Round Robin 방식으로 부하분산을 수행한다. 이후 특정 서버로부터 상태변경에 관한 Control Packet(state : 0, 1) 수신 시 컨트롤러는 해당 서버 상태를 변경하고 부하가 임계치에 도달한 서버를 스위치의 포워딩에서 제외하고 정상 서버를 대상으로 요청을 할당한다.

3.1.2 서버의 동작

시스템이 시작되면 서버는 Control Packet을 보내 컨트롤러에 서버 목록으로 등록하고, 사용자 요청 패킷 처리를 수행한다. 서버의 부하가 설정된 특정 임계치에 도달 시 서버는 본인의 상태변경에 관한 사항을 Control Packet(State : 1, High-load)을 통해 컨트롤러로 전송한다. 이후 기존 데이터의 처리가 완료되어 부하가 임계치 이하로 하락하게 될 시 서버는 다시 상태변경에 관한 사항을 Control Packet(State : 0, Normal)을 통해 컨트롤러로 전송한다.

서버의 부하에 관한 판단은 아래 식을 따른다[12].

$$SL_i = (W_1 \times C_i) + (W_2 \times M_i) \quad (1)$$

if (SL_i > Threshold) then

$$\text{Alarm\_Message}i = 1 \quad (2)$$

Else

$$\text{Alarm\_Message}i = 0 \quad (3)$$

i : 서버 번호

SL_i : 각 서버의 자원에 대한 부하 상태

C_i : CPU 사용율 ($0 \leqq CPUi \leqq 1$)

M_i : 메모리 사용율 ($0 \leqq$ 메모리$i \leqq 1$)

$W_{1\sim2}$: 부하 판단 요소별 가중치

Threshold : 서버의 부하 판단 임곗값

식 (1)은 각 서버의 자원 값에 요소별 가중치를 적용한 값이다. 해당 계산을 통해 도출된 값을 기준으로 부하를 판단한다. 식 (2)는 경고 메시지를 보낼 수 있는 상태이다. 각 서버의 자원을 통해 연산된 값이 설정된 임곗값과 같거나 클 때 경고 메시지를 컨트롤러에 보내게 된다. 이는 현재 자기 서버에서 부하량이 많을 경우를 말한다. 식 (2)을 만족하면 서버는 자신의 상태를 High Load로 변경하고 경고 메시지의 값은 1을 전달하게 된다. 이후 서버의 자원이 설정된 임곗값보다 낮아지면 식 (3)에 따라 본인의 상태를 Normal로 변경하고 컨트롤러로 메시지 값 0을 전달한다.

Ⅳ. 실험 결과

4.1 시스템 구현

서버 부하 임계치 경고를 통한 부하분산의 효과를 확인하기 위해 패킷 생성기 역할을 하는 Client와 서버 클러스터의 부하상태 정보를 수집하고 부하분산 정책을 적용하는 컨트롤러, 수신받은 부하분산 정책을 통해 Client의 요청을 포워딩하는 스위치, Client의 요청을 수신하고 처리하는 서버를 Riverbed Modeler로 구축하였다.

4.2 실험 환경

제안하는 부하분산 기법을 가상 네트워크 환경에서 구현하여 효과를 평가하였다. 실험 환경은 CPU Intel (R) Core(TM) i3-8100 @ 3.60GHz, OS Windows 10, RAM 8GB, Riverbed Modeler 18.8.0을 사용하였다.

4.3 실험 방법

실험은 그림 1의 시스템 설계에 따른다. 실험은 대조기법과 제안기법의 서버별 부하율, 응답시간, 컨트롤러 - 서버 간 부하정보 교환을 위한 트래픽 이 3가지를 통해 제안기법의 효율을 측정한다. 본 논문에서는 특정 서버 자원에 중점을 두지 않고 CPU와 메모리 가중치를 1:1로 설정하였으며, 서버별 임계치는 70%로 설정하였다.

처음으로 서버별 부하율 측정은 대조기법(Round Robin)과 제안기법의 부하분산 기법을 적용하여 서버 클러스터의 평균 부하율이 최대 75% 수준으로 유지되도록 패킷을 증가시키면서 각 서버의 부하율을 측정한다. 서버의 부하율을 기반으로 부하분산을 수행하는 제안기법의 효과를 확인하기 위해 첫 번째 서버의 성능치를 기준으로 두 번째 서버부터 5%씩 증가하면서 임의의 성능 차등을 부여함으로써 제안기법의 균등한 자원 활용적인 측면을 검증한다.

두 번째로 제안기법과 대조기법(Round Robin)의 응

답시간을 측정하며, 마지막으로 컨트롤러-서버 간 발생하는 패킷을 제안기법과 주기적 통신을 요구하는 부하분산 기법과 비교한다. 마지막 실험의 대조기법은 Round Robin의 경우 컨트롤러-서버 간 별도의 통신을 수행하지 않기 때문에 주기적 서버 상태 확인 기법과의 비교를 통해 이를 비교하는 방식을 선택하였다.

4.4 서버별 부하율 측정

첫 번째 실험에 관한 결과로써 각각 제안기법과 대조기법(Round Robin) 적용 시 각 서버의 부하율을 측정한 결과는 그림 3, 그림 4, 표 1, 표 2와 같다. 제안기법과 대조기법의 각각 최대 부하율은 70.9226%, 75.2126%로 최대 부하율에서 약 5%의 부하분산 효과를 나타냈으며, 서버별 표준편차의 경우 제안기법은 7.6628%, 대조기법은 8.5707%로 제안기법이 대조기법보다 약 12%가량 향상된 부하 균등효과를 보였다. 이는 제안기법의 경우 특정 서버가 설정된 임계치에 도달 시 상태를 High Load로 컨트롤러에 보고하게 되고, 이후 컨트롤러는 Normal 상태의 서버들을 대상으로 부하분산을 수행하여 서버 간 자원 사용을 균등하게 유지한다. 대조기법의 경우 서버의 부하를 고려치 않고 들어오는 요청 순서에 따라 서버에 할당하여 서버 간 부하율의 차이가 지속하여 발생하게 된다.

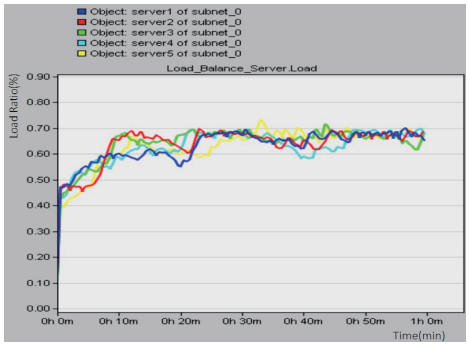

그림 3. 제안기법의 서버별 부하율 측정
Fig. 3 load by server of the proposed method

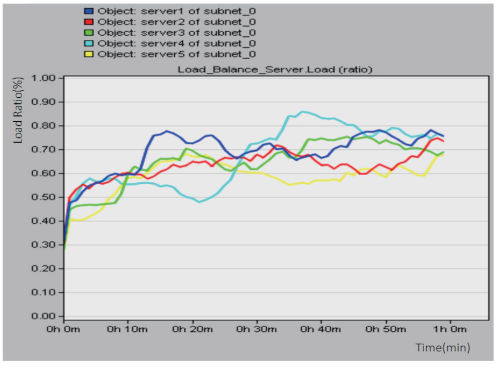

그림 4. 대조기법의 서버별 부하율 측정
Fig. 4 load by server of Round Robin

표 1. 제안기법의 서버별 부하율 측정 결과
Table 1. Load for server of the proposed method

Object	Minimum	Average	Maximum	Std Dev
server1	0.13095	0.62747	0.69998	0.070744
server2	0.12729	0.6343	0.69678	0.078218
server3	0.13295	0.63874	0.71767	0.074444
server4	0.10764	0.62554	0.69816	0.074841
server5	0.12717	0.625	0.73354	0.084897
Total Avg	0.1252	0.63021	0.70922	0.076628

표 2. 대조기법의 서버별 부하율 측정결과
Table 2. Load for server of Round Robin

Object	Minimum	Average	Maximum	Std Dev
server1	0.30139	0.68534	0.7805	0.090642
server2	0.29289	0.66246	0.79156	0.098239
server3	0.27765	0.66578	0.73775	0.082266
server4	0.26969	0.63716	0.74871	0.078986
server5	0.26079	0.61607	0.70211	0.078405
Total Avg	0.28048	0.65336	0.75212	0.085707

4.5 응답시간 측정

제안기법과 대조기법의 응답시간을 측정한 결과는 그림 5, 표 3과 같다. 해당 실험에서 대조기법이 제안기법보다 빠른 응답시간을 보였으나 평균 응답시간에서 10ms, 최대 응답시간에서 1ms의 근소한 차이를 나타내었으며, 일부 구간에서는 제안기법의 응답시간이 단축됨을 확인하였다.

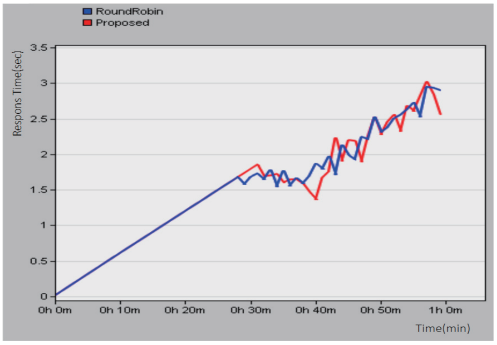

그림 5. 제안기법과 대조기법의 응답시간 측정
Fig. 5 Response Time of Proposed method and RR

표 3. 제안기법과 대조기법의 응답시간 측정결과
Table 3. Response Time of Proposed method and RR

Category	Minimum	Average	Maximum
Proposed	0.54342	1.42546	3.5194
Round Robin	0.54342	1.41536	3.5185

응답시간의 근소한 차이는 제안기법의 경우 기본적으로 임계치 전후로 Round Robin 방식을 사용함으로 기존의 대조기법과 응답시간에서 큰 차이를 보이지 않는 것으로 판단된다. 다만, 임계치 이상의 서버가 발생 시 대조기법보다 적은 수의 서버를 대상으로 부하분산을 수행하게 됨에 따라 응답시간의 일부 손실이 발생할 수 있다.

제안기법의 응답시간이 단축된 일부 구간은 대조기법에서 특정 서버에 부하가 집중되는 시점으로 제안기법의 효율적인 서버 자원 활용의 효과로 판단된다.

4.6 컨트롤러 - 서버 패킷 측정

마지막 실험 결과는 Server로부터 컨트롤러가 수신한 Control Packet의 수를 측정한 결과이며 그림 6과 같다. 컨트롤러에서는 Control Packet을 시뮬레이션 시작 후 서버의 부하가 임계치에 도달하는 순간(12분경)을 시작으로 시뮬레이션 종료까지 총 458회의 패킷을 수신한다.

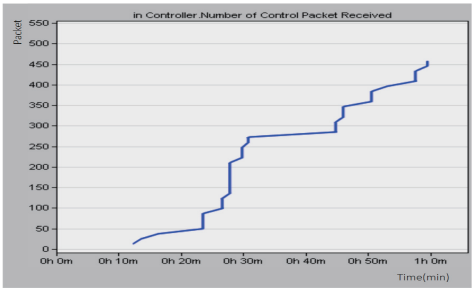

그림 6. 컨트롤러에서 수신한 Control Packet 수
Fig. 6 number of control packets received

표 4. 제안기법과 대조기법의 컨트롤러-서버 패킷 측정
Table 4. Controller-Server Packet of Proposal and Periodic Check Techniques

Category	transmission Interval		
	5sec	30sec	1min
Periodic Check	7,200	1,200	600
Proposed	458		

일반적인 로드밸런서에서는 서버의 상태 및 부하를 확인하기 위해 기본값 5초, 장애판별을 위한 재시도 횟수 3회로 설정한다. 일반적인 주기와 확인 주기를 더욱 늘렸을 경우를 비교한 내용은 표 4와 같다.

본 논문의 제안기법은 Server에서 매개변수에 대한 부하를 연산을 마친 후 상태(High Load : 1, Normal : 0)만 통보하는 방식을 사용하고 있으나, [8]의 경우 Server로부터 매개변수 정보를 Controller가 취합해야 하므로 Server의 CPU idle, Free Memory 등의 정보를 패킷에 포함해야 한다. 그로 인해 제안기법보다 패킷에 포함해야 하는 정보의 양이 증가하게 되며 이는 컨트롤러-서버 간 데이터 처리량이 증가함을 나타낸다.

V. 결론 및 향후 연구

본 논문에서는 SDN 환경에서, 서버로부터의 부하정보 보고를 기반으로 컨트롤러에서 서버 상태를 변경하고 이를 반영하여 부하분산을 수행하는 방안을 제시하였다. 실험 결과와 같이 제안기법은 Round Robin과 비교하여 서버 간 부하 차이를 줄이는 데 효과적이다. 또한, 응답시간을 비교한 결과 Round Robin 기법과 QoS 측면에서 큰 차이가 없음을 확인

하였다. 또한, 부하분산 정책 변경을 위해 서버의 부하 상태를 주기적으로 확인하는 기법과 비교했을 때 컨트롤러-서버 간의 트래픽이 효과적으로 감소함을 확인하였다.

이를 통해 제안기법은 서버 클러스터의 규모가 커질수록 효율성이 더욱 증가할 것으로 기대된다. 특히 본 논문의 내용은 SDN 환경에서 구현되어 네트워크 관리자가 일정 수준의 트래픽이 유지되는 서버 환경과 같은 특정 상황에서 선택할 수 있는 하나의 옵션이 될 수 있으며 이를 통해 네트워크의 트래픽 관리에 유연하게 대처할 수 있을 것이다.

향후 연구로, 데이터센터 내 서버 클러스터별 특성에 맞는 서버 부하 요소별 정정한 가중치 판단과 임계치 이후 동적 부하분산 정책을 적용하는 하이브리드 기법의 효과를 확인하고자 한다.

References

[1] Martin McKeay, “Adapting to the Unpredictable,” *Akamai, State of the Internet*, vol. 7, Issue 1, 2021.

[2] B. Yoon, “Future Networking Technology of SDN,” *Electronics and Telecommunications Trends*, vol. 27, no. 2, 2012.

[3] D. Min, “Market Trends of SDN/NFV Supply and Demand,” *Electronics and Telecommunications Trends*, ETRI, 2016.

[4] IDC, “Worldwide Data center Software-Defined Networking Forecast, 2019-2023,” Nov, 2019

[5] J. Jang "SDN/NFV Military application plan (Based on the Defense Integrated Data Center(DIDC)),” *The Korea Institute of Electronic Communication Sciences*, vol.15, no.4, 2020, pp. 687-694.

[6] M. LEE "Implementation of a Platform for the Big Scientific Data Transfers,” *The Korea Institute of Electronic Communication Sciences*, vol.13, no.4, 2018, pp. 881-886.

[7] J. Holusha, "Commercial Property/Engine Room for the Internet; Combining a Data Center With a 'Telco Hotel,” New York Times, 2019.

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

[9] P. Deepalakshmi, "D-Serv LB: Dynamic server load balancing algorithm,” *International Journal of Communication Systems*, vol. 32, Issue. 1, 2019, pp. 293-310.

[10] M. Yang, "Research on Load Balancing Algorithm Based on the Unused Rate of the CPU and Memory,” *International Conference on Instrumentation and Measurement, Computer, Communication and Control(IMCCC)*, 2015, pp. 542-545.

[11] S. Kiarash, “SD-WLB: An SDN aided mechanism for web load balancing based on server statistics,” *ETRI*, vol.41, Issue. 2, 2018, pp. 197-206.

[12] B. Lee "A Study on Efficient Load Balancing Mechanism in Distributed Web Cluster System,” *Korea Society of Computer Information*, vol.16, no.8, 2011, pp. 11-18.

저자 소개

이준영(Jun-Young Lee)

2011년 한서대학교 항공기계학과 졸업(공학사)
2020년 ~ 현재 국방대학교 대학원 컴퓨터공학과 석사과정

※ 관심분야 : 네트워크, SDN

권태욱(Tae-Wook Kwon)

1986년 육군사관학교 컴퓨터공학과 졸업(공학사)
1995 미국 해군대학원 컴퓨터공학(공학석사)
2001년 연세대학교 컴퓨터공학과(공학박사)
2007년 ~ 현재 국방대학교 컴퓨터공학과 교수

※ 관심분야 : 네트워크, Sensor Networking, CCN, SDN, NFV