Persistent Connection을 지원하는 웹서버 모델링 및 성능분석

민 병 석[†]·남 의 석[††]·이 상 문[†]·심 영 석[†]·김 학 배[†††]

요 약

웹서버가 처리하는 웹 트래픽 양이 폭발적으로 증가하고, 다양한 형태의 웹 서비스에 대한 웹서버의 성능 개선이 요구되고 있다. 이를 위해, HTTP 트래픽의 특성에 대한 분석과 웹서버의 적절한 튜닝이 요구되고 있지만 이에 대한 연구는 아직 미진한 상태이다. 특히, 현재 대부분의 어플리케이션이 HTTP 1.1에 기반하여 구현되고 있음에도 불구하고, 대부분의 연구들이 HTTP 1.0에 기반하여 성능 분석이 이루어진 반면 HTTP 1.1에 대한 모델링과 성능분석은 거의 이루어지지 못하였다. 따라서, 본 논문에서는 Persistent Connection을 지원하는 HTTP 1.1 프로토콜을 기반으로 하여 서버내의 세부 하드웨어 특성 등을 고려하여, 웹서버가 사용자의 요청을 받아들이면서부터 서비스를 마칠 때까지의 과정을 Tandem 네트워크 큐잉 모델을 사용하여 해석적인 웹서버 모델을 제안한다. 그리고, HTTP 1.0에 대한 HTTP 1.1의 개선된 점과 과부하 하에서의 문제점 등을 분석하고, 웹 서버에 요청하는 파일크기, 파일전송 사이의 OFF 시간, 요청빈도, 요청시간에 대한 지역성과 같은 HTTP 트래픽에 대한 특성을 분석한다. 제안된 모델은 실제 웹서버에서 웹 서비스 요청율의 변화에 따른 서버의 처리량에 대한 비교를 통해 검증하였다. 또, HTTP 1.1 기반의 웹서버에 있어서, TCP 요청 대기큐 크기와 HTTP 쓰레드의 개수 및 네트워크 버퍼 크기와의 상관 관계에 따른 웹 서버의 성능분석을 하였다.

Modeling and Performance Evaluation of the Web server supporting Persistent Connection

Byung seok Min[†]·Eui seok Nahm[††]·Sang moon Lee[†]
Young seok Sim[†]·Hag bae Kim[†††]

ABSTRACT

Amount of the web traffic web server handles are explosively increasing, which requires that the performance of the web server should be improved for the various web services. Although the analysis for the HTTP traffic with the proper tuning for the web server is essential, the research relevant to the subject are insignificant. In particular, although most of applications are implemented over HTTP 1.1 protocol, the researches mostly deal with the performance evaluation of the HTTP 1.0 protocol. Consequently, the modeling approach and the performance evaluation over HTTP 1.1 protocol have not been well formed. Therefore, basing on the HTTP 1.1 protocol supporting persistent connection, we present an analytical end-to-end tandem queueing model for web server to consider the specific hardware configuration inside web server beginning at accepting the user request until completing the service. we compare various performances between HTTP 1.0 and HTTP 1.1 under the overloading condition, and then analyze the characteristics of the HTTP traffic that include file size requested to web server, the OFF time between file transfers, the frequency of requests, and the temporal locality of requests. Presented model is verified through the comparing the server throughput according to varying requests rate with the real web server. Thereafter, we analyze the performance evaluation of the web server, according to the interrelation between TCP Listen queue size, the number of HTTP threads and the size of the network buffers.

키워드 : 웹서버(Web server), 큐잉 모델(Queueing model), HTTP 트래픽(HTTP traffic)

1. 서 론

현재의 웹서버는 일반적으로 TCP/IP에서 동작하는 HTTP 프로토콜을 기반으로 서비스를 하고 있으며, HTTP 프로토콜은 브라우저에 통해 구현되어 있는 프로토콜으로, 패킷 네트워크 대부분의 트래픽을 차지하고 있다. 특히 시간이 지날수록 웹 서비스에 대한 요청이 폭발적으로 증가하고 실시간 웹 서비스 및 다양한 멀티미디어 서비스에 대한 웹 서버의 성능 개선이 요구되고 있다. 이에 대해 웹 서비스 요청률이나 요청되는 파일 사이즈의 분포, 전송시간, 유저

※ 본 논문은 산업자원부 중기거점과제인 "초고속 Scalable Web Server 개발" 사업에 의해 지원되었습니다.
† 준 회 원 : 연세대학교 대학원 전기전자공학과
†† 정 회 원 : 연세대학교 전기전자공학과 BK21 연구교수
††† 정 회 원 : 연세대학교 전기전자공학과 교수
논문접수 : 2002년 3월 13일, 심사완료 : 2002년 6월 20일

세션 같은 웹 트래픽 특성에 대한 기존의 많은 연구들[1-3]을 통하여 해석적으로 웹 트래픽을 모델링 하였다. 또한 이와 연관하여, 연결 지향적인 TCP 위에서 구현되는 HTTP 프로토콜에 대하여 접속 연결 과정 및 접속 연결 유지 시간의 변화에 따른 웹 요청 처리 성능에 대해 HTTP 1.0과 HTTP 1.1의 여러 변수들에 대한 각각의 성능분석에 대한 연구[1, 2, 5, 6, 9]가 이루어져 왔다. 특히 HTTP 1.0에 기반한 웹서버의 해석적 모델링[1, 6]에 대한 연구가 이루어졌지만, 그 대부분은 웹서버 내의 메모리 관리 방식이나 서버 하드웨어 플랫폼 및 네트워크 대역폭, I/O 버퍼 관리, 파일 사이즈, 캐쉬 메모리 관리 같은 HTTP와 TCP/IP 하위레벨의 세세한 과정 및 요소들을 간과한 채 모델링 되어, 웹 서비스 요청율과 전체 서비스 처리시간, 프로세서 개수 같은 단지 몇 개의 변수만을 가지고 웹서버의 성능을 평가하기에는 큰 무리가 있었다. 그리고, Persistent Connection을 지원하지 않는 HTTP 1.0 프로토콜에 대한 모델만이 제시되었기 때문에 현재 대부분의 브라우저가 지원하는 HTTP 1.1 프로토콜에 대한 모델에 적용하기에는 무리가 있다. 따라서, 본 논문에서는 Queueing 웹서버 모델로서 Persistent Connection을 지원하고, 쓰레드 개수나 TCP 요청 대기큐 개수, 메모리 크기나 I/O 버퍼 및 네트워크 대역폭 같은 하드웨어 및 소프트웨어적인 세부요소를 고려하여, 웹 요청이 들어오면서부터 TCP 서브 시스템과 HTTP 서브 시스템 그리고 Network 서브 시스템의 3개 서브 시스템을 거치면서 웹 요청을 처리해 나가는 Tandem Queueing 모델을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 HTTP 프로토콜에 대한 전반적인 설명과 HTTP1.0과 1.1의 차이점 분석 및 성능분석에 대한 관련 연구들이 서술되고 3장에서는 HTTP 트래픽에 대한 분석과 웹 Workload의 특성 분석이 이루어진다. 4장에서는 3-서브 시스템 Tandem Queueing 모델이 제시되고 각 부분 요소들에 대한 성능평가가 이루어진다. 5장에서는 실제 시스템과의 벤치마킹을 통하여 제안한 웹서버 모델을 검증하고, Discrete Event 시뮬레이션을 통해 각 변수의 변화에 따른 시스템 성능을 평가하고 6장에서는 결론을 맺는다.

2. Hyper Text Transfer Protocol

TCP 계층에서의 HTTP 트래픽에 대한 모델링[2, 5]과 HTTP 프로토콜의 성능 개선에 대한 연구[12]가 이루어져 왔다. 과거 웹 트래픽의 대부분을 차지한 HTTP 1.0 프로토콜은 클라이언트가 서버에 SYN 메시지를 보내고 서버가 이에 대해 SYN- ACK를 보내면 마지막으로 클라이언트가 이에 ACK 메시지를 보내어 한 개의 접속을 형성하는 3-way Handshaking을 통하여 연결을 맺은 후[15], 새 포트 번호와 해당 자원할당 및 서비스 요청에 해당하는 적절한 자료구조를 생성한다. 따라서 접속 연결 초기화에 많은 오버헤드가 발생한다. 연결 설정 후 요청한 파일을 다운받으면 그 접속은 바로 해제되고 다음 파일의 요청에 대하여 다시 똑같은 오버헤드를 가지고 Connection Setup을 맺고 파일 전송을 하는 방식을 취하고, 한 페이지에서 여러 파일을 다운받을 때 다중 동시 병렬 연결을 지원함으로써 이 단점을 보완하였다. 이때 웹 파일들은 보통 이미지 파일 같은 파일들을 참조하고 있기 때문에 한 단일 웹 파일에 대한 요청은 웹서버로부터 다수의 파일들을 전송하게 한다. 따라서 앞으로 우리는 전송되어야 할 모든 파일들을 포함하는 한 웹파일을 "웹 객체"라고 부르기로 한다.

하나의 연결에 한개의 파일만을 전송하는 HTTP 1.0에 반해 HTTP 1.1은 요청하는 파일을 다운받은 후에도 일정 시간동안 연결을 유지함으로써 하나의 연결에 여러 파일들을 전송받게 함으로써, 연결 초기화에 드는 오버로드를 생략할 수 있다. 또한, 파이프라인을 통하여 같은 연결 상태 상에서 그 전 파일의 전송을 완료하기 전에 다음 파일에 대한 요청을 함으로써, 불필요한 자원낭비를 줄이고, 전송시간을 줄이게 된다. 추가적으로 Link 레벨에서 문서 압축을 지원하고 캐시를 지원한다. 따라서, 일반적인 상태에서는 HTTP 1.1이 HTTP 1.0보다 웹 서비스 요청에 대한 서비스 시간에서 우월한 성능을 보이지만, 과부하 상태에서는 그렇지 못하다. 일반적으로 HTTP 1.1은 새 연결을 맺는 데에 대한 오버로드는 감소하지만, 연결지속 시간동안 접속 상태를 유지함으로써 HTTP 1.0에 비해 작은 수의 접속을 유지할 수 있으므로 과부하시 HTTP 1.1은 가능한 동시 연결수가 최대 제한값에 도달하면 새 연결에 대한 요청이 거부되어 응답시간 측면에서 HTTP 1.0에 비해 현저히 저하된 성능을 보인다.

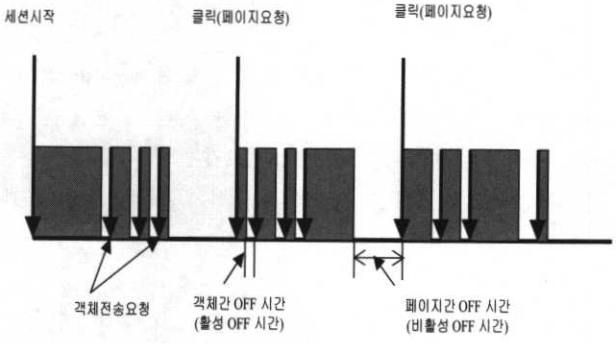
HTML latency 측면에서는, HTTP 1.0의 연결들이 비록 모든 TCP 연결 설립 과정과 TCP slow start 과정을 거친다 하더라도 다중 동시 병렬 연결을 통하여 더 작은 latency를 갖게 되기 때문에, 단일 연결상에서 한 페이지의 모든 객체들에 대한 요청을 나열하는 HTTP 1.1이 우수하지 못하다. HTTP 1.1 기반에서는 네스케이프 브라우저가 6개의 병렬 Persistent Connection을, 익스플로어 브라우저가 2개의 동시 병렬 Persistent Connection을 사용하여 요청된 파일들을 더 신속하게 제공하여 개선된 성능을 보일 것 같지만, 실제 HTTP 1.1에서 병렬 연결의 수가 많을수록 응답시간이 지수적으로 증가한다[1]. 또한, 대역폭이 좁은 경우에만 다중 병렬 연결로 인한 이득이 있지만 대역폭이 증가하면 그 이득이 사라지게 되므로, 실질적으로 작은 병렬 연결의 수를 유지하는 것이 HTTP 1.1의 성능을 개선시키는 방법 중 하나이다.

웹 서버의 성능이 파일전송에 의해 제한 받을때 HTTP

1.1처럼 연속적인 웹 객체들을 전송하고 타임 아웃시간까지 연결상태를 유지하는 것이, HTTP 1.0처럼 단일 객체의 전송하는 동안만 연결을 유지하는 것보다 저하된 성능을 보일 수 있다. 따라서, 웹 객체를 전송하는 동안만 지속연결 및, 파이프라인 연결상태를 유지하고, 일련의 웹 객체들의 전송후에 연결을 닫는 메커니즘을 고려한 것이 HTTP 1.1/EC (HTTP 1.1/Early-Close)이다. 이것은 클라이언트가 각각의 객체들에 대해 GET 대신 GETALL 요청을 하여 현재 페이지에 있는 모든 객체들에대한 전송을 요구하면, 서버는 전체 웹 객체들을 전송하고 응답하는 형태를 지닌다. 따라서 각 객체 당 K개의 파일들이 내재되어 있고, 클라이언트가 이런 객체를 N개 요청한다고 하면 HTTP 1.0에서는 KN개의 연결이 필요하며, HTTP 1.1에서는 Persistent Connection이 보장되므로 1개의 연결만이 필요하고, HTTP 1.1/EC에서는 N개의 연결만이 필요하다. 즉 HTTP 1.1/EC는 연결 개수에 있어서 HTTP 1.1보다 많기 때문에 더 많은 초기화 시간을 필요로 하지만, 과부하 상태에서는 타임아웃 시간까지 연결을 지속할 필요가 없으므로 더 많은 연결을 만들 수 있고, HTTP 1.0보다는 적은 연결 수를 가지므로 상대적으로 적은 초기화 시간을 요구한다는 장점이 있다. 하지만 과부하가 아닌 보통상태에서는 여전히, Persistent Connection을 지원하는 HTTP 1.1이 더 우수한 성능을 보인다[1, 12].

3. HTTP 트래픽 모델링

HTTP 1.1에서 사용자 Session은 (그림 1)에서 볼 수 있듯이 웹 서버가 가지고 있는 어떤 페이지에 대한 요청을 함으로써 시작하고, Persistent Connection 지속시간이 만료됨으로써 종료하며, 요청된 페이지에 내재된 객체들에 대한 연속적인 전송 요청 및 그 파일에 대한 전송으로 이루어진다. 전송과 전송사이에 파일전송이 이루어지지 않는 시간을 OFF 시간이라 하고, OFF 시간은 각 객체 전송사이에 이루어지는 활성 OFF 시간과 한 페이지에 대한 전송이 모두 이루어진 후 다음 페이지에 대한 요청이 있을 때까지 이루어지는 비활성 OFF 시간으로 이루어진다. 보통 비활성 OFF 시간은 사용자가 브라우저를 보면서 생각하는 시간에 해당하고, 활성 OFF 시간은 단일 웹 객체를 구성하는 파일들의 전송간에 브라우저가 웹 파일들을 파싱(parsing)하고 새로운 TCP 연결을 시작하기 위해 준비하는데 소요하는 처리 시간에 해당한다. HTTP 1.1은 비활성 OFF 시간이 정해진 타임아웃 시간보다 작을 때까지는 연결을 지속하게 하고, 요청한 파일에 대한 전송이 이루어진 후 ACK를 기다리지 않고 연이어서 다음 파일의 전송을 시작하는 파이프라인을 지원한다. 이때, 한 객체를 전송받고 나서 다음 객체에 대해 전송을 시작할 때까지의 OFF 시간(활성 OFF 시간)과 한 페이지를 모두 전송받고 나서 클릭한 후 다음 페이지에 대한 요청이 있기까지의 OFF 시간(비활성 OFF 시간), 웹서버가 제공하는 파일 크기, 페이지의 이동을 나타내는 클릭 횟수, 파일 전송 시간등은 모두 랜덤변수로 정의되며 웹서버에서 서비스하는 파일에 대한 요청빈도와 시간적인 지역성과 함께 시스템의 특성을 결정한다.



(그림 1) 사용자 Session 정의

3.1 파일 크기 특성

웹 서버의 서비스 시간이 주로 파일의 전송시간에 지배하면, 결국 파일이 전송되는 시간은 요청되는 파일의 크기에 비례한다고 할 수 있다. 웹 서버에서 파일 크기의 분포는 보통 heavy-tailed distribution을 따른다[13]. 이것은 보통 클라이언트에 의해 요청된 파일 사이즈와 네트웍 연결길이, 그리고 서버에 저장된 파일의 크기를 통해 알 수 있다. 만약 랜덤 변수 X 가 heavy-tails distribution을 따르면, $P[X > x] = x^{-\alpha}$ 이며, $0 < \alpha < 2$ 이다. 보통 파일 사이즈 분포의 몸통 부분은 lognormal 분포를 가지고, 꼬리 부분은 $0.40 < \alpha < 0.63$ 의 매개변수를 가진 $\alpha k^{\alpha} x^{-(\alpha+1)}$의 확률밀도 함수를 가지는 Pareto 확률분포를 가진다[20]. Pareto 모델은 4KB에서 4MB 사이의 파일 크기에 적합하고, 긴 꼬리 분포를 가진다[19]. Heavy tailed distribution을 따르는 랜덤 변수는 보통 사이즈에 있어서 극히 높은 변동성을 보여준다. 그리고, 보통 아주 적은 개수의 매우 큰 용량의 파일이 웹 서버에서 부하의 대부분을 차지하고 파일의 대부분은 작은 크기의 용량을 가지는 파일에 밀집되어 있다.

3.2 OFF 시간 특성

파일크기를 나타내는 확률밀도 함수를 F (x), 연결지속 종료시간을 T_{out}, 그리고 활성 OFF 시간을 나타내는 랜덤변수를 Y라고 하면, 타임아웃 시간이 지나 세션이 종료되는 확률 P는 $P[Y > T_{out}]$ 으로 나타낼 수 있고, 연결지속시간이내에 활성 OFF 시간의 평균값은 식 (1)과 같이 나타낼 수 있다.

$$E[Y|Y<T_{out}]=\frac{1}{1-p}\int_0^{T_{out}}F(x)dx \qquad (1)$$

그리고, 활성 OFF 시간은 보통 식 (2)와 같은 확률밀도함수를 가지는 Weibull 확률분포를 가지고, 비활성 OFF 시간은 $\alpha k^{\alpha}x^{-(\alpha+1)}$ 의 확률 밀도함수를 가지는 Pareto 확률분포를 가진다[1, 4].

$$P(y)=\frac{bx^{b-1}}{a^b}e^{-(\frac{x}{a})^b} \qquad (2)$$

이와 같이 OFF 시간에 대한 분석을 통해 웹서버 모델에 사용자가 브라우저를 보면서 생각하는 시간과, 단일 웹 객체를 구성하는 파일들의 전송간에 브라우저가 웹 파일들을 파싱하고 새로운 TCP 연결을 시작하기 위해 준비하는데 소요하는 처리 시간을 적용할 수 있다.

3.3 요청 빈도

각 파일에 대한 상대적인 요청횟수는 캐쉬의 역할에 강한 영향을 미친다. 파일에 대한 요청빈도의 분포는 Zipf's Law[4, 14]를 따르는데, 이는 만약 파일이 가장 요청빈도가 높은 것에서 낮은 순으로 정렬되어 있으면, 한 파일에 대한 참조회수를 P라 할 때 참조회수는 그것의 랭크 r에 반비례하여 $p=kr^{-1}$이 되고 이때 k는 임의의 양수가 된다. 이 성질은 비록 웹 workload에서 이런 현상에 대한 이유가 불분명함에도 불구하고 관찰할 수 있는 현상이다. 웹 파일들간에 참조들의 이러한 분포는 어떤 파일은 극히 요청빈도가 높은 반면에 대부분의 파일은 상대적으로 적은 참조 횟수를 가짐을 나타낸다.

3.4 시간적인 지역성

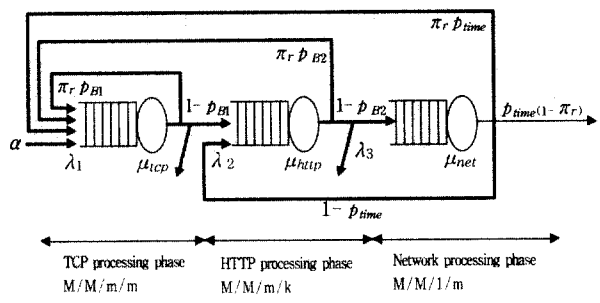
보통 한 파일이 요청되면, 그 파일은 얼마 후에 다른 사용자에 의해 다시 요청되는 경향이 있다. 그리고, 캐쉬효과는 시간적인 지역성이 존재할 때 크게 증가하므로 시간적인 지역성의 정확한 특성화가 요구된다. 이를 위해 보통 스택 거리측정[14]을 사용하는데, 파일들이 아래방향으로 밀어 넣는 스택에 저장되어 있다고 할 때, 요청된 파일을 스택의 맨 위로 이동하고, 다른 것은 밑으로 밀어 넣으면, 각 요청된 파일이 발견되는 스택 깊이가 요청된 파일의 스택 거리이다. 즉 스택 거리가 작으면 서로 가까운 시간에 발생하는 같은 파일에 대한 요청을 의미하고, 초기 스택의 내용을 알 때, 스택 거리 배열은 그 요청에 대한 배열과 동등한 정보를 가지고 보통 Lognormal 분포를 가진다. 하지만 이 스택에 대한 공통 리스트를 공유할 수 없으므로, 멀티캐스트 운영시에는 동기화에 따른 오버헤드가 요구된다.

4. 웹 서버의 해석적 모델링

4.1 Tandem Queueing 모델

대부분의 웹 서버 모델링[1, 8, 13]은 HTTP 하위의 필수적인 세부사항은 무시하고 단지 파일에 대한 요청과, 파일의 전송 등에 의해서만 이루어졌으므로 클라이언트와 서버간의 통신에 대한 종단간 성능을 보여주기에는 부족하다. 이에 반해, [6]은 웹 서비스를 TCP 연결을 맺는 단계와 HTTP 요청을 처리하는 단계, 그리고 이에 따른 I/O 작업 단계로 나누어 각 요소에 대한 성능평가가 가능하도록 하였고, [7]은 HTTP 1.0기반에서 동적인 요청에 따른 웹 서비스 모델로 확장하였다. 그러나, [6]은 한 웹 요청이 TCP 3-way Handshaking에 의해 받아들여진 후 세 개의 Tandem 큐들을 모두 거치면 서비스를 종료하고 다시 요청을 해야하는 HTTP 1.0기반으로 모델링되었기 때문에 Persistent Connection을 지원하는 HTTP 1.1의 모델로는 적합하지 못하다. 또한, 3개의 각 서브 시스템으로 들어가는 트렌젝션 요청은 단지 그 앞에 위치한 큐에서 서비스를 받고 나온 트렌젝션만을 고려하였기 때문에, 그 뒤에 위치한 큐에서 차단되어 앞에 위치한 큐로 다시 들어가는 트렌젝션 요청은 고려하지 않았다.

본 논문에서 제시하는 웹 서버 모델은 [6]에서 제시하는 큐잉모델을 기반으로 웹 서비스를 3개의 서브 시스템을 거치는 Tandem Queueing 모델로 나타내었지만 그와는 다르게, 각 서브 시스템에서 서비스가 이루어질 수 없을 때 재시도하는 상황과, HTTP 1.1에서 제공하는 연결지속시간을 고려한 세션종료 확률에 따른 시스템 변화를 고려한다. 전체적인 시스템 모델은 아래 그림처럼, TCP 서브 시스템과 HTTP 서브 시스템, 그리고 Network 서브 시스템의 3개의 서브 시스템이 Tandem으로 연결된 잭슨 네트워크(Jackson Network)[16]로 해석될 수 있다.



(그림 2) 잭슨 네트워크에 의한 웹서버 모델

일반적으로 웹 서비스에서 사용자의 요청은 대기큐에 여분이 없거나 요구하는 자원이 사용 중이어서 거부될 경우 시스템 안으로 다시 요청되므로 잭슨 네트워크로 해석될

수 있다. 잭슨의 이론에 의하면, 임의의 시간 t에 각 큐들에 있는 트랜잭션 요청의 개수는 독립적인 랜덤 변수로 해석될 수 있고, 정상상태에서 세개의 큐에 각각 N_1, N_2, N_3개의 트랜잭션 요청이 있을 확률은 TCP 큐에 N_1개, HTTP 큐에 N_2개, 네트워크 큐에 N_3개 있을 확률을 곱한 것과 같다 [16]. 따라서 웹 서버에 들어오는 외부 세션 도착율을 α라 하고, TCP 요청 대기큐(TCP Listen Queue)에 여분이 없어서 더 이상 요청을 받아들일 수 없게 되어 TCP 큐에서 차단될 확률을 P_{B1}, HTTP 쓰레드를 사용하기 위해 기다리는 HTTP 요청 대기큐(HTTP Listen Queue)에 여분이 없어서 차단될 확률을, P_{B2} 그리고 TCP 연결을 유지하는 타임 아웃 시간이 지나 세션이 종료될 확률을 P_{time}이라 하며, 각 서브 시스템에서 차단되었을 때 시스템을 떠나지 않고 다시 재시도할 확률을 π_r이라 할 때, 각 서브 시스템에서 트랜잭션 요청에 대한 관계는 식 (3)~식 (5)와 같이 구할 수 있다.

$$\lambda_1 = (\pi_r P_{B1})\lambda_1 + (\pi_r P_{B2})\lambda_3 + \alpha \tag{3}$$

$$\lambda_2 = (1 - P_{B1})\lambda_1 + (1 - P_{time})\lambda_3 \tag{4}$$

$$\lambda_3 = (1 - P_{B2})\lambda_2 \tag{5}$$

4.2 TCP 연결 설정 서브 시스템

TCP 서브 시스템은 서버 데몬이 서비스하는 TCP 요청 대기큐로 구성된다. TCP 연결 설정은 세방향 핸드쉐이크 과정에 의하여, 클라이언트가 서버에 SYN 을 통해 서비스 연결 요청을 하고, 이에 대해 서버가 자신의 TCP 요청 대기큐에 빈 슬롯이 있으면, 서버 데몬은 한 슬롯을 할당하고 SYN-ACK를 보낸다. 이때 만약 빈 슬롯이 없다면 연결 거부 메시지를 보내고, 그 요청은 버려진다. SYN-ACK를 받은 뒤, 클라이언트는 이에 대해 ACK와 GET같은 트랜잭션 요청을 같이 보내고, 서버가 그 요청을 받자마자 TCP 요청 대기큐에 할당되어 있던 슬롯은 해제된다. 이후 TCP 소켓 생성 후 서버 데몬은 그 요청을 HTTP 서브 시스템으로 포워드(forward)하고, 그 트랜잭션이 모두 완료되면 서버는 클라이언트에게 FIN을 보내고 소켓을 닫는다.

TCP 연결 설정 단계에서 서버는 요청 대기큐에 있는 슬롯의 개수 만큼만 들어오는 요청을 서비스 할 수 있고, 슬롯이 꽉 찬 뒤의 요청은 기다리는 공간이 없이 그냥 버려지기 때문에, 서버의 개수가 슬롯 개수 m_{tcp}이고, 서비스 시간은 서버가 SYN-ACK를 보낸 후부터 클라이언트로부터 ACK를 받을 때까지의 시간이므로 결국 RTT(Round Trip Time)값을 가지게 된다. 따라서 RTT 값이 평균이 μ_1인 지수분포를 이루고, 큐의 사이즈가 m_{tcp}이면 이 서브 시스템은 $M/M/m_{tcp}/m_{tcp}$ 큐잉 모델로 해석될 수 있다. 따라서 이 서브 시스템으로 들어오는 서비스 요청율 λ_1은 식 (3)~식 (5)로부터 유도할 수 있고, 이 서브 시스템에서의 서비스 처리율 μ_1과 TCP 요청 대기큐가 꽉 차서 추가로 들어오는 요청이 블라킹 될 확률 P_{B1}, 그리고 TCP 서브 시스템에 있는 평균 트랜잭션 요청의 수 $E[N_{tcp}]$는 다음과 같이 구할 수 있다.

$$\lambda_1 = \frac{\alpha}{(1-P_{B1})(1-\pi_r)}, \quad \mu_1 = \frac{1}{RTT} \tag{6}$$

$$P_{B1} = \frac{a^{m_{tcp}}}{m_{tcp}! \sum_{k=0}^{m_{tcp}} \frac{a^k}{k!}}, \quad a = \frac{\lambda_1}{\mu_1} \tag{7}$$

$$E[N_{tcp}] = \frac{\lambda_1}{\mu_1} = \frac{a}{\mu_1(1-\pi_r)(1-P_{B1})} \tag{8}$$

4.3 HTTP 서브 시스템

HTTP 서브 시스템은 크게 HTTP 요청 대기큐와, 여러 개의 HTTP 쓰레드를 가진 HTTP 데몬으로 구성된다. HTTP 데몬은 트랜잭션 요청이 들어오기를 기다리고 있다가, 요청이 들어오면 그 요청을 처리할 HTTP 쓰레드를 할당하고, 그 쓰레드는 트랜잭션이 요청하고 있는 파일을 파일 시스템이나 캐쉬에서 패치(fetch)하여 메모리 버퍼에 복사해서 넣는다. 그리고 나서 그 쓰레드는 해제되어 다른 요청을 처리한다. 이때 만약 모든 메모리 버퍼가 사용중이라면, 한 메모리 버퍼가 빌 때까지 대기(idle) 상태로 기다리고, 만약 한 메모리 버퍼를 여러 쓰레드가 기다리고 있다면 해당하는 할당규칙에 의해 우선순위를 주어 처리한다. 이때 메모리 버퍼의 사용 가능여부는 메모리 버퍼에 있는 자료를 네트워크 서브 시스템에서 네트워크 출력 버퍼에 실을 수 있는지 여부에 의존하고, 네트워크 출력 버퍼에 있는 내용은 그것을 전송하고 ACK를 받을 때까지 출력 버퍼에 유지하고 있어야 하기 때문에 결국 네트워크의 정체 상태에 의존하게 되므로 랜덤한 요소로 볼 수 있다. 그리고, HTTP 데몬이 그 트랜잭션 요청을 처리할 쓰레드가 모두 사용중이어서 찾을 수 없을 때, 그 요청을 처리할 수 있는 쓰레드가 생길 때까지, HTTP 요청 대기큐에서 기다리도록 한다. 이때 만약 HTTP 요청 대기큐에 여분이 없어서 더 이상 들어오는 요청들을 수용할 수 없을 때 그 연결은 취소되고, 클라이언트는 거부 메시지를 받아서 그 시스템을 떠나거나 다시 TCP 요청 대기큐로 들어간다.

HTTP 서브 시스템에서 쓰레드의 개수를 m_{http}라 하고, HTTP 요청 대기큐의 사이즈를 Q라 하면 이 서브 시스템은 $M/M/m_{http}/(m_{http}+Q)$ 시스템으로 해석할 수 있다. 그리고 이 서브 시스템에서의 서비스 시간은 요청한 파일을 패치하는 시간과 메모리 버퍼에 복사하는 시간, 또는 만약 메모리 버퍼가 모두 사용중이라면 사용 가능한 버퍼가 생길 때

까지 기다리는 시간으로 이루어진다. 따라서 HTTP 서브 시스템에서의 서비스 요청율 λ_2,는 식 (3)~식 (5)로부터 유도할 수 있고, 서비스 처리율 μ_2, HTTP 요청 대기큐가 꽉 차서 요청이 거부될 확 P_{B2}, HTTP 쓰레드를 얻기 위해 기다리는 평균 대기 트랜젝션수 $E[N_{Q\_http}]$, HTTP 서브 시스템에 있는 총 평균 트랜젝션 요청수 $E[N_{http}]$, 그리고 HTTP 서브 시스템에서의 총 시스템 시간 T_{http}는 다음과 같이 구할 수 있다.

$$\lambda_2 = \frac{\alpha}{(1-\pi_r)\{1-(1-P_{time})(1-P_{B2})\}}, \quad \mu_2 = \frac{1}{T_{fetch}+T_{wait\ I/O}+T_{write\ I/O}} \tag{9}$$

$$P_{B2} = \frac{a^{m_{http}}}{m_{http}!}\left(\sum_{k=0}^{m_{http}-1}\frac{a^k}{k!}+\frac{a^{m_{http}}(1-\rho^{Q+1})}{m_{http}!(1-\rho)}\right)^{-1}, \quad a=\frac{\lambda_2}{\mu_2},\ \rho=\frac{\lambda_2}{m_{http}\mu_2} \tag{10}$$

$$E[N_{Q\_http}] = \sum_{k=m_{http}}^{m_{http}+Q}(k-m_{http})\rho^{k-m_{http}}P_{B2} = \frac{\rho^{Q+1}\{1-Q(\rho-1)\}}{(1-\rho)^2}P_{B2} \tag{11}$$

$$E[N_{http}] = E[N_{Q\_http}]+\frac{\lambda_2}{\mu_2} = \frac{\rho^{Q+1}\{1-Q(\rho-1)\}}{(1-\rho)^2}P_{B2}+\frac{\lambda_2}{\mu_2} \tag{12}$$

$$T_{http} = \frac{E[N_{q\_http}]}{\lambda_2}+\frac{1}{\mu_2} = \frac{\rho P_{B2}[1-\rho^Q(1+Q(1-\rho))]}{(1-\rho)^2} \tag{13}$$

4.4 네트워크 서브 시스템

네트워크 서브 시스템에서, 메모리 버퍼에 있는 파일들은 네트워크를 통해 전송할 수 있는 크기인 1개의 MSS 블록(Maximum Segement Size)으로 나누어진다. 이 과정은 서버가 메모리 버퍼에 접근하여, 버퍼에 저장되어 있는 파일을 가져와서 MSS 블록사이즈로 나눈 뒤, 해당 블록을 출력버퍼에 놓고 네트워크를 통해 그 블록을 보내면, 클라이언트가 네트워크 카드를 통해 해당 블록을 읽고 ACK를 보내주는 과정으로 구성된다. 이때 생기는 블록의 수는 파일의 크기를 출력버퍼의 크기로 나눈 값을 정수로 올린 값이고, 출력 버퍼의 내용은 연결 지향적인 TCP의 특성으로 인해 ACK를 받기 전까지는 출력 버퍼에 유지하고 있어야 하므로, 앞의 HTTP 서브 시스템에서 사용 가능한 메모리 버퍼를 얻기 위해 기다리는 시간 $T_{wait\ I/O}$은 네트워크의 정체 상태에 의존하고 있음을 알 수 있다.

따라서, 네트워크 서브 시스템에서의 서비스 처리시간은 메모리에 올라와 있는 파일들을 MSS 블록으로 나누는 시간($T_{partition}$)과 각각의 블록들을 출력버퍼에 놓는 시간(T_{buf}), 네트워크를 통해 서버 시스템에서 클라이언트 시스템으로 전송하는 시간($T_{transfer}$), 그리고 클라이언트가 네트워크 카드를 통해 해당 블록을 읽는 시간(T_{client})과 클라이언트로부터 ACK가 오는 시간(T_{ACK}) 값을 더한 값이 된다. 그리고 k개의 블록으로 나누어 졌다고 할 때, MSS 블록으로 나누는 시간외에 다른 시간들은 k를 곱한 만큼의 시간이 되므로, 네트워크 서브 시스템에서의 서비스 요청율 λ_3은 식 (3)~식 (5)로부터 유도할 수 있고, 서비스 처리율 μ_3은 다음과 같이 나타낼 수 있다.

$$\lambda_3 = \frac{(1-P_{B2})\alpha}{(1-\pi_r)(1-(1-P_{time})(1-P_{B2}))} \tag{14}$$

$$\mu_3 = \frac{1}{T_{partition}+k(T_{buf}+T_{transfer}+T_{client}+T_{Ack})} \tag{15}$$

여기에서 파일의 크기를 F, 출력 버퍼의 크기를 B_{net} 라 할 때 메모리 버퍼에 저장된 파일을 MSS 블록으로 나누어 생기는 블록의 개수 k는 다음과 같이 나타낼 수 있다.

$$k = \left\lceil \frac{F}{B_{net}} \right\rceil \tag{16}$$

네트워크 컨트롤러는 메모리 버퍼에 있는 정보를 출력버퍼로 보낼 때, 사용 가능한 출력버퍼가 없을 때 해당하는 요청을 출력 버퍼가 생길 때까지 무한대로 기다리게 할 수 있으며, 네트워크 컨트롤러에 의해 어느 한 순간에 출력 버퍼 m_{net}개 중 단 한 개만을 네트워크에 내보낼 수 있고, 출력 버퍼의 개수만큼 유지하고 있을 수 있으므로, 결국 1개의 서버를 가지고 m_{net}의 버퍼를 가진 시스템으로 해석이 가능하다. 따라서 네트워크 서브 시스템은 $M/M/1/m_{net}$으로 해석할 수 있다. 따라서 정상 상태에서 네트워크 서브 시스템에 있는 트랜젝션의 수가 k개일 확률은 식 (17)과 같이 나타낼 수 있다.

$$P[N=k] = \frac{(1-\rho)\rho^k}{1-\rho^{m_{net+1}}} \tag{17}$$

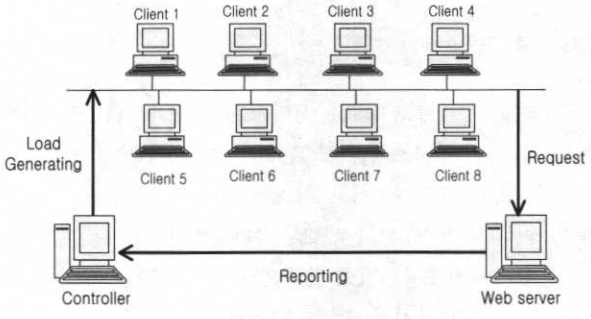
그리고 트랜젝션이 HTTP 서브 시스템을 지나 네트워크 서브 시스템에 왔을 때, 사용 가능한 출력 버퍼가 모두 사용 중 이어서 기다리는 트랜젝션까지 고려하면 네트워크 서브 시스템 내에 있는 총 트랜젝션의 수 $E[N_{net}]$와 서브 네트워크 시스템에서의 총 서비스 시간 T_{net}은 다음과 같이 구할 수 있다.

$$E[N_{net}] = \sum_{k=0}^{m_{net}} kP[N(t)=k] = \frac{\rho}{(1-\rho)} - \frac{(m_{net}+1)\rho^{m_{net+1}}}{1-\rho^{m_{net+1}}},\ \rho=\frac{\lambda_3}{\mu_3} \tag{18}$$

$$T_{net} = \frac{E[N_{net}]}{\lambda_3} = \frac{\rho}{(1-\rho)\lambda_3} - \frac{(m_{net}+1)\rho^{m_{net}+1}}{(1-\rho^{m_{net}+1})\lambda_3} \quad (19)$$

5. 검증 및 시뮬레이션

본문에서 제시한 웹서버 모델을 검증하기 위하여, 이산사건 시뮬레이션을 통한 성능과, 실험실 내의 LAN 환경에서 벤치마킹을 통해 웹서버의 성능을 측정한 값을 비교하였다. 시뮬레이션에 입력되는 모든 변수들은 가능한 실제 웹서버에서 사용하고 있는 값들과 시뮬레이션에 사용된 값에 근거하여 실제 값과 근사하게 설정하였다. 먼저, 실험실 내의 서버 구성은 (그림 3)과 같이, 부하를 발생시키고 각 클라이언트 컴퓨터들의 요청들에 대한 웹서버의 상태를 기록하는 1대의 컨트롤러 컴퓨터와 1대의 웹서버, 그리고 8대의 컴퓨터를 고속 이더넷(100Mbps)을 통해 연결하여, 한대의 컨트롤러 컴퓨터가 여러대의 클라이언트 컴퓨터들이 웹서버에 서비스 요청을 하게 함으로써 그에 따른 서버의 대역폭을 측정하는 WEB BENCH 4.1[21]을 통하여 수행하였다.



(그림 3) 테스트 서버 구성 환경

컨트롤러 컴퓨터는 클라이언트 서버로 하여금 웹서버에 저장되어 있는 파일들을 요청하게 함으로써 부하를 발생시키고, 그 요청하는 클라이언트 컴퓨터의 수를 조정함으로써 요청수를 변화시킬 때마다 웹서버에서 내보내는 대역폭을 측정한다. 그리고 웹서버는 펜티엄-II 300MHz CPU와 128M RAM, 10Mb/s 랜카드로 구성되고, OS는 Linux 커널 2.4.2이며 아파치 버전 1.3.19 웹서버 프로그램을 구동하였다. 웹서버에 적용된 파라미터들은 <표 1>과 같다.

〈표 1〉 웹서버에 적용된 파라미터

파라미터	m_{tcp}	m_{http}	Q	T_{feich}	m_{net}	B_{net}
값	64개	30개	10개	10 ms	60개	1024개

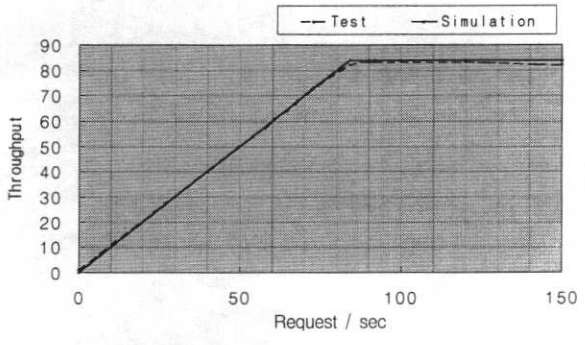
먼저 m_{tcp}의 값은 리눅스에서 usr/src/linux/net/TUNABLE 파일내의 backlog 큐 값을 64로 하여 설정하였고, μ_1의 값은 실험실내의 RTT 값을 측정하여 구하였다.

서비스 처리 시간에서 요청한 파일을 찾아 패치하는데 걸리는 시간 T_{fetch} 값은 5400RPM 하드디스크가 초당 약 100개의 랜덤 I/O 작업을 수행한다고 할 때, 디스크 엑세스 시간의 대부분이 탐색시간이므로 대략 10ms의 평균값을 가진다고 볼 수 있다[17]. 그리고, 파일의 크기가 7KByte 의 평균을 가진 지수함수 분포이고[13], 하드 디스크가 10MB/sec의 전송율을 가진다고 하면, 보통 한 파일에 대한 메모리 버퍼에 데이터를 쓰는 시간 $T_{write\ I/O}$ 는 0.7μs으로 생각할 수 있으므로 무시할 수 있다. 웹서버에서 제공하는 파일의 평균 크기가 7KByte로 가정하였기 때문에 한 개의 파일을 네트워크 출력 버퍼에 쓰는 시간 T_{buf} 는 거의 무시할 수 있을 정도의 작은 값이고, 또 파티션하는데 소요되는 시간 $T_{Partition}$ 또한 무시할 수 있을 정도로 작은 값이기 때문에 0으로 근사화 하였다.

네트워크 서브 시스템에서, 실험실내의 클라이언트가 웹서버로부터 평균 550Kb/s의 대역폭으로 파일을 전송 받음을 실험에 의해 구할 수 있다. 또, 파일 평균 크기가 7KByte의 평균을 가진 지수분포를 가진다고 할 때, 파일을 클라이언트에게 전송하는 시간은 평균 100ms의 지수 분포를 가진다고 할 수 있다. 그리고 클라이언트가 받은 패킷을 읽고, ACK를 보내는 시간 RTT값 역시 전송하는 시간에 비해 너무 작은 값이므로 무시 네트워크 서브 시스템에서의 총 서비스 처리시간은 100ms의 평균을 가진 지수함수 분포로 해석하여 시뮬레이션을 행하였다. 실제 웹서버의 성능을 가시적으로 나타내기 위해 다음과 같이 트렌젝션당 총 시스템 시간 T_{system}에 의해 웹서버의 성능을 정의하고 T_{http}와 T_{net}은 각각 식 (13)과 식 (19)에서 구할 수 있다.

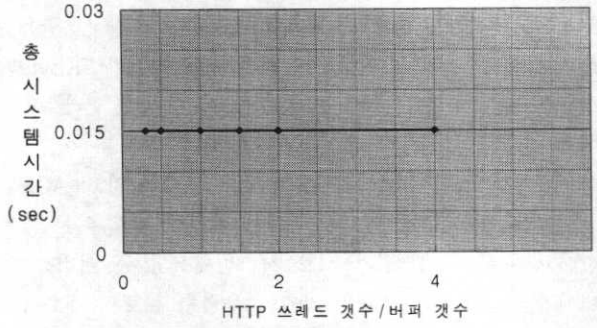
$$T_{system} = \frac{1}{\mu_1} + T_{http} + T_{net} \quad (20)$$

웹 벤치 4.1을 통해 얻은 웹서버의 성능과 시뮬레이션에 의한 웹서버의 성능 비교는 아래 그림과 같다.



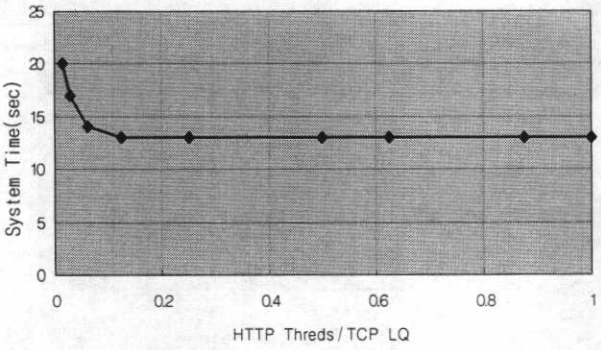
(그림 4) 실험값과 시뮬레이션 결과의 비교

시스템에서의 여러 변수들 중에서도 HTTP 1.1은 세션이 종료되기 전에 비활성 OFF 시간이 이미 설정된 타임 아웃 시간을 초과하여 세션이 종료되는 확률과, 세션이 종료되었을 때 그 페이지를 다시 서비스 받기 위해 재연결을 시도하는 확률에 강력하게 의존하고 있기 때문에 앞에서 제시한 웹서버 모델이 검증되기 위해서는 시스템 처리 상황에 따라 다양한 재시도 확률 π_r 값과 세션이 타임 아웃되는 확률 $P_{timeout}$ 값이 적용되어야만 했고 그 값은 웹 서버가 어떤 종류의 서비스를 하는가와 그에 따른 웹서버의 Persistent Connection 지속시간에 강력히 의존한다. 테스트 결과 우리는 $\pi_r = 0.15$ 과 $P_{timeout} = 0.3$ 인 값에서 WEB BENCH의 결과 값과 유사한 시뮬레이션 결과를 얻을 수 있었다. 앞에서 제시하고 있는 웹서버 모델은 아래 그림에서 볼 수 있는 것처럼 실험실에서 WEB BENCH를 통해 얻은 결과와 거의 유사함을 볼 수 있다.



(그림 5) HTTP 쓰레드 vs 버퍼

시스템 변수들의 관계에 따른 시스템 성능 변화를 알아보기 위해, TCP 요청 대기큐의 크기와 HTTP 쓰레드 개수 및 HTTP 대기큐의 크기를 변화시켜 보았다. 먼저 HTTP 쓰레드 개수와 HTTP 쓰레드가 모두 사용중일 때 기다리는 버퍼의 크기에 따른 시스템 시간값의 변화는 (그림 5)와 같이 $m_{http\,/\,Q}$ 값에 상관없이 일정함을 알 수 있다.



(그림 6) TCP 요청 대기큐 vs HTTP 쓰레드

그리고, TCP 요청 대기큐의 크기와 HTTP 쓰레드 개수의 비율 변화에 따른 시스템 시간값의 변화는 (그림 6)과 같다. HTTP 쓰레드의 수가 TCP 요청대기큐의 값보다 작을 때에는 큰 시스템 시간값을 가짐으로써 좋지 않은 성능을 보이고 그 값이 증가함에 따라 시스템 시간이 감소하여 성능이 개선되지만 그 비율이 0.1을 넘어서면 거의 일정한 시스템 시간 값을 볼 수 있다. 따라서 웹서버는 TCP 요청 대기큐의 사이즈의 10% 이상의 쓰레드 개수만 있으면 일정한 성능을 유지하고, 비록 쓰레드의 개수가 증가하여도 시스템 성능이 크게 변하지 않는다고 볼 수 있다. 이것은 네트워크 서브 시스템에 있는 출력 버퍼의 데이터가 연결 지향적인 TCP의 특성으로 인해 ACK를 받기 전까지는 출력 버퍼에 유지되어야 하므로 HTTP 서브 시스템에서 사용 가능한 메모리 버퍼를 얻기 위해 기다리는 시간 $T_{wait\ I/O}$ 은 네트워크의 정체 상태에 의존하고 있기 때문이다. 지속연결 종료확률은 타임 아웃값을 어떻게 설정하느냐에 의존하고, 연결이 종료되었을 때 재 시도할 확률은 서버가 어떤 종류의 서비스를 운영하고 있는 지에 의존하므로, 이 두 값을 찾은 뒤, 시뮬레이션의 결과를 이용하면 더 효율적인 웹서버의 운영이 가능하다.

6. 결 론

본 논문에서는 웹서버의 서비스를 TCP와 HTTP, 그리고 Network 서브 시스템으로 Tandem으로 연결된 큐잉 모델을 제시하고, 각각의 서브 시스템 내에서 일어나는 과정을 큐잉 이론을 통해 해석적으로 나타내었다. 이 모델은 실험실 LAN 환경에서 WEB BENCH 4.1을 통한 벤치마킹과 실험실 환경과 유사한 값에 기반하여 해석적으로 이산사건 시뮬레이션을 통하여 검증하였다. 또한 각 서브 시스템에서의 시스템 성능에 영향을 미치는 변수들을 변화시킴으로써, 최적의 웹 서비스를 제공하는 조건들을 확인할 수 있었다. 이 모델이 HTTP 1.1을 기반으로 모델링 되었기 때문에, 세션종료 전에 타임아웃 되어 연결이 종료되는 확률과, 타임아웃 되었을 때 재시도하는 확률은 아파치와 같은 웹서버 프로그램내의 HTTP 1.1 Persistent Connection 타임아웃 값 설정에 따라 큰 영향을 받기 때문에, 시스템 성능 평가에 앞서 그 웹서버의 설정에 따른 성능 분석이 요구된다.

일반적인 웹서버에 대해 제안한 웹서버의 모델을 보완하고 일반화하기 위하여 고려되어야 할 사항은 다음과 같다. 먼저 웹 트래픽이 Poisson 분포를 따르지 않으며, 웹서버에 요청하는 파일의 크기가 지수 분포가 아닌 Heavy-tailed 분포를 따른다는 점, 또한 각 서브 시스템 내에서의 서비스 시간이 지수분포가 아닌 일반분포를 가진다는 점과 웹서버

내의 캐시에 대한 영향에 따라 이 모델은 더 확장 될 여지가 있다.

참 고 문 헌

[1] Zhen Liu, N. Niclause, C. Jalpa-Villaneva, "Traffic model and Performance Evaluation of web servers," Performance Evaluation 46, pp.77-100, 2000.

[2] P. Barford, M.E. Crovella, "A Performance Evaluation of Hyper Text Transfer protocols," proceedings of the ACM Sigmetrics'99, 1999.

[3] B. A. Mah, "An empirical model of HTTP network traffic," Proceedings of the IEEE INFOCOM'97, 1997.

[4] P. Barford, M. E. Crovella, "Generating repr besentative web workloads for network and server performance evaluation," Proceedings of the ACM Sigmetrics '98, 1998.

[5] E. Casilari, F. J. Gonzalez, F. Sandoval, "Modeling of HTTP traffic," IEEE Communications Letters, Vol.5, No.6, June, 2001.

[6] R. D. Van der Mei, R. Hariharan, P. K. Reeser, "Web Server Performance Modeling," Proceedings of 4th Informs Telecom Conference, Special Issue of Telecommunication.

[7] P. K. Reeser, R. Hariharan, "Analytical Model of Web Servers in distributed Environment," Proceedings of the second international workshop on software and performance, September, 2000.

[8] J. Dilley, R. Friedrich, T. Jin, J. Rolia, "Web server performance measurement and modeling techniques," Performance Evaluation 33, pp.5-26, 1998.

[9] J. Heidenmann, K. Obraczka, J. Touch, "Modeling the Performance of HTTP Over Several Transport Protocols," IEEE transactions on networking, Vol.5, No.5, October, 1997.

[10] J. Mogul, "The Case for Persistent-Connection HTTP," Proceedings of ACM SIGCOMM '95, Cambridge, MA, pp. 299-313, 1995.

[11] M. Arlitt, "Characterizing Web User Sessions," Hewlett-Packard Laboratories Technical Report HPL-2000-43, May, 2000.

[12] Padmanabhan, Mogul, "improving HTTP latency," proceedings of the second WWW conference, 1994.

[13] M. Crovella, R. Frangioso, M.Harchol-Balte, "Connection scheduling in web servers," proceedings of the 1999 USENIX Symposium on internet Technologies and Systems(USITS '99), October, 1999.

[14] V. Almeida, A. Bestavros, M. Crovella, A. de Oliveira, "Characterizing reference locality in the WWW," proceedings of 1996 International Conference on Parallel and Distributed Information Systems, pp.92-103, December, 1996.

[15] Douglas E. Comer, "Internetworking with TCP/IP vol I : Principle, and Architecture 3rd edition," 1995, Prentice-Hall, Inc.

[16] Alberto Leon-Garcia, "Probability and Random Processes for Electrical Engineering," second edition, Addison Wesley.

[17] Patreick Killelea, "Web Performance Tuning," O'Reilly, 1998.

[18] http://httpd.apache.org/docs.

[19] Allen B. Downey, "The Structual cause of file size distributions," MASCOTS'01.

[20] M. F. Arlitt, C. L. Williamson, "Web Server Workload Characterization : The Search for Invariants," ACM SIGMETRICS '96, pp.126-137, May, 1996.

[21] http://www.etestinglabs.com.

민 병 석

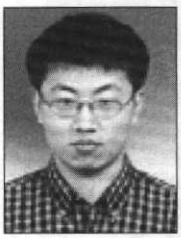
e-mail : carlos@yonsei.ac.kr
2001년 연세대학교 전기공학과(학사)
2001년~현재 연세대학교 대학원 전기전자 공학과 석사과정
관심분야 : 인터넷 QoS, 웹서버 모델링

남 의 석

e-mail : nahmes@yonsei.ac.kr
1991년 연세대학교 전기공학과(공학사)
1993년 연세대학교 전기공학과(공학석사)
1998년 연세대학교 전기공학과 대학원 (공학박사)
1996년~2002년 LG산전 시스템사업부 과장
2002년~현재 연세대학교 전기전자공학과 BK21 연구교수
관심분야 : 시스템 모델링, 지능형 제어, 수요예측, 서버 시스템

이 상 문

e-mail : sangmoon@yonsei.ac.kr
1997년 영남대학교 전기공학과(학사)
1999년 연세대학교 대학원 전기컴퓨터 공학과(공학석사)
1999년~현재 연세대학교 대학원 전기전자공학과 박사과정
관심분야 : 실시간 시스템, 고장포용 시스템, 인터넷 서버



심 영 석

e-mail : youngseok@yonsei.ac.kr
2001년 연세대학교 전기공학과(학사)
2001년~현재 연세대학교 대학원
전기전자공학과 석사과정
관심분야 : 인터넷 QoS, Embedded system

김 학 배

e-mail : hbkim@yonsei.ac.kr
1988년 서울대학교 전자공학과(학사)
1990년 미국 미시간대학교 전기 및 컴퓨터
공학과(EECS)(공학석사)
1994년 미국 미시간대학교 전기 및 컴퓨터
공학과(EECS)(공학박사)
1994년~1996년 미국 National Research Council(NRC) Research
Associate at NASA Langley Research Center
1996년~현재 연세대학교 전기전자공학과 부교수
관심분야 : 실시간 시스템, 인터넷 웹서버 기술, 디지털 시스템
고장포용 및 신뢰도 평가분야