논문 21-46-08-08 The Journal of Korean Institute of Communications and Information Sciences '21-08 Vol.46 No.08
https://doi.org/10.7840/kics.2021.46.8.1270

로드밸런서 헬스체크 성능 향상 방안

전 수 철•, 주 양 익*, 박 민 호°

Improvement of Load Balancer Health Check

Soo-Cheol Jeon•, Yang-Ick Joo*, Minho Park°

요 약

많은 사용자들의 요청으로 인한 과부하를 여러 대의 서버로 분산해 주는 역할을 하는 로드밸런서는 서버 장애로 인한 서비스 중단 방지를 위해서 연결되어있는 서버들의 헬스체크(Health Check)를 수행한다. 지금까지 로드밸런서에서 사용하고 있는 헬스 체크 방식은 일정한 주기로 전송하는 헬스체크 패킷을 사용하여 서버의 상태를 판단한다. 사용자의 설정에 따라 ARP, ICMP, TCP, Script 등 다양한 종류의 패킷을 지원하며, 로드밸런서는 서버의 상태를 식별하고 장애 발생 시 사용자의 개입 없이 장애가 발생한 장비를 제외함으로써 끊김 없는 서비스를 지원한다. 그러나 현재 로드밸런서에서 제공하고 있는 헬스체크 방식은 설정한 주기로 장애를 감지하며 설정된 횟수 이상 장애가 감지되어야 장애복구를 시도하기 때문에 오랜 장애감지 시간을 갖는 문제점을 가지고 있다. 따라서 본 논문에서는 서버와 연결된 인터페이스에서 수신하는 패킷 통계를 사용하여 장애 상황을 빠르게 감지할 수 있는 헬스체크 방안을 제안한다.

Key Words : Network Management, Health Check, Load Balancer, Failure Detection, L4 Switch

ABSTRACT

Load balancers, which distribute traffic to multiple servers, perform health checks on connected servers to prevent service disruptions caused by server failures. So far the health check methods have used health check packets that are transmitted at a certain interval to determine the status of the server. Depending on the user's settings, various types of packets such as ARP, ICMP, TCP, Script, etc. are supported, and the load balancer supports uninterrupted service by identifying the status of the server and excluding equipment that has failed without user intervention in the event of a failure. However, the health check method currently provided by the road balancer has a problem of having a long disability detection time. Therefore, in this paper, a health check method is proposed that can detect the failure situation quickly using packet statistics received from the interface connected to the server.

Ⅰ. 서 론

인터넷의 확산과 빠른 성장을 통해 많은 사용자들이 시간과 장소에 관계없이 노트북, 핸드폰, 태블릿 등 다양한 기기를 통해 네트워크에 연결되어 다양한 서비스를 사용한다. 사용자들의 접속량은 빠르게 증가하고 있으며 많은 사용자를 수용하기 위해서 다수의 서버를 사용한다. 대부분은 어느 한 서버로 부하가 집

※ 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2020R1F1A1076795).
• First Author : Department of IT Convergence, Soongsil University, jesc1249@naver.com, 정회원
° Corresponding Author : School of Electronic Engineering, Soongsil University, mhp@ssu.ac.kr, 종신회원
* Corresponding Author :Division of Electrical and Electronics Engineering, Korea Maritime and Ocean University, yijoo@kmou.ac.kr, 종신회원
논문번호 : 202104-083-B-RN, Received April 16, 2021; Revised May 6, 2021; Accepted May 11, 2021

중되는 것을 방지하기 위해서 로드밸런서를 통한 부하분산을 하게 된다[1]. 만약 로드밸런서가 장애가 발생한 서버에 사용자 요청을 전달한다면 사용자는 서비스 장애를 경험하기 때문에. 로드밸런서는 연결되어 있는 서버들이 장애 없이 정상동작을 하는지 확인해야 한다. 이러한 방법을 헬스체크라고 하며 헬스체크에는 Layer 별로 다양한 종류가 있어서 용도에 따라 적절한 것이 선택되어야 한다[2].

헬스체크가 중요한 이유는 인터넷 사용자들의 인내심이 크지 않기 때문에, 비교적 짧은 시간 동안이라도 서비스를 받지 못하면 사이트 방문을 포기하기 때문이다. 사용자들이 인터넷을 사용할 때 웹사이트 장애로 페이지 응답이 1초 지연되면 전환율이 7% 감소하며. 사용자는 2초 이내에 페이지가 로드될 것을 예상하므로 3초 이상 걸리는 페이지는 포기한다[3,4]. 그러나 현재 로드밸런서에서 제공하고 있는 헬스체크 방식은 설정한 주기로 장애를 감지하며 설정된 횟수 이상 장애가 감지되면 장애 복구를 시도하기에 수초의 대기시간을 갖는 문제점을 가지고 있다. 따라서 본 논문에서는 기존의 헬스체크 방식의 문제점들을 해결하여 장애감지 효율성을 향상하는 방안에 관해 연구하고자 한다.

Ⅱ. 관련 연구

2.1 Health Check

헬스체크 (Health Check)는 로드밸런싱 서비스를 제공하는 다수 서버의 상태를 점검하기 위해 사용되는 기술로 서버의 상태를 주기적으로 점검하여 서버의 상태가 통신이 불가능할 경우 서버를 서비스에서 제외하여 서비스를 원활하게 제공하기 위해 이용되는 방법이다[5]. 서버의 상태를 주기적으로 점검하는 방법으로는 Link, ARP, ICMP, TCP, Script 방식이 있으며, 헬스체크에서는 서버로 일정 시간 간격으로 서비스에 대하여 Open과 Close를 반복적으로 진행하여 서비스 제공 가능 유무를 점검한다[6].

헬스체크를 위해 사용되는 패킷의 종류, 패킷을 전송하는 주기와 서버의 응답을 기다리는 시간, 응답이 수신되지 않았을 때 다시 패킷을 보내주는 횟수는 사용자의 설정에 따라 동작한다. 설정할 때 고려되어 할 사항은 전송 주기와 재전송 횟수다. 전송 주기는 장애를 판단하기 위해 패킷을 전송하는 주기로 전송 주기가 짧을수록 서버의 장애 여부를 정확하게 파악할 수 있다 하지만, 헬스체크 패킷이 네트워크의 부하가 될 수 있으므로 네트워크 상태에 따라 적절한 값이 설정

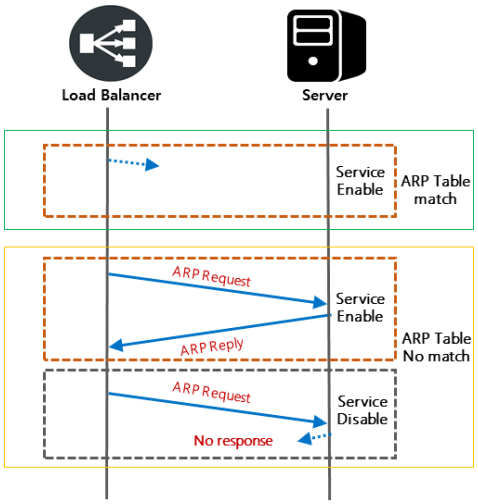


그림 1. ARP 헬스체크
Fig. 1. ARP Health Check

되어야 한다[7]. 재전송 횟수는 장애를 판단하기 위한 헬스체크 패킷 재전송 횟수로 응답을 보내지 못한 서버의 상태를 더 확실하게 확인하기 위해 추가로 헬스체크를 시도하는 횟수다.

2.1.1 Link Health Check

Link 헬스체크는 인터페이스의 상태를 이용하여 서버의 상태를 점검하는 1계층 헬스체크 기능이다. 서버와 연결된 인터페이스의 UP/DOWN 상태에 따라서 서비스 서버의 서비스 가능 유무를 판단한다[8].

2.1.2 ARP Health Check

ARP 헬스체크는 ARP를 이용하여 서버의 상태를 검사하는 2계층 헬스체크 기능이며 ARP 테이블과 ARP 테이블 갱신을 통하여 점검한다. 이 기능은 로드밸런서의 ARP 테이블에 서버의 MAC Address의 존재 여부와 ARP를 이용하여 ARP Request 메시지 전송과 ARP Reply 메시지 수신 여부에 따라서 서비스 서버의 서비스 가능 유무를 판단한다[9].

2.1.3 ICMP Health Check

ICMP 헬스체크는 ICMP를 이용하여 서버의 상태를 검사하는 3계층 헬스체크 기능으로 ICMP의 특성을 이용하여 서버의 IP Address 활성화를 확인하여 점검한다. 이 기능은 서버의 IP Address에 ICMP를 이용하여 ICMP Echo Request 메시지를 전송한 후 ICMP Echo Reply 메시지 수신 여부에 따라서 서비스 서버의 서비스 가능 유무를 판단한다[10].

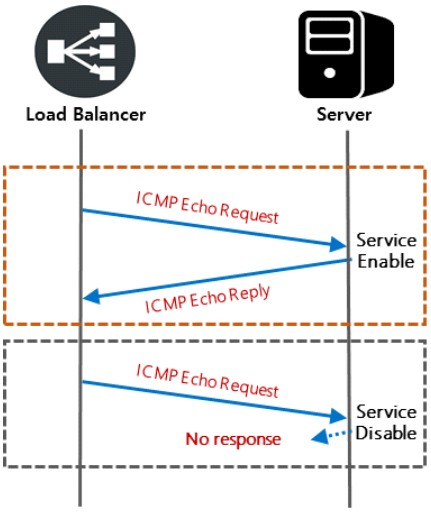


그림 2. ICMP 헬스체크
Fig. 2. ICMP Health Check

2.1.4 TCP Health Check

TCP 헬스체크는 TCP에서 제공되는 서비스 포트를 이용하여 서버의 상태를 검사하는 4계층 헬스체크 기능이며, 이미 지정된 포트를 사용하는 FTP, HTTP, Telnet 등과 개발자가 가용할 수 있는 영역에 있는 포트 번호를 이용하여 서비스 제공 가능 유무를 점검한다. 이 기능은 서비스를 위한 포트의 상태를 TCP의 초기화에서 사용되는 3 way handshake를 이용하여 서버의 서비스 포트로 TCP_SYN을 전송하고 그에 따른 SYN_ACK로 응답 여부에 따라서 서비스 서버의 서비스 가능 유무를 판단한다. 이후 서버의 소켓 낭비

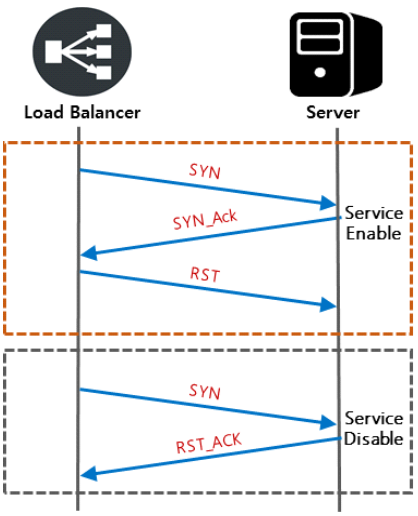


그림 3. TCP Halp-Open 헬스체크
Fig. 3. TCP Halp-Open Health Check

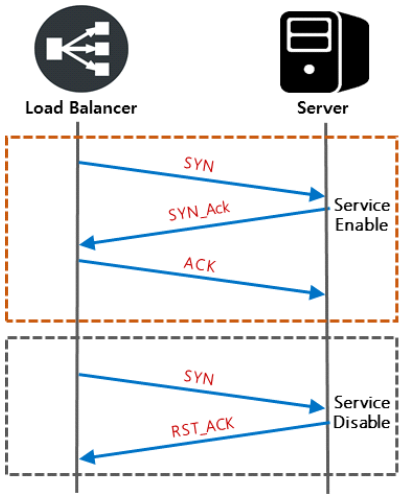


그림 4. TCP 헬스체크
Fig. 4. TCP Health Check

를 막기 위해 SYN_ACK 응답 수신 시 바로 RST 패킷을 전송하여 세션을 종료하는 Half-Open 방식과 ACK 패킷을 전송하는 TCP Open 방식으로 나뉜다[11].

2.1.5 Script Health Check

스크립트 헬스체크는 Script를 이용하여 Application Demon을 점검하는 7계층 헬스체크 기능이며, Script로 작성된 순서에 의하여 서버로 메시지를 전송하고 그에 따른 응답 메시지 유무에 따라서 서버의 서비스 가능 유무를 판단한다. 기본적으로는 서버와 TCP 세션을 맺고 Request 메시지 전송을 통해 응답 코드 확

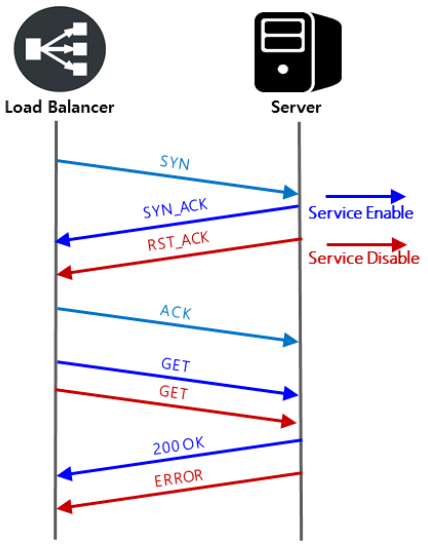


그림 5. 스크립트 헬스체크
Fig. 5. Script Health Check

인한다[12].

Ⅲ. 제안 기술

3.1 기존 헬스체크 기술의 문제점

로드밸런서에서 제공하는 헬스체크는 상대 장비의 상태를 설정된 주기로 전송하는 헬스체크 패킷을 사용하여 장애를 감지한다. 이를 바탕으로 로드밸런서는 서버의 상태를 식별하고 장애 발생 시 설정변화 없이 장애가 발생한 장비를 제외하고 실시간 서비스를 수행한다.

그림 6은 로드밸런서가 실시간 서비스 도중 헬스체크를 통해 서버3에 장애가 감지된 상황이다. 이러한 상황에서 로드밸런서는 서버3과 연결된 모든 세션을 삭제하고 서버3의 상태가 정상적으로 복구될 때까지 서버1과 서버2로 로드밸런싱 서비스를 제공한다. 로드밸런서는 헬스체크 패킷만을 사용하여 장애를 감지한다.

헬스체크는 설정된 주기로 전송하여 설정된 횟수 이상 연속적인 장애가 감지되어야 장애를 식별한다. 대부분 상용 장비의 기본값으로 설정될 경우 5초에 1번 패킷을 전송하여 연속된 3번의 장애가 감지되어야 장애 상태로 식별하기 때문에 장애 발생 시 10초~20초의 매우 긴 장애 시간이 발생하게 된다.

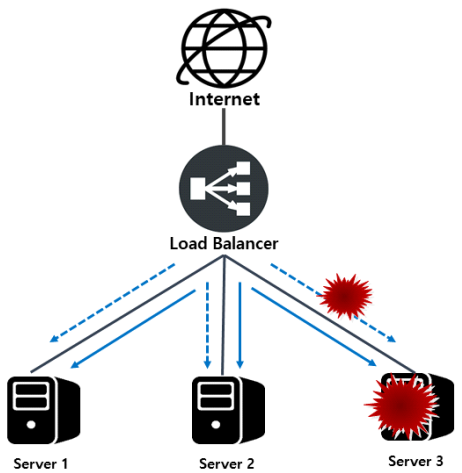


그림 6. 헬스체크 장애감지 상황
Fig. 6. Health Check failure detection situation

3.2 제안하는 향상된 헬스체크 기술

서버를 포함한 네트워크 장비들은 다양한 네트워크 프로토콜을 구동하고 있으므로 실제 데이터 송수신을 하지 않더라도 그 프로토콜들이 주고받는 수많은 패킷이 전송된다. 즉 서버와 연결된 로드밸런서의 인터페이스는 해당 서버로부터 나오거나 들어가는 패킷들을 처리하게 된다. 만약 로드밸런서의 인터페이스에서 패킷 수신량이 증가하지 않는다면, 해당 서버의 장애를 의심해 볼 수 있다는 것이 본 논문에서 제안하는 기법의 기본적인 아이디어이다. 따라서 인터페이스에서 수신하는 패킷 통계를 통하여 서버 장애를 빠르게 탐지하여 장애 시간을 줄일 수 있는 효과적인 방안을 제안한다. 그림 7은 인터페이스의 상태 정보를 조회한 결과이다.

로드밸런서의 인터페이스 상태 정보 조회를 통해 서비스하는 대상 서버와 연결된 인터페이스에서 수신한 패킷의 수와 수신한 용량을 알 수 있다. 본 논문의 기본 아이디어는 연결된 서버의 상태를 로드밸런서의 인터페이스 상태를 기반으로 판단하는 것이다. 즉 정상적으로 동작하는 서버일 경우에는 지속해서 패킷을 보내는 것이 일반적이므로 패킷 통계가 증가하는 인터페이스는 정상으로 판단해도 된다. 문제는 패킷을 보내지 않는 서버의 경우에는 장애로 인해 동작하지 않거나 정상이지만 패킷 전송이 없는 2가지 상황에 대해서 고려해야 하는 점이다. 이를 위해서 본 논문에서는 그림 8과 같은 장애 탐지 알고리즘을 제안한다.

헬스체크 상태는 서버가 통신할 수 있으며 패킷 수신이 증가하는 Enable, 서버가 통신은 가능하지만 패킷 수신이 없는 None Enable, 서버가 통신할 수 없는 Disable로 구성된다.

설정 파라미터는 기존의 헬스체크 방식에서 사용한

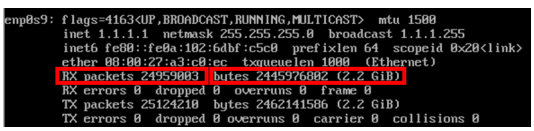


그림 7. 인터페이스 상태
Fig. 7. Interface status

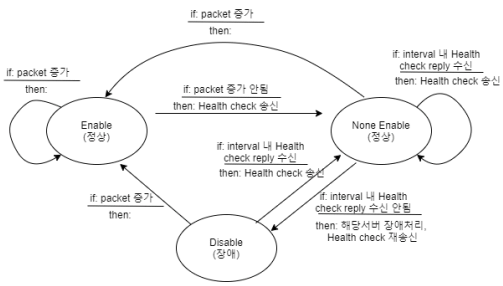


그림 8. 제안하는 로드밸런서의 헬스체크 프로세스 Finite State Machine
Fig. 8. Proposed load balancer Health Check process Finite State Machine

헬스체크 Layer를 지정하는 Type, 전송 주기를 결정할 Interval, 장애 판별을 위한 재시도 횟수를 결정할 Retry, 헬스체크 패킷을 전송할 인터페이스의 IP 주소인 Source IP, 헬스체크 패킷을 수신할 대상의 IP 주소인 Target IP에 헬스체크 대상 장비와 연결된 인터페이스의 수신 상태 체크를 위한 설정 파라미터로 Source Interface가 추가된다.

로드밸런서는 0.5초 간격으로 인터페이스 상태 정보 조회를 통해 수신한 패킷 양을 점검한다. 수신 패킷이 증가하는 경우에는 통신이 가능한 상태로 판단하고 헬스체크 패킷을 전송하지 않는다. 수신 패킷의 증가가 멈춘 경우에는 서버 장애 발생으로 인해 통신이 중지된 것일 수도 있으므로 즉시 health check 패킷을 송신하고 None Enable 상태로 천이한다. 이후에는 interval 마다 health check 패킷을 보내 상태를 검사한다. 만약 서버 장애가 아니라면 정해진 health check reply를 보낼 것이므로 이를 수신한 로드밸런서는 None Enable 상태를 유지한다. 그러나 정해진 interval 동안 health check reply가 수신되지 않는다면 장애로 간주하고 Disable상태로 처리한다. 수신 패킷이 증가하여 통신이 복구되었다고 판단 될 경우 헬스체크 패킷 전송을 중단하고 Enable 상태가 된다.

3.3 패킷 수신량 확인 프로세스

설정된 인터페이스의 현재 패킷 수신량을 점검하여 0.5초 전의 패킷 수신량과 비교한다. 패킷이 수신되고 있는 경우, 즉 패킷 수신량이 증가한 경우에는 서비스 대상 장비가 통신이 가능한 상태로 판단한다. 헬스체크의 상태가 통신이 불가능한 상태인 Disable이거나, 통신이 가능하지만 패킷 수신량이 없는 None Enable인 경우에는 헬스체크의 상태를 Enable 상태로 변경한다. Enable 상태인 경우 헬스체크 상태변화 없이 0.5초를 대기한 이후 패킷 전송 수치를 체크한다.

패킷 수신이 중단되면 해당 서버는 None Enable이나 Disable 상태라고 판단한다.

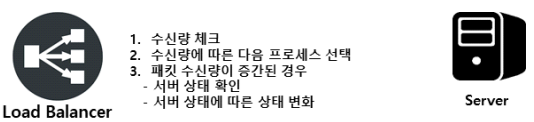


그림 9. 패킷 수신량 확인
Fig. 9. Packet reception amount check

3.4 패킷 수신 중단 상태 동작 프로세스

헬스체크 상태가 Enable인 경우 패킷 수신이 중단된 것으로 식별되면 Retry 수만큼의 헬스체크 패킷을 서비스 대상 장비에 전송한다.

헬스체크 패킷 전송 결과 응답 패킷이 있는 경우, 헬스체크 상태를 서비스 대상 장비가 통신은 가능하지만 패킷 수신이 없는 None Enable 상태로 변환한다.

헬스체크 패킷 전송 결과 응답 패킷이 없는 경우 헬스체크 상태를 서비스 대상 장비가 통신이 불가능한 상태인 Disable 상태로 변환한다.

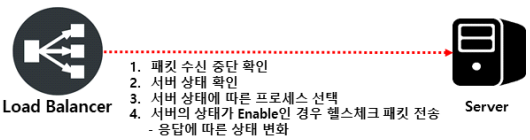


그림 10. 패킷 수신 중단 상태
Fig. 10. Packet reception interruption status

3.5 패킷수신 중단 상태 유지 동작 프로세스

헬스체크 상태가 Disable, None Enable인 경우 Enable 상태와 동일하게 Retry 수만큼의 헬스체크 패킷을 지속해서 전송하면 네트워크의 부하가 될 수 있으므로 기존 헬스체크 방식과 유사하게 동작한다.

기존 헬스체크 방식과 유사하게 동작하기 위해 변수를 활용하여 누적된 값이 Interval과 동일한 경우 변수를 초기화시킨 이후 서비스 대상 장비에 헬스체크 패킷을 전송한다.

헬스체크 패킷 전송 결과 응답이 있고 헬스체크 상태가 None Enable인 경우 현재 상태를 유지하고, Disable인 경우 헬스체크 상태를 None Enable로 변경한다.

헬스체크 패킷 전송 결과 응답이 없고 헬스체크 상태가 Disable인 경우 현재 상태를 유지하고, None Enable인 경우 Fail Count를 1 증가시킨다. Fail Count가 Retry와 같아지는 경우 Fail Count를 초기화시키며 헬스체크 상태를 Disable로 변경한다.

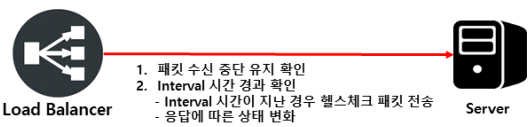


그림 11. 패킷 수신 중단 상태 유지
Fig. 11. Keep packet reception stopped

Ⅳ. 실험 및 성능 평가

4장에서는 제안한 헬스체크 기법을 적용하여 실험을 진행하고, 기존의 헬스체크 방식과 성능을 비교한다.

4.1 실험 환경

이 실험은 헬스체크를 기존 로드밸런서에 설정된 간격으로 서버에 헬스체크 패킷을 전송하여 설정된 횟수 이상의 시간 동안 장애가 유지되면 장애 상황으로 판단하는 것이 아니라, 로드밸런서 인터페이스의 패킷 수신 상태를 점검하여 패킷 수신량이 없어질 때 즉시 다량의 헬스체크 패킷을 전송하여 장애 상황인지를 판단할 때 장애감지 시간이 절감되는 것을 확인하기 위함이다.

그림 12는 서버에 요청을 보내는 Client와 장애 상황을 발생할 수 있는 서버, 서버의 헬스체크 기능을 수행하는 로드밸런서로 구성된 시험환경이다.

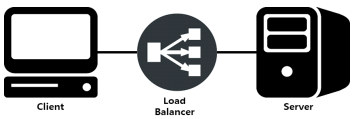


그림 12. 성능평가 구성도
Fig. 12. Performance evaluation configuration diagram

실험에 환경에 사용된 도구는 표 1과 같다.

표 1. 실험 도구
Table 1. Experiment tool

구분	설명	비고
Client	CPU: Intel Core i7-6800K 3.40 GHz NIC: 10/100/1000 Mbps * 4 HDD: 120 GB RAM: 16 GB OS: CentOS 7 Linux release 7.3.1611 64 bit Kernel : 3.10.0-514.10.2.el7.x86_64	패킷 발생기
Load Balancer		
Server		장애 발생기

4.2 실험 항목

성능평가 방법은 기존에 로드밸런서에서 제공하는 ICMP 헬스체크와 제안하는 헬스체크를 두 가지 항목으로 첫 번째는 전송량이 있는 경우 장애감지에 걸리는 시간과 두 번째로 전송량이 없는 경우 장애감지에 걸리는 시간으로 구별하여 비교하고 동작 방식 변경에 따른 오버헤드를 확인한다.

4.3 실험계획

성능시험에 장애감지에 걸리는 시간 확인을 위해 NTP(Network Time Protocol)를 사용하여 클라이언트와 로드밸런서, 서버의 시간을 동기화시킨다.

로드밸런서에서는 헬스체크를 동작시키며 서버의 상태변화가 감지되면 Logger 명령을 활용하여 Syslog에 기록한다. 헬스체크 설정은 일반적인 상용 로드밸런서에서 제공하는 기본 설정인 5초에 한번 헬스체크 패킷을 전송하여 3회 연속 장애가 감지되면 장애 복구하도록 Interval 5, Retry 3으로 설정하여 시험한다. 추가적으로 설정값에 따른 결과 차이 확인을 위해 빠른 장애감지를 위한 설정으로 기본 설정의 절반 값인 Interval 3, Retry 2와 네트워크와 서버의 노후화로 인해 헬스체크 패킷 처리량을 줄인 설정으로 기본 설정의 두 배 값인 Interval 10, Retry 6으로 설정하여 시험한다.

서버에서는 물리적 링크 혹은 장비의 상태 이상으로 발생하는 장애 상황을 재현하기 위해 인터페이스를 무작위 간격으로 Up/Down시키며 상태변화를 시키는 상황과, 인재(관리자의 관리실수) 발생으로 인해 장비의 설정변화가 생겨 연결 장애가 발생하는 상황을 재현시키기 위해 무작위 간격으로 인터페이스의 IP 주소를 변경한다.

로드밸런서는 서버의 상태변화가 발생할 때 Logger 명령을 활용하여 Syslog에 기록한다.

클라이언트에서는 전송량이 없는 상황 재현을 위해서는 아무런 동작하지 않으며 전송량이 있는 상황을 재현하여 장애감지에 걸리는 시간을 시험하기 위해서 서버로 Ping -f 옵션을 통하여 다량의 패킷을 전송한다.

4.4 실험 결과

4.4.1 물리적 장비 고장으로 인한 상황

표 2, 그림 13은 실험 결과로 헬스체크 설정별 평균 수치이다. 사용자가 장애 사항을 체감할 수 있는 전송량이 존재하는 상황에서 장애가 발생하는 경우 기존 헬스체크 방식과 비교하면 설정에 따라

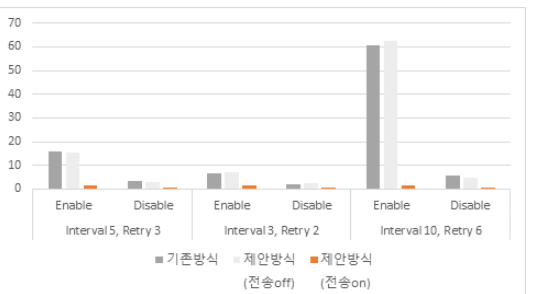


그림 13. 인터페이스 UP/DOWN 실험결과
Fig. 13. Interface UP/DOWN test result

표 2. 인터페이스 UP/DOWN 실험결과
Table 2. Interface UP/DOWN test result

인터페이스 UP/DOWN (단위 초)		기존 방식	제안 방식 (전송 off)	개선율(%)	제안 방식 (전송 on)	개선율 (%)
Interval 5 Retry3	장애 발생 감지	3.34	3	11%	0.22	93%
	장애 해소 감지	15.72	15.5	1%	1.72	89%
Interval 3 Retry2	장애 발생 감지	2.12	2.32	-9%	0.44	79%
	장애 해소 감지	6.50	7.08	-8%	1.74	73%
Interval 10 Retry6	장애 발생 감지	5.72	4.78	20%	0.78	86%
	장애 해소 감지	60.58	62.42	-3%	1.68	97%

79%~93% 향상된 장애감지 시간을 확인할 수 있었고 복구되는 경우 기존 헬스체크 방식과 비교하면 설정에 따라 73%~97% 향상된 감지 시간을 확인할 수 있었다. 사용자가 장애 상황을 체감할 수 없는 전송량이 없는 상황에서 장애가 발생하거나 복구되는 경우에는 기존 헬스체크 방식과 유사한 성능 수치를 확인 할 수 있었다.

4.4.2 인재(관리자의 관리실수)로 인한 상황

표 3, 그림 14는 인터페이스 IP 주소 변환 실험 결과로 헬스체크 설정별 평균 수치다. 사용자가 장애 사항을 체감할 수 있는 전송량이 존재하는 상황에서 장애가 발생하는 경우 기존 헬스체크 방식과 비교하면 설정에 따라 73%~92% 향상된 장애감지 시간을 확인할 수 있었고 복구되는 경우 기존 헬스체크 방식과 비

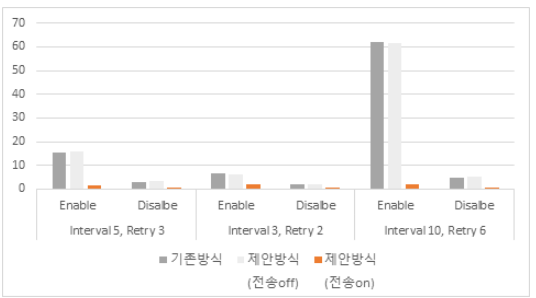


그림 14. 인터페이스 IP 주소 변환 실험결과
Fig. 14. Interface IP address conversion experiment result

표 3. 인터페이스 IP주소 변환 실험결과
Table 3. Interface IP address conversion experiment result

인터페이스 IP 주소 변환 (단위 초)		기존 방식	제안 방식 (전송 off)	개선율 (%)	제안 방식 (전송 on)	개선율 (%)
Interval 5 Retry3	장애 발생 감지	2.86	3.48	-18%	0.38	87%
	장애 해소 감지	15.38	15.86	-3%	1.78	88%
Interval 3 Retry2	장애 발생 감지	2.14	2.08	3%	0.58	73%
	장애 해소 감지	6.68	6.34	5%	1.84	72%
Interval 10 Retry6	장애 발생 감지	4.72	5.46	-14%	0.40	92%
	장애 해소 감지	62.12	61.74	1%	1.80	97%

교하면 설정에 따라 72%~97% 향상된 감지 시간을 확인할 수 있었다. 사용자가 장애 상황을 체감할 수 없는 전송량이 없는 상황에서 장애가 발생하거나 복구되는 경우에는 기존 헬스체크 방식과 유사한 성능 수치를 확인할 수 있었다.

4.4.3 성능 평가 및 고찰

표 4는 앞서 진행된 성능 시험의 장애 유형별 평균 수치다. 사용자가 장애 사항을 체감할 수 있는 전송량이 존재하는 상황에서 서버의 장애가 발생하는 경우 기존 헬스체크 방식과 비교하면 장애가 발생한 경우 기존 방식 3.73초 제안하는 방식 0.48초로 87% 향상된 장애감지 시간을 확인할 수 있었고, 복구되는 경우에는 기존 헬스체크 방식 27.6초 제안하는 방식 1.71

표 4. 평균 감지 시간(단위 초)
Table 4. Average detection time (unit second)

		장애 발생	인재 발생
기존 방식	장애 발생 감지	3.73	3.24
	장애 해소 감지	27.60	28.06
제안 방식 (전송 off)	장애 발생 감지	3.37	3,67
	장애 해소 감지	28.33	27.98
제안 방식 (전송 on)	장애 발생 감지	0.48	0.45
	장애 해소 감지	1.71	1.81

초로 94% 향상된 장애감지 시간을 확인할 수 있었다. 또한, 관리자의 실수로 인재가 발생한 경우는 기존 헬스체크 방식 3.24초 제안하는 방식 0.45초로 86% 향상된 장애감지 시간을 확인할 수 있었고, 복구되는 경우에는 기존 헬스체크 방식 28.06초 제안하는 방식 1.81초로 94% 향상된 장애감지 시간을 확인할 수 있었다. 사용자가 장애 상황을 체감할 수 없는 상황에서 장애가 발생하거나 복구되는 경우에는 기존 헬스체크 방식과 유사한 성능 수치를 확인할 수 있었다.

1시간 이상 헬스체크를 동작시킨 이후 자원 사용량을 비교하여 오버헤드를 확인하였다.

그림 15는 기존 방식으로 동작한 헬스체크에서 1시간 동안 사용된 자원으로 1460KB 메모리와 0%의 CPU를 사용하였다.

그림 16은 제안 방식으로 동작한 헬스체크에서 1시간 동안 사용된 자원으로 1540KB 메모리와 0%의 CPU를 사용하였다.

제안하는 헬스체크를 사용하기 위해는 기존 방식으로 할 때보다 더 많은 모니터링을 해야 하는데, 그에 대한 오버헤드는 약 80KB로 무시할 수 있을 만큼 작았다.

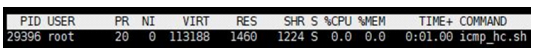


그림 15. 기존방식 자원 활용량
Fig. 15. Existing method resource utilization

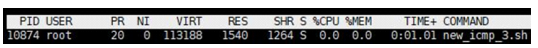


그림 16. 제안방식 자원 활용량
Fig. 16. Proposed method resource utilization

V. 결 론

본 논문에서는 로드밸런서에서 제공하는 기존 헬스체크 방식에서 가지고 있는 서비스 대상 장비의 상태와 상관없이 일정한 주기로 헬스체크 패킷을 전송하여 장애를 감지하는 문제점을 확인하고, 인터페이스에서 수신하는 패킷 양식별을 통하여 장애 상황 발생 시 발생하는 장애 시간을 줄일 수 있는 최적화 방안을 제안하였다.

인터페이스에서 수신하는 패킷이 증가하는 경우에는 헬스체크 패킷을 전송하지 않고, 수신 패킷의 증가가 멈추면 즉시 다량의 패킷을 전송함으로써 장애감지를 감지하며, 이후 통신이 중지된 상태가 유지될 경우에는 기존 헬스체크 방식과 유사하게 동작할 경우

표 5. 헬스체크 설정 별 개선율
Table 5. Improvement rate by health check setting

		장애 발생	인재 발생
Interval 5 Retry3	장애 발생 감지	93%	87%
	장애 해소 감지	89%	88%
Interval 3 Retry2	장애 발생 감지	79%	73%
	장애 해소 감지	73%	72%
Interval 10 Retry6	장애 발생 감지	86%	92%
	장애 해소 감지	97%	97%

헬스체크의 장애감지 효율성이 증가하는지에 대해 구현, 시험해 보았다.

제안하는 헬스체크 방식을 사용하였을 때 사용자가 장애 사항을 체감할 수 있는 전송량이 존재하는 상황에서 장애가 발생하는 경우 표 5와 같이 Interval, Retray 설정에 따라 기존 헬스체크 방식보다 72%~97% 향상된 장애감지 시간을 확인할 수 있었다.

References

[1] H.-Y. Bahk, H.-Y. Choi, and S.-G. Min, “A study on server load balancer’s load reduce by using the direct server return,” *KIPS Spring Conf.*, vol. 13, no. 2, pp. 1395-1398, 2006.

[2] 이토 나오야, 카츠미 유키, 다나카 신지, 히로세 마사아키, 야스이 마사노부, 요코가와 카즈야, *24시간 365일 서버/인프라를 지탱하는 기술*, 제이펍, 2009.

[3] Kissmetrics, *How loading time affects your bottom line* (2011), Dec., 21 2020, https://blog.kissmetrics.com/loading-time/

[4] Akamai, Akamai Online Retail Performance Report: *Milliseconds Are Critical*(2017), Dec., 21, 2020, https://goo.gl/72fpyT

[5] PLOLINK, *PIOLINK Application Switch-K 1500/2400/2800/4200/4400 사용 설명서 2012*, pp. 617-636, 2008.

[6] 바바 토시아키, *웹 엔지니어가 알아야 할 인프라의 기본*, 한빛미디어, 2015.

[7] 미야타 히로시, *네트워크 이해 및 설계 가이드*, 제이펍, 2014.

[8] redware, *AlteonOS-32.2.1-WBM_Application_Guide*, 2019.

[9] Y. Cho, "Enhanced route management using health check for next hop in multi-path environment," M.S. Thesis, Sogang Univ., 2009.

[10] 타테오카 마모루, 이마이 토모아키, 나가후치 코코, 마세 테츠야, 미우라 사토루, *탄력적 개발로 이끄는 AWS 실천 기술*, 제이펍, 2016.

[11] PumpkinNetworks, *AEN Series Configurations User's Guide - Part 1*, 2019.

[12] kakao, *kakao의 L3DSR 구성 사례*(2014), Dec., 21, 2020, https://tech.kakao.com/2014/05/28/l3dsr/

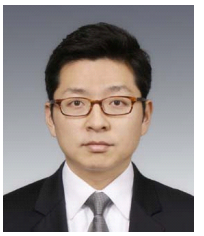
전 수 철 (Soo-Cheol Jeon)

2016년 8월 : 한양사이버대학교, 컴퓨터공학과 졸업

2020년 8월 : 숭실대학교 IT융합학과 석사

<관심분야> 컴퓨터공학, 네트워크 보안

주 양 익 (Yang-Ick Joo)



1998년 2월 : 고려대학교 전자공학과 졸업

2000년 2월 : 고려대학교 전자공학과 석사

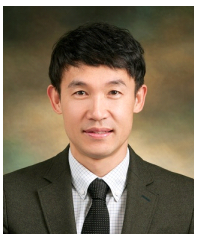
2004년 8월 : 고려대학교 전자공학과 박사

2004년~2012년: 삼성전자

2012년~현재: 한국해양대학교 교수

<관심분야>유/무선네트워크,WPAN/WBAN, VANET

박 민 호 (Minho Park)



2000년 2월 : 고려대학교 전자공학과 졸업

2002년 2월 : 고려대학교 전자공학과 석사

2010년 2월 : 서울대학교 전기컴퓨터공학부 박사

2002년~2004년 : 삼성전자

2013년~현재 : 숭실대학교 부교수

<관심분야> 유/무선네트워크 및 보안