확장가능한 클라이언트를 이용한 웹서버 성능평가 기법

이승규*, 박영록*, 이건화*, 배철수**

Evaluating the capacity of a Web Server using Scalable Client

Seung-Kyu Lee*, Yung-Rok Park*, Geon-Wha Lee*, Cheol-Su Bae*,

요 약

서버의 성능 평가를 위해 확장 가능한 클라이언트를 생성한 후, UDP 프로토콜을 이용하여 서버로 패킷을 전송한다. 전송된 패킷 중에 네트워크 손실을 제외하고 실제적으로 서버에 전달된 패킷에 대한 응답을 TCP 프로토콜을 이용하여 클라이언트로 보내는 구조를 제안한다. 또한, POSIX Thread를 이용해 생성한 확장 가능한 클라이언트는 서버의 용량을 뛰어넘는 클라이언트의 수를 생성할 수 있다. 그러므로 기존의 방법에 비교하여 실제적으로 서버에 적용되는 트래픽를 이용하여 서버를 대한 성능평가를 한다.

Abstract

As the fast growth of using Internet, the requests of clients having different types and pressing loads on the server have been increased in World Wide Web. Thus the interesting issue is how to measure the real capacity of a Web Server. There have been much recent studies about measuring the capacity of web server. But the cause of Server response time delay is not just server itself but also network packet loss. To measure the practical capacity of web server, we generate scalable clients using Posix Thread, transport packets which were generated by scalable clients to the server using UDP and receive the packets which were the remain packet from network packet loss using TCP. In this paper, we propose a method to measure the practical capacity of a web server using the Scalable Clients based on Posix Thread and the transport on Application level.

키워드 TCP/IP, Posix Thread, Web server, RTT

Ⅰ. 서론

인터넷 사용의 고속 증가로 인해 서버에 대한 폭발적이며 다양한 형태의 클라이언트의 요구들이 발생하고 있다. 이러한 점은 네트워크와 서버에 부하를 가중시켰으며, 웹이 기반으로 하는 프로토콜에도 부담을 주었다. 이러한 웹 환경의 성능을 개선하기 위해 웹 캐싱 방법, http 프로토콜의 성능 향상 법, 웹 서버의 성능 향상 법, 서버 운영체제에 대한 연구, 프록시 서버에 대한 연구 등 많은

* 관동대학교 전자통신공학 박사과정
** 교신저자 : 관동대학교 의료공학과 교수
접수일자 : 2013년 10 월 06 일, 수정일자 : 2013년 11 월 06 일, 심사완료일자 : 2013년 12 월 06 일

연구들이 있어왔다.

특히, 서버의 성능에 주요 관심을 가지게 되면서 서버의 성능을 측정하기 위한 다양한 형태의 방법들이 제안되었다. 예를 들어, 데이터의 크기와 타입, 전송 경로 등을 고려한 방법, 웹 서버와 프록시 서버를 이용한 방법 등이 연구되어왔다. 하지만 이러한 방법들은 실제적이고 종합적인 웹 트래픽을 생성하는 데에 어려움을 나타내고 있다. 또한, 제한된 클라이언트 머신으로 서버의 용량을 초과하는 트래픽을 생성하는 데에도 어려움을 가지고 있다.

서버 응답 시간의 딜레이가 서버만의 문제라고 볼 수는 없기 때문에 네트워크상에서 데이터 전송시에 데이터에 대한 경로 추적을 통한 네트워크 딜레이와 서버 측 딜레이에 대한 연구도 있다. 서버와 클라이언트 사이에는 통신을 위한 프로토콜이 존재하며, 서버와 클라이언트는 프로토콜을 기반으로 네트워크를 통해 데이터를 교환하기 때문에 그 과정 중에는 패킷의 손실로 인해 발생하는 딜레이 현상은 서버의 문제가 아니라 네트워크 자체의 문제로 생각 할 수 있다. 물론 이러한 점을 개선하기위해 서버와 클라이언트간의 패킷 전송 실패 시 효율적인 재전송 방법에 대한 연구들도 있다. 즉, 서버의 성능을 측정하기 위해서는 네트워크에서 발생하는 손실을 제외한 실제적으로 서버에 전달된 패킷만을 대상으로 서버 성능을 측정해야한다.

본 논문에서는, 서버의 성능 평가를 위해 확장가능한 클라이언트를 생성한 후, UDP 프로토콜을 이용하여 서버로 패킷을 전송한다. 전송된 패킷 중에 네트워크 손실을 제외하고 실제적으로 서버에 전달된 패킷에 대한 응답을 TCP 프로토콜을 이용하여 클라이언트로 보내는 구조를 제안한다. 또한, 포식스 스레드(Posix Thread)를 이용해 생성한 확장 가능한 클라이언트는 서버의 용량을 뛰어넘는 클라이언트의 수를 생성할 수 있다. 우리의 연구는 기존의 방법에 비교하여 실제적으로 서버에 적용되는 트래픽를 이용하여 서버를 대한 성능평가를 한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 관련연구에 대하여 기술 할 것이며, 3장에서는 제안기법인 포식스 스레드(Posix Thread)를 이용한 확장 가능한 클라이언트생성 기법, TCP와 UDP를 이용한 어플리케이션 레벨에서의 서버 성능 측정 기법을 제안한다. 그리고 4장에서는 제안된 기법을 구현하여 실제 서버에 대한 성능 측정 결과를 살펴본다. 마지막으로 5장에서는 결론과 제안기법의 제약사항을 이야기함으로써 끝을 맺는다.

II. 관련연구

대용량 서버의 성능 평가를 위해서는 우선적으로 서버와 클라이언트의 구조, 현재의 유동적인 웹 환경, 그리고 TCP와 UDP 전송방식에 대해 살펴볼 필요가 있다. 또한, 서버 성능 평가를 위한 기존의 연구들을 살펴본다.

2.1 서버와 클라이언트 모델

클라이언트와 서버는 두 개의 컴퓨터 프로그램 사이에 이루어지는 역할 관계를 나타내는 것으로, 클라이언트는 다른 프로그램에게 서비스를 요청하는 프로그램이며 서버는 그 요청에 대해 응답을 해주는 프로그램이다. 클라이언트와 서버 개념은 단일 컴퓨터 내에서도 적용될 수 있지만, 네트워크 환경에서 더 큰 의미를 가진다. 네트워크상에서 클라이언트와 서버 모델은 여러 다른 지역에 걸쳐 분산되어 있는 프로그램들을 연결시켜주는 편리한 수단을 제공한다.

오늘날 만들어지고 있는 대부분의 업무용 프로그램들은 클라이언트와 서버모델을 적용하고 있으

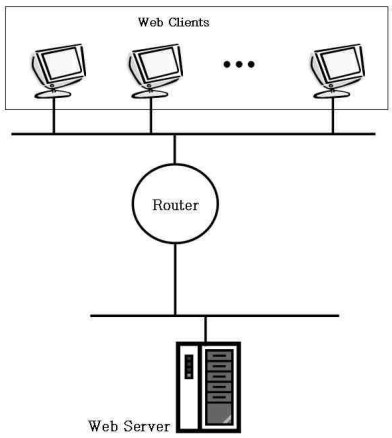


그림 1. 서버와 클라이언트 구조

며, 인터넷의 주요 프로그램인 TCP/IP 또한 마찬가지다. 인터넷의 경우를 예를 들면 웹 브라우저는 인터넷상의 어딘가에 위치한 웹서버에게 웹 페이지나 파일의 전송을 요구하는 클라이언트 프로그램이다.

일반적인 클라이언트와 서버 모델에서는 서버프로그램이 먼저 활성화된 상태에서 클라이언트의 요구사항을 기다리는데, 대체로 다수의 클라이언트 프로그램이 하나의 서버 프로그램을 공유한다.

2.2 Web 환경의 유동성

월드와이드 웹에서 수많은 클라이언트들에 의해 생성된 요청(request)들은 특정 서버를 접속함에 있어서 넓게 퍼진 분포 값을 나타낸다. 즉, 일정한 평균과 오차를 가지지만 컴퓨터 사용자의 접속 패턴이나, 새롭게 웹에 게시되는 데이터의 스케줄, 클라이언트 머신의 성능 등이 클라이언트 요청(request)의 서버 접속에 영향을 미쳐 각각의 접속 시간은 서로 상이한 경우가 발생한다. 이러한 이유로 서버의 성능을 측정하기 위해 웹 환경과 유사한 실제적 요청(request)을 인위적으로 생성하는 것은 매우 어려운 일이다.

적은 수의 클라이언트 머신에 의해 생성된 간단한 형태의 요청(request)들은 접속성립 시 일정한 분포 값을 가질 수밖에 없다. 결론적으로 서버 성능 평가를 위해 인위적으로 생성된 간단한 형태의 요청(request)들은 서버에 적은 영향을 미친다.

2.3 TCP와 UDP 전송방식

HTTP 프로토콜에서 클라이언트와 서버가 TCP 연결을 하기위해서는 클라이언트가 연결에 대한 요청(request)를 보내고 서버가 응답(response)를 하는 형태를 가진다. 즉, 하나의 웹 페이지를 보기 위해서 브라우저는 HTTP 프로토콜을 이용해 접속을 초기화하고 텍스트나 이미지 등과 같은 다양한 데이터를 전송받아야한다.

그림 2는 TCP 연결 성립 과정을 보여준다. 접속 시작을 위해 웹 서버는 일반적으로 80포트로 들어오는 요청(request)을 기다린다. 클라이언트로부터 오는 TCP SYN Packet에 대해 서버가 SYN-ACK Packet으로 응답을 하면 새로운(완전한 연결이 성립되지 않은) 소켓을 생성해서 Listen

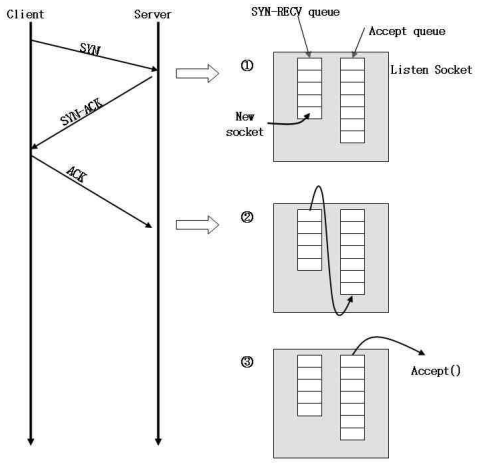


그림 2. TCP 연결 성립 과정

Socket에 있는 SYN-RECV queue로 보낸다. 그 후에 클라이언트가 서버 측의 SYN-ACK Packet에 대한 응답으로 ACK packet을 보내면, 서버는 SYN-RECV queue에 있던 소켓을 Accept queue로 이동시킨다. Accept queue에 있는 Socket들은 서버의 accept() call을 기다리며, 일단 Accept가 이루어지면 웹 서버는 클라이언트로부터의 요청(request)을 받아서 적절한 응답을 한다.

TCP의 이러한 3-way 핸드셰이킹 과정은 서버와 클라이언트간의 신뢰성 있는 접속을 보장한다. 즉, TCP를 통해 데이터를 전송 할 때는 모든 패킷이 수신 될 때 까지 접속이 유지되며 데이터의 순서도 정확히 유지된다. 하지만 네트워크 트래픽이 증가하면 전송지연 현상이 발생하여 응답시간 딜레이가 생기는 단점이 있다.

반면 UDP는 데이터를 전송하는데 있어 신뢰성을 보장하지 못한다. 즉, 그림 3처럼 UDP는 TCP와 같은 핸드셰이크 과정이 없기 때문에 데이터가 정확히 도착 할 경우도 있고 그렇지 못 할 경우도 있다. 혹은 데이터의 순서가 바뀌어서 전달 될 수도 있다. UDP의 경우 데이터를 전송한 후, 데이터가 정확히 전송되었는지 확인을 하지 않기 때문에 전송에 대한 신뢰도가 떨어진다. 하지만 전송속도는 TCP에 비해 성능이 우수하다. UDP는 패킷이 정확히 도착을 했는지 확인을 하지 않고 계속 보내기 때문에 네트워크에서 패킷 손실이 발생하더

라도 다음 패킷은 대기를 하지 않고 계속 전송된다. 즉, 전송 지연 시간은 TCP처럼 발생하지 않지만 신뢰성을 보장할 수 없다는 단점이 있다.

그림 3에서와 같이 UDP의 또 다른 특징은 연결 관리를 하지 않는다는 것이다. TCP는 연결 지

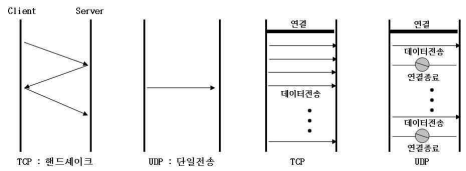


그림 3. TCP와 UDP의 데이터 전송

향형 프로토콜이고 UDP는 비 연결 지향형 프로토콜이다. 즉, TCP는 데이터 전송이 모두 완료 될 때까지 연결 상태를 유지하지만 UDP는 데이터를 전송 할 때마다 연결과 끊음을 반복한다.

2.4 서버의 성능평가 기법

웹 서버의 성능을 측정하기 위한 다양한 연구들이 있었으며 몇 몇 연구들을 살펴보도록 하겠다. 실제 트래픽을 발생시켜 웹 서버의 성능을 평가한 몇 몇 연구가 있었지만, 이러한 방법은 실험환경에 대한 어려움을 가지고 있다. 또한 실제 웹 트래픽에서 나타나는 요소들에 기반을 둔 종합적인 트래픽을 발생시켜 서버의 성능을 측정한 방법들도 있었지만, 이러한 방법들은 제한된 수의 클라이언트로 서버에 부하를 주는 트래픽을 생성하는 단점을 가지고 있다. 이러한 방법들은 현실과는 다른 환경에서 서버의 성능을 측정하려 했기 때문에 실제 서버의 성능과는 편차를 보인다. 이러한 단점을 극복하기위해 제한된 수의 클라이언트 머신으로 서버의 용량을 초과하는 트래픽을 생성하기 위하 연구도 있었다.

2.5 스레드에 대한 고찰

대중적인 PC에서의 다중프로세서의 사용이 급증하고 있으나 단일 처리기 시스템에서 운영되도록 개발된 응용 프로그램의 경우 다중처리기 시스템에서 운영하더라도 하나의 처리기 만이 할당되어 수행되고, 나머지 처리기는 유휴 상태로 있게 된다. 이와 같은 현상이 일어나는 주요 원인으로는 운영체제의 스케줄링 단위의 문제점과 병렬 프로그래밍으로 구성되어 있지 않은 점이다. 운영체제의 스케줄링 단위의 경우 전통적인 UNIX는 프로세스 단위로 설계되었으나, 현대의 운영체제는 마이크로 커널을 도입하여 스레드 단위의 스케줄링으로 처리하고 있다. 병렬 프로그래밍에 대한 연구는 최근 객체지향 프로그래밍, 원도우 프로그래밍, 네트워크 프로그래밍, 화상처리 등 비교적 큰 처리 단위의 응용분야가 확산되면서 그에 대한 해결책으로 스레드를 사용한 프로그램의 병렬화가 실용적이고 현실적인 대안으로 제시되고 있다.

2.5.1 프로세스

“운영체제 내에 PCB(Process Control Block)가 할당됨으로써 의미가 부여되는 개체”로써 운영체제는 모든 자원관리(할당 및 회수) 및 보안 관리를 PCB에 근간을 두고 수행한다. 특정 프로세스가 차지하는 메모리는 크게 실행 문, 데이터, 힙 그리고 스택 부분으로 구성된다. 여기서 실행 문 영역은 기계 명령어들의 집합으로 텍스트(text)라 부르고, 프로세스가 운영되는 한 내용이나 크기가 변경되지 않는다. 데이터 영역은 프로그램에서 전역 적 또는 고정적(static)으로 정의된 변수들이 차지하는 곳으로 변수들의 값을 프로세스가 운영되는 동안 영구적으로 유지한다. 힙 영역은 프로세스가 운영되면서 동적으로 할당받은 메모리 영역으로 다시 운영체제에 반납하지 않는 한 데이터 영역과 동일한 성격을 가진다. 마지막으로 스택 영역은 프로세스가 활동하면서 각 함수들을 호출할 때 문맥(제어정보, 레지스터 값 등), 매개변수, 지역 변수들의 저장을 위해 동적으로 할당되어 스택포인터(SP), 프레임포인터(FP)에 의하여 관리된다.

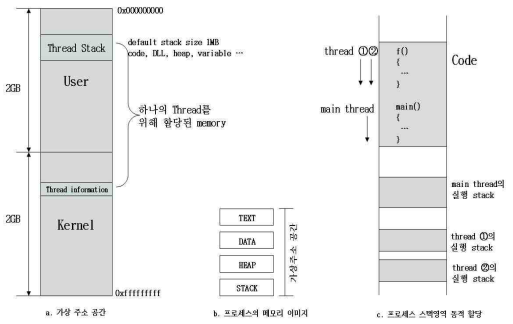


그림 4. 가상주소공간의 프로세스 스택영역 동적 할당

그림 4와 같이 모든 메모리 영역은 가상주소공간에 위치하고, 프로세스가 실행되면서 각 스택 영역을 할당 받는다. 그리고 주소변환 테이블에 의해 물리메모리와 연결된다. 문맥은 프로세스의 현재 진행 상태를 나타내는 일종의 스냅 샷으로서 진행 위치, 명령어 실행결과, 스택포인터, 기타 범용 레지스터 값 등의 집합이며, 문맥 교환(Context Switch)이 일어나게 되면 이와 같은 정보가 PCB에 저장된다.

2.5.2 스레드

스레드는 텍스트와 데이터 영역은 공유하지만 스택 부분은 별도로 할당되어 독립적으로 수행되는 단위를 일컫는다. 스레드를 사용하는 주된 이유 중의 하나는 프로세스 간 통신(IPC, Inter Process Communication)으로 파이프, 공유메모리, 메시지 큐, 세마포 등 운영체제의 중계에 의지하므로 복잡하고 효율이 떨어진다. 그러나 스레드 간에는 전역변수자체가 모두 공유되므로 하나의 프로세스 공간 내에서 정보를 쉽게 교환할 수 있다. 스택 영역은 각 스레드의 고유 데이터를 저장하는 장소로 분리되어 있고, 여기에는 스레드가 생성된 이후에 사용되는 모든 지역변수 및 관련 문맥이 저장된다. 이렇게 각 함수에 정의된 지역변수들은 독립된 장소에 위치하므로 스레드간의 혼란을 막고 유일성을 유지할 수 있다.

그림 5는 스레드를 단일 스레드와 다중 스레드로 분류한 것이다. 스택을 여러 개 관리하는 경우를 다중 스레드, 단지 하나의 스레드만 관리하는 것을 단일 스레드라고 한다.

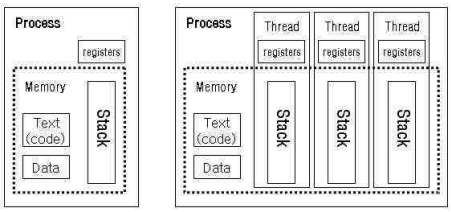


그림 5. 단일 스레드와 다중 스레드

2.4.3 스레드 적용 모델

다중 스레드를 적용하는 프로그램밍 모델은 그림 8과 같이 크게 3가지가 있다.

a. Boss/Worker Model(Worker/Queue Model)

작업 큐를 boss가 관리하면서, 유휴 스레드에게 전체 일거리를 할당하는 형태로 통신 서비스 처리 등에 사용된다.

b. Work Crew Model

하나의 커다란 일거리를 여러 스레드가 분담하여(분할·정복) 처리하는 형태이다.

c. Pipelining Model

한 스레드의 출력을 다른 스레드가 입력으로 받아 처리하는 과정의 연결 형태로서 본 논문에서 확장 가능한 클라이언트 구현에 적용할 모델이다.

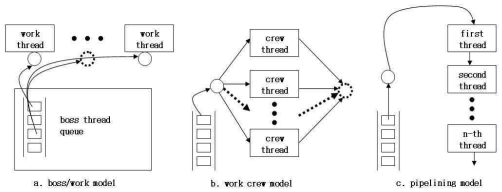


그림 6. 스레드 적용 모델

Ⅲ. 제안 기법

3.1 MultiThread 방식을 이용한 확장 가능한 클라이언트 생성 방법 제안

인터넷 사용자가 폭발적으로 증가함에 따라 웹 서비스가 중단되거나 원활하게 통신이 이루어지지 않으면 대부분의 서버 관리자는 네트워크의 문제로 돌리고 네트워크 관리자는 서버에 문제로 돌리는 경우가 많이 있다.

이러한 경우 앞에서 제기한 여러 가지 제약사항이 있는 이유로 실제 서비스 가능한 클라이언트 수를 계산하기란 매우 어렵다. 대부분의 실험들은 WAN상에서 테스트되기보다는 LAN에서 테스트되어 졌는데, 이때 현실세계와 동일한 실험환경을 만들 수 없으므로, 효율적이며 확장 가능한 클라이언트를 구현하기위해서는 각 클라이언트들은 MultiThread방식을 이용하여 서버에 접속하게 된다.

확장 가능한 클라이언트에는 여러 프로세스들이 실행되게 된다. 그리고 한 프로세스안에 여러개의 스레드가 존재하므로 하나의 프로세스는 클라이언트로 가정하고 프로세스안의 여러 스레드는 서버

로의 빈번한 접속으로 가정하게 된다.

스레드 인터페이스는 시스템 및 공급자에 따라 고유하게 구현될 수 있으나, 호환성을 위하여 POSIX(Portable Operating System Interface)의 표준안에 따라 pthread interface를 사용하여 구현하였다.

한 프로세스안에는 CET(Connection Est ablish-ment Thread)와 CHT(Connection Handling Thread)로 나누게 된다. 첫 번째 CET는 서버에 요청(request)을 생성하고 이후 지속적인 연결이 확립되면 두 번째 CHT에게 socket의 descriptor를 넘겨 응답(response)을 받도록 한다. 그림 7은 확장 가능한 클라이언트 모형이다.

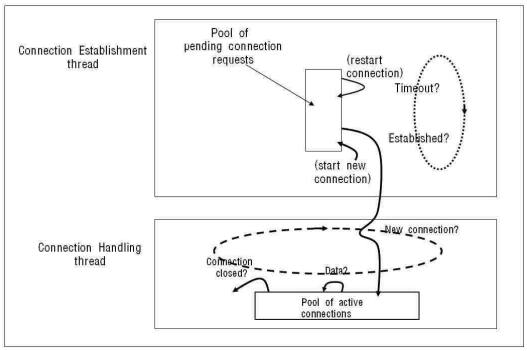
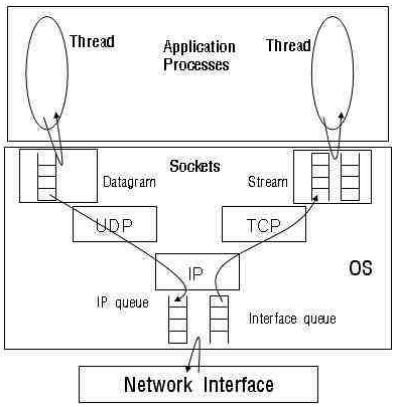


그림 7. 확장 가능한 클라이언트 모형

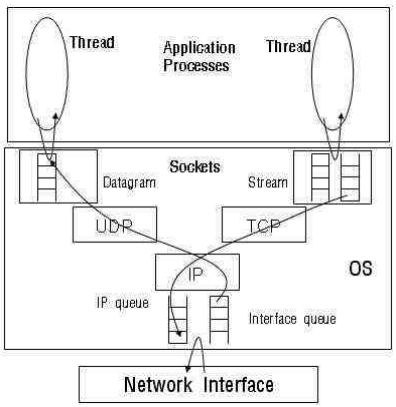
3.2 TCP와 UDP를 이용한 application level에서의 실제적 서버 성능 측정 방법 제안

서버의 성능을 테스트하기 위해 클라이언트는 서버에게 빈번한 요청(request)를 보내게 된다. 호스트간의 전송 지연에 관한 테스트에서 ping을 사용하는 경우가 있으나, ping의 경우 Network Layer간의 delay 측정이라, 서버 프로세스 부하, 시스템 상태 등등에 영향을 받는 실제 application의 데이터 송수신 delay와는 차이가 있다. 즉, TCP 통신상의 딜레이는 delayed ACK, nagle algorithm 등등의 영향도 받기 때문에, 단순히 ping으로 측정한 delay를 기반으로 계산하는 경우 서버의 성능을 측정할 수 없게 된다.

그림 7에서의 CET(Connection Establishment Thread)는 application level에서의 실제적인 서버 성능 측정을 위해 아래 그림 8.와 같이 구현하였다. 서버에게 요청(request)은 UDP 패킷을 사용하여 보내고, 서버는 클라이언트와 연결된 TCP stream을 통해 응답메시지를 보내게 된다. 이때 전송된 1, 2, 3, 4번의 UDP 패킷이 stream을 통해 1, 4번만 도착했다면 UDP 송신 손실률을 50%라고 측정하게 되고, 보낸 UDP 패킷에 time stamp를 찍어 돌아온 패킷과의 전송 지연시간도 측정하게 된다.



a. 클라이언트



b. 서버

그림 8. application level에서의 서버 성능 측정 모형

Ⅳ. 결론

본 논문에서는 application level에서의 서버의 성능을 평가하기 위한 확장 가능한 클라이언트 모형을 제시하였다. 현실세계와 같이 많은 클라이언트를 테스트할 수 없으므로, 하나의 프로세스안에 여러개의 스레드를 생성시켜 서버에 빈번하게 접

속하게 하였다.

이때 표준안으로 채택되어 활용하고 있는 POSIX 스레드로 구현하여 다른 시스템과의 호환성을 가지게 되었다. 클라이언트는 서버에 빈번한 접속을 위해 요청 패킷을 UDP를 사용하여 지속적인 요청을 하게 되고, 서버는 클라이언트와 TCP stream을 연결하여 응답 패킷을 전송하게 된다. Network Layer에서의 테스트는 서버의 application에서 데이터 송수신의 지연과는 많은 차이를 보이고 있다. 그러므로 UDP 패킷으로 요청하고 TCP stream으로 응답하여 전체 패킷의 손실률 및 각 패킷의 time stamp를 비교하여 application level에서의 실질적인 서버 성능을 평가할 수 있다. 하지만 UDP의 경우 TTL을 설정하여 패킷의 life time을 지정할 수 있고, 전송을 제한하는 네트워크도 있어 실질적인 WAN에서의 테스트에는 제한되어진다.

향후에는 LAN환경에 WAN환경을 테스트하기 위해 라우터를 추가하여 보다 실질적인 테스트가 되도록 한다면 보다 application level에서의 서버 성능 평가를 보다 유용하게 활용할 수 있다.

참고문헌

[1] 이형봉, 백청호(2002) : POSIX 스레드에 의한 재귀적 알고리즘의 병렬화에서 병렬성 제어 방안. 정보처리학회 논문지 2002.

[2] C. Maltzahn, K. J. Richardson, and D. Grunwald(1997) :Performance Issues of Enterprise Level Web Proxies. In Proceedings of the ACM SIGMETRICS 1997.Conference, Seattle, WA, June 1997.

[3] Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J.Worrell(1996) :Hierarchical Internet Object Cache. In Proceedings of the 1996 Usenix Technical Conference, Jan. 1996.

[4] Gaurav Banga, Peter Druschel(1997) :Measuring the Capacity of a Web Server. USENIX Symposium on Internet Technologies and Systems 1997.

[5] Moshe Bar, "kHTTPd, a Kernel-Based Web Server", Linux J., 2005

[6] Paul Barford, Mark Crovella(2000) : Critical Path Analysis of TCP Transactions. SIGCOMM 2006.

[7] TUX "Answers from Planet TUX: Ingo Molnar Responds",

[8] Vivek S. Pai, Peter Druschel, and Willy Zwaenpoel, "Flash: An Efficient and Portable Web Server", In Proceedings of the USENIX 1999 Annual Technical Conference, 2007

저자약력

이 승 규 (Seung-Kyu Lee) **종신회원**

2002년 한국교육개발원 기계공학과 공학사
2910년 평생진흥교육원 제어계측공학 공학사
2011년 평생진흥교육원 자동차공학 공학사
2012년 중부대학교 인문산업대학원 공학석사
2012년-현재 관동대학교 전자통신공학과 박사과정

<관심분야 >자동차 전자 제어, 영상처리, 영상압 축

박 영 록 (Young-Rok Park) **종신회원**

중부대학교 인문산업대학원 자동차관리학과(공학석사)
관동대학교 전자통신공학 박사과정
중부대학교 자동차관리학과 산학협력 교수
2013년 4월~현재 관동대학교 전자통신공학과 박사과정

<관심분야 >자동차 전자 제어, 영상처리, 신호처리시스템

이 건 화 (Geon-Wha Lee) **종신회원**

2011년 2월 중부대학교
자동차관리학과(공학석사)
2013년 4월~현재
관동대학교 전자통신공학과
박사과정

<관심분야 >자동차 전자 제어, 영상처리, 신호처리시스템

배 철 수(Cheol-Soo Bae) **종신회원**

1979년 명지대학교
전자공학과 공학사
1981년 명지대학교 대학원
공학석사
1988년 명지대학교 대학원
공학박사
1999-2001년 관동대학교
공과대학 학장
1981년 현재 관동대학교
의료공학과 교수

<관심분야 >영상처리, 신호처리시스템, 영상압축