제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

웹서버 과부하 방지 및 서비스 차별화 기법

심영석*, 이상문*, 남의석*, 김학배*, 장휘**
*연세대학교 전기전자공학과
**(주)엔버젼스
e-mail:hbkim@yonsei.ac.kr

An Overload Protection and Service Differentiation Scheme for Webservers

Youngseok Sim*, Sangmoon Lee*, Euiseok Nahm*, Hagbae Kim*, Whie Chang**
*Dept of Electrical & Electronic Engineering, Yonsei University
**Nvergence, Inc.

요 약

QoS를 보장하기 위해서는 웹 서버의 용량을 충분히 확보하는 것이 중요하다. 하지만 웹 서버의 용량을 충분히 확보한다고 하더라도 웹 트래픽의 돌발적인 속성으로 인해 서버 과부하는 거의 필연적으로 발생한다. 본 논문에서는 수락 제어 기법과 트래픽 세이핑 기법을 하이브리드하여 웹 서버의 과부하를 방지하는 커널 레벨 메카니즘을 제안한다. 수락 제어에 의해 웹 서버로 유입되는 트래픽을 제한하고, 지속 연결이 서버 과부하에 미치는 영향에 대해서는 트래픽 세이핑을 통해 제한하여 웹 서버가 최적의 상태에서 서비스를 계속할 수 있도록 한다.

1. 서론

최근 몇 년 사이 고속 인터넷의 대중적인 보급 및 관련 인프라의 확충으로 인하여 인터넷 사용 인구가 급격하게 증가하게 되었고 이에 따라 인터넷 사용량 또한 폭발적으로 증가하고 있는 추세이다. 이와 같이 증가하는 사용자를 수용하기 위해서는 네트워크 대역폭과 함께 웹 서버의 용량을 충분하게 확보해야 한다. 그렇지 않을 경우 웹 서버의 과부하 또는 네트워크의 병목 현상으로 인한 심각한 응답 지연으로 서비스의 만족도는 떨어지게 되며 상업적인 목적을 위해 인터넷 서비스를 제공하는 업체에게 이는 곧 금적적인 손실로 직결된다. 그러나 네트워크 대역폭과 웹 서버의 용량을 충분히 확보한다고 하더라도 웹 트래픽 자체의 돌발적인 속성으로 인해 순간적으로 가해지는 부하량의 변화가 극심하여 모든 상황에 완벽하게 대응할 수 있는 웹 서버를 마련하는 것은 현실적으로 불가능하고 비용 또한 문제가 된다. 따라서 필연적으로 발생하는 웹 서버의 과부하 상태를 극복하기 위한 방법이 필요하게 된다.

일반적으로 웹 서버 과부하 방지를 위해서는 유입되는 트래픽에 제한을 가하는 것이 필요하며 이와 관련하여 크게 두 가지 접근 방식이 가능하다. 하나는 연결 기반의 수락 제어 방식으로 이 기법에서는 유입되는 트래픽이 증가하여 어떤 정해진 수준에 이르게 되면 클라이언트에 의한 서버로의 연결 요청 자체를 거부함에 의해 웹 서버로 들어오는 트래픽을 제한한다[2]. 다른 하나의 접근 방식은 패킷 기반의 트래픽 세이핑 방식으로 이 기법에서 트래픽에 대한 제어는 패킷을 그 최소 단위로 하여 이루어지며 패킷은 IP 주소에 따라 특정 클라이언트군과 연관된다. 각 클라이언트군은 서로 다른 우선 순위를 가지며 우선 순위에 따라 해당 클라이언트군에 대한 패킷의 유입률을 조절함에 의해서 트래픽의 제어가 이루어진다[1].

본 논문에서는 연결 기반의 수락 제어 기법과 패킷 기반의 트래픽 세이핑 기법의 하이브리드 모델을 제안한다. 수락 제어 기법을 통해 트래픽을 제한하고 개별 사용자간 차별적인 서비스를 제공하며, 지속 연결 상에서의 부하 유동에 의한 영향은 트래픽 세이핑 기법을 적용하여 제거하도록 한다.

2. 웹 서버 과부하 방지 메카니즘

2.1 시스템 모델

본 논문에서 제안하는 모델의 전체적인 시스템

구조가 그림 1에 나타나 있다. 시스템은 수락 제어기, 트래픽 세이퍼, 부하 모니터로 구성된다.

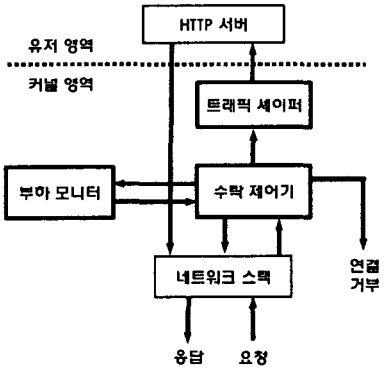

그림 1. 시스템 모델

2.2 수락 제어기

본 논문에서는 허용 여부를 결정해야 할 클라이언트를 식별하기 위해서 HTTP 헤더에 포함되는 쿠키 정보를 이용한다. 기존에는 클라이언트를 식별하는 기법에 있어 IP 주소를 이용하는 것이 일반적이었고 사실상 상태가 없는 HTTP 프로토콜로 인하여 IP 주소를 이용하는 것은 클라이언트를 식별할 수 있는 거의 유일한 방법이었다[1]. 하지만 요즘과 같이 다수의 개인 사용자들이 NAT(Network Address Translation)이나 DHCP(Dynamic Host Configuration Protocol) 서비스를 통해 인터넷 엑세스를 하는 동적 인터넷 환경에서는 IP 주소만으로 클라이언트를 식별하는 데에는 한계가 있고 따라서 이를 극복하기 위해 최근에는 HTTP 헤더에 포함된 쿠키 정보를 이용하는 방법에 대해 많은 연구가 이루어졌다[3].

쿠키는 HTTP 헤더에 포함되는 정보이므로 쿠키를 이용하기 위해서는 HTTP 헤더를 파싱하여 이로부터 클라이언트에 대한 정보를 얻어내는 기능의 구현이 필수적이다. 본 논문에서는 kHTTPd 커널 레벨 웹 데몬[4]의 HTTP Get 엔진을 이용하여 HTTP 헤더를 파싱하고 이를 통해 커널 레벨에서 수락 제어 기법을 적용한다. 논의를 단순화하기 위해서 우선 순위가 높은 클라이언트(세션, 연결)와 우선 순위가 낮은 클라이언트(세션, 연결)로만 클라이언트를 분류하며 CPU 사용율에 의해서만 서버 부하 상태를 측정하기로 한다.

서버로 유입되는 트래픽에 의해 서버의 부하가 적정한 수준 이하로 유지될 경우에는 우선 순위가 높은 클라이언트에 의한 연결이나 우선 순위가 낮은 클라이언트에 의한 연결이나 서버로의 유입이 허용된다. 본 논문에서는 이에 대한 최대 수준을 CPU 사용율의 80%로 가정하였다. 하지만 서버로의 트래픽이 증가하여 서버의 부하 수준이 80%를 초과하게 되면 우선 순위가 낮은 클라이언트에 의한 연결을 거부함에 의해서 1차적으로 서버 부하를 조절하게 된다. 하지만 이 때에도 여전히 우선 순위가 높은 클라이언트에 의한 연결은 허용되므로 서버의 부하는 계속 증가하게 되며 서버의 부하가 더욱 증가하여 90%에 이르게 되면 이 때에는 우선 순위의 높고 낮음에 관계없이 서버로의 모든 연결을 거부하여 서버 부하를 서버 처리 용량 이하로 유지하게 된다. 이에 대한 개념적 모델이 그림 2에 나타나 있다.

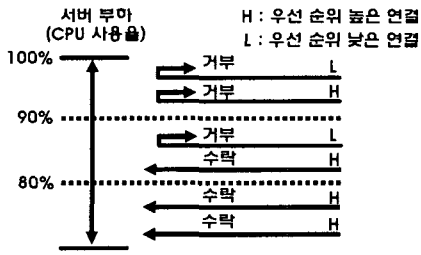

그림 2. 수락 제어

2.3 트래픽 세이퍼

서버 과부하 상황이 발생할 경우 수락 제어기에 의해 1차적으로 시스템 내부로 들어오는 트래픽에 제한이 가해진다. 하지만 이것만으로는 서버 과부하 상황을 극복하기에는 충분하지 않으며 클라이언트와 서버간 지속 연결이 서버 과부하에 미치는 영향에 대한 고려가 필요하다.

지속 연결을 사용할 경우 클라이언트는 동일한 연결 상에서 여러 요청을 서버에 보낼 수 있어 연결 설정에 따르는 오버헤드를 줄이는 것이 가능하다. 최근의 브라우저는 거의 모두 지속 연결을 지원하며 따라서 클라이언트와 서버간 연결의 대부분은 지속 연결에 의해서 이루어진다[5]. 하지만 동일한 연결이 하나의 요청이 아닌 여러 요청에 대해서 사용될 때 그 첫 번째 요청은 동일 연결 상에서 이후에 따라오는 요청에 의해 서버의 리소스가 얼마나 소모될 것인지에 대해서 어떠한 정보도 제공하지 않는다[3]. 뿐만 아니라 수락 제어기에 의해서 1차적으로 웹 서버로 들어오는 트래픽을 제한한다고 해도 허용된 세션의 지속 연결 상에서의 부하 유동으로 인해 서버

에 가해지는 부하는 서버의 처리 용량을 넘어설 수도 있게 된다. 그림 3은 허용된 세션의 지속 연결 상의 부하가 서버 처리 용량을 넘어서는 것을 보여준다.

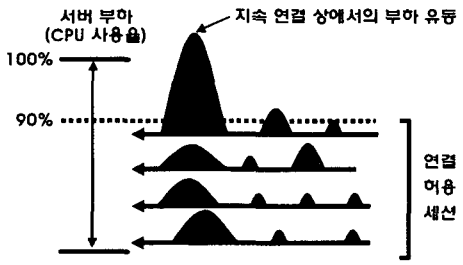

그림 3. 지속 연결 상에서의 부하 유동

따라서 허용된 세션의 지속 연결 상에서의 부하 유동이 서버 과부하에 미치는 영향을 제거하기 위해 본 논문에서는 트래픽 셰이핑 기법을 적용한다.

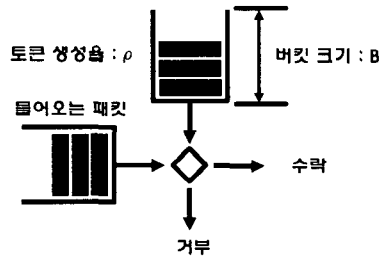

그림 4. 토큰 버킷 알고리즘

트래픽 셰이핑을 위해서 잘 알려진 토큰 버킷 알고리즘을 이용한다. 그림 4에 나타난 것과 같이 토큰 버킷 알고리즘은 토큰 공급율 ρ와 버킷 크기 B의 두 개의 파라미터로 구성된다. 들어오는 패킷은 토큰을 얻어야만 서버로의 유입이 허락되며 따라서 버킷 크기 B는 돌발적으로 들어올 수 있는 패킷의 최대 개수를 나타내며 토큰 생성율 ρ는 허용되는 패킷의 유입율을 나타낸다[6]. 이 토큰 생성율을 조절함에 의해서 허용되는 패킷의 유입률을 제어하는 것이 가능하다. 그림 5는 지속 연결 상에서 서버의 처리 용량을 넘어서는 부하가 트래픽 셰이퍼에 의해서 조절되는 것을 개념적으로 나타내고 있다.

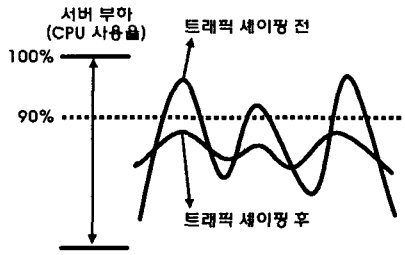

그림 5. 트래픽 셰이핑 전후의 부하 유동

2.4 부하 모니터

서버 과부하 시에 신속하게 서버 과부하 상태를 극복하는 동적 메카니즘을 구현하는 데 있어 부하 모니터는 필수적인 요소이다. 부하 모니터는 주기적으로 실행되면서 CPU 사용율, 메모리 사용량, I/O 사용량 등 서버의 부하 상태를 반영하는 인자들에 대한 측정을 통해 시스템 전체적인 관점에서 서버의 부하 상태를 평가한다. 본 논문에서는 앞서 가정하였던 바와 같이 논의를 단순화시키기 위해 서버의 부하 상태를 파악하기 위한 인자로 CPU 사용율만을 사용하기로 한다. 또한 부하 모니터는 갑작스런 부하 증가에 따른 과부하 시에도 서버의 부하 상태를 측정할 수 있어야 하기 때문에 런타임 오버헤드가 작고 과부하 상태에서도 수행이 제대로 되어야 한다. 따라서, 부하 모니터는 커널 쓰레드에 그 기능을 포함시킨 독립적인 커널 모듈의 형태로서 구현한다.

3. 실험

3.1 실험 환경

본 논문에서 제안하는 모델의 타당성을 검토하기 위해서 그림 6과 같은 실험 환경을 구성하였다.

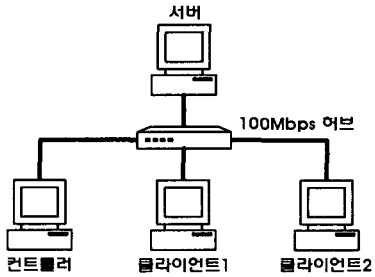

그림 6. 실험 환경

2 대의 클라이언트는 트래픽을 발생시키고 정해진 규칙에 따라 서버에 데이터를 요청하고 그 양을

점진적으로 증가시킨다. 1 대의 컨트롤러로서 실험을 전체적으로 제어하고 결과 데이터를 수집하여 그 특성을 분석한다. Intel Express10/100 Stackable Hub를 사용하여 100Mbps 기반의 고속 이더넷 환경을 구축하였고 서버에는 저속의 Intel Celeron 300MHz 프로세서와 512Mbyte 메모리를 사용하였다. 웹 서버 어플리케이션은 Apache 1.3.9를 커널은 Linux 2.4를 설치하였고 웹 서비스를 제외한 다른 기능은 모두 제거하였다. 2대의 클라이언트와 1대의 컨트롤러는 Pentium III 1GHz 프로세서와 512Mbyte 메모리, 3Com 3CSOHO100-TX NIC을 사용하였다. 실험을 통해 본 논문에서 제안된 메카니즘을 적용하기 전후의 데이터를 얻고 이를 비교한다.

3.2 결과 비교

본 논문에서 제안하는 메카니즘에 의해서 서버에 가해지는 부하가 제한되는 정도를 알아본다. 이를 위해 일반적인 웹 서버와 제안된 메카니즘이 적용된 웹 서버 2대를 마련하였고 각각에 대해 클라이언트에서 요청하는 데이터량을 단계적으로 증가시켜 각 서버가 최대로 처리할 수 있는 임계점까지 도달할 수 있도록 하였다. 그림 7은 각 경우에 대해 측정된 결과를 나타낸다. 그림 7의 결과에서 본 논문에서 제안된 메카니즘의 경우 일반적인 웹 서버에 비해 웹 서버로 유입되는 부하가 70%~80% 정도로 제한되는 것을 알 수 있다. 일반적인 웹 서버의 경우 서버로 유입되는 트래픽이 증가함에 따라 서버 처리 용량의 최대치에 이르게 되지만 본 논문에서 제안된 메카니즘이 적용된 경우 서버로 들어오는 트래픽을 적절하게 제한함에 의해서 서버의 과부하를 방지하고 있다.

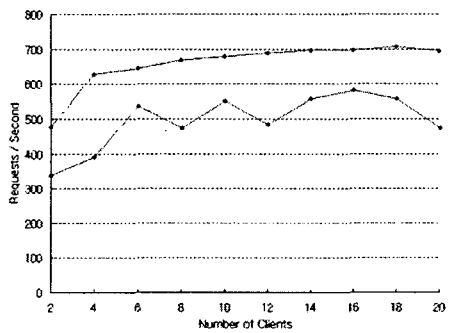

그림 7 웹 서버 과부하 방지 메카니즘 적용 전후의 부하 비교

4. 결론

클라이언트간 서비스의 차별화를 위해서는 IP 정보를 기반으로 하여 세션을 식별하고 커널 레벨의 네트워크 스택 레이어에서 해당 패킷을 처리하는 것이 일반적으로 사용되는 방식이다. 하지만 본 논문에서 제안하는 구조에서는 클라이언트간 서비스 차별화를 제공하기 위해서 HTTP 헤더의 쿠키 정보에 기반하여 클라이언트를 식별하고 뿐만 아니라 kHTTP 커널 레벨 웹 데몬을 이용하여 커널 레벨에서 패킷에 대한 처리를 한다. 따라서 DHCP나 NAT와 같은 동적 인터넷 환경에서 더 효과적으로 대처하는 것이 가능하고 또한 서버 과부하 상황이 발생하게 되면 해당 요청에 대해 서버 시스템 자원의 할당이 이루어지기 이전에 신속하게 이에 대응하는 것이 가능하다.

또한 연결 기반의 수락 제어 기법과 패킷 기반의 트래픽 셰이핑을 함께 사용하여 서버로 들어오는 트래픽을 적절하게 제한함에 의해서 서버 과부하 상태를 미연에 방지하여 트래픽이 폭주하는 경우에도 서버를 최적의 상태로 유지하면서 클라이언트간에는 우선 순위에 따라 차등적인 서비스를 제공하는 것이 가능하다.

참고문헌

[1] Hani J, John R, Kang G. Shin, "QGuard: Protecting Internet Servers from Overload" Realtime Computing Laboratory, University of Michigan, 2000.

[2] L. Ckerkasova, P. Phaal, "Session Based Admission Control: a Mechanism for Improving Performance of Commercial Web Sites", Technical Report, Hewlett Packard, 1999.

[3] T. Voigt, P. Gunningberg, "Kernel-based Control of Persistent Web Server Connections", ACM Performance Evaluation Review, pp. 20-25, September 2001.

[4] kHTTPd, http://www.fenrus.demon.nl.

[5] Persistent Connection Behavior of Popular Browsers, http://www.cs.wisc.edu/~cao/papers/persistent-connection.html

[6] William Stalling, "HIGH-SPEED NETWORKS: TCP/IP and ATM Design Principles", Prentice-Hall International, Inc., 1998