웹 환경에서의 평균 대기 시간 및 평균 반환 시간의 분석

이 용 진†

요 약

HTTP(HyperText Transfer Protocol)는 월드 와이드 웹 분산 시스템이 객체를 인출하기 위해 사용하는 전송 프로토콜이다. HTTP는 연결 지향 프로토콜이므로 트랜스포트 계층에서 TCP(Transmission Control Protocol)를 사용한다. 그러나 HTTP는 TCP와의 상호 운용이 좋은 편이 아닌 것으로 알려져 있다. 본 연구에서는 TCP의 성능에 영향을 주는 요인들을 살펴보고, HTTP 접근을 위해 TCP의 slow-start 오버헤드 및 연결에 소요되는 트랜잭션 시간과 TCP의 성능 향상 대안중의 하나인 T-TCP(Transaction TCP)의 트랜잭션 시간을 검토한다. 평균 대기 시간과 평균 반환 시간은 사용자의 서비스 품질을 만족시키기 위한 중요한 파라메터들이다. TCP와 T-TCP 트랜잭션 시간이 주어지는 경우 그러한 파라메터들의 계산 공식이 유도되었다. 실험 및 계산 경험을 통해 제안된 공식이 잘 작동됨을 확인하였고, 대역폭의 확장이 필요한 환경에 적용될 수 있으며 T-TCP의 시간 특성이 TCP 보다 우수함을 확인하였다. 아울러, 평균 대기 시간과 평균 반환 시간을 줄이기 위해 대역폭을 조합하여 서버를 분산하는 방법이 제시되었다.

Analysis of Average Waiting Time and Average Turnaround Time in Web Environment

Yong-Jin Lee†

ABSTRACT

HTTP (HyperText Transfer Protocol) is a transfer protocol used by the World Wide Web distributed hypermedia system to retrieve the objects. Because HTTP is a connection oriented protocol, it uses TCP (Transmission Control Protocol) as a transport layer. But it is known that HTTP interacts with TCP badly. It is discussed about factors affecting the performance of HTTP over TCP, the transaction time obtained by the per-transaction TCP connections for HTTP access and the TCP slow-start overheads, and the transaction time for T-TCP (Transaction TCP) which is one of methods improving the performance of HTTP over TCP. Average waiting time and average turnaround time are important parameters to satisfy QoS (Quality of Service) of end users. Formulas for calculating two parameters are derived. Such formulas can be used for the environment in which each TCP or T-TCP transaction time is same or different. Some experiments and computational experiences indicate that the proposed formulas are well acted, can be applied to the environment which the extension of bandwidth is necessary, and time characteristics of T-TCP are superior to that of TCP. Also, the load distribution method of web server based on the combination of bandwidths is discussed to reduce average waiting time and average turnaround time.

키워드 : HTTP, TCP, T-TCP, 평균 대기 시간(Average Waiting Time), 평균 반환 시간(Average Turnaround Time)

1. 서 론

인터넷은 응용 계층에서 HTTP[1]를 기본 프로토콜로 사용하는 네트워크로 트랜스포트 계층은 TCP[10]를 사용하여 구현된다. 그런데, TCP는 몇 가지의 성능 문제를 가지고 있는 것으로 알려져 있으므로 인터넷 역시 TCP의 성능상의 문제를 그대로 갖게 된다[16, 20].

첫 번째 문제는 연결 설정(connection setup) 문제로 TCP는 3-way-hand-shake를 사용하여 연결을 설정하므로 전체 트랜잭션 시간은 최소한 2×RTT(Round Trip Time)가 되어야 한다.

두 번째 문제는 데이터 세그먼트 문제로, TCP의 최대 세그먼트 크기(MSS : Maximum Segment Size)는 원격 연결인 경우 기본 값(default)은 536바이트이지만 대부분의 실제 구현에서는 주로 512바이트가 사용되기 때문에 커다란 파일이 전송되는 HTTP 환경에서는 여러개의 세그먼트 전송이 요구된다.

세 번째 문제는 윈도우 및 slow start 문제로, 수신자는 각 세그먼트 마다 송신자에게 액노리지먼트 없이도 받을 수 있는 최대 데이터 크기인 윈도우 크기를 알려주지만 네트워크의 실제 전달률을 알 수가 없으므로 TCP는 slow start라는 프로세스를 사용하여 최대 이용률을 결정한다. 따라서 HTTP는 slow start 때문에 TCP 연결이 생성되고 나서도 몇 개의 RT(Round Trip) 동안에는 네트워크의 대역폭을 충

† 정 회 원 : 우송대학교 컴퓨터전자정보공학부 교수
논문접수 : 2002년 4월 18일, 심사완료 : 2002년 9월 2일

분히 이용하지 못한다.

네 번째 문제는 지연 시간과 대역폭 문제인데 RTT로 측정되는 지연 시간은 트랜잭션에 대한 고정 비용으로 파일의 크기와는 상관이 없으며 대역폭은 데이터를 전송하는데 있어서 걸리는 시간과 관계되는 척도이다.

다섯 번째 문제는 트랜잭션마다 하나의 연결을 설정하는 문제로 HTTP는 하나의 요구 내에서 여러개의 객체(multiple objects)를 요청하는 방법이 없기때문에 각 인출은 하나의 트랜잭션을 요구한다. 이것은 HTTP의 낮은 버전을 사용하는 경우 매번 새로운 연결을 설정해야 함을 의미한다. 설사 연결이 설정된채 유지가 가능한 프로토콜을 사용한다고 하더라도 각 요구/응답은 분리된 RT 지연을 발생시킨다.

마지막 문제는 하나의 요구에 대해 하나의 연결을 설정할 때 생기는 TIME_WAIT 문제[4]로, 서버가 TCP 연결을 종료한 이후에도 일정 시간 동안 그 연결에 대한 정보를 유지해야만 한다.

이러한 문제들을 프로토콜 측면에서 해결하는 방법으로는 T-TCP, Persistent HTTP, Shared TCP Control Blocks 등이 있다. T-TCP는 3-way-hand-shake를 생략하고 TIME_WAIT 상태를 240초에서 12초 정도로 단축시킴으로써 문제를 해결한다. TCP의 클라이언트-서버 과정에서 트랜잭션 시간은 적어도 2 × RTT + 파일 전송 시간인데 비해 T-TCP에서는 이 시간이 RTT + 파일 전송 시간으로 대폭 감소된다[2, 3]. Persistent HTTP[11]는 HTTP/1.1[5], HTTP-NG[15], S-HTTP[14] 등의 프로토콜을 포함하는 것으로 연이어오는 트랜잭션 요구는 새로운 연결 설정을 하지 않으며 각 새로운 트랜잭션에 대해 slow-start를 피한다. 한편, Shared TCP Control Blocks(S-TCB)[19]는 TCP에 대한 HTTP의 slow start 요소를 최적화 한다.

기존의 연구들[6-9, 12, 13]이 TCP 상에서 연동되는 HTTP의 성능 문제를 검토하고, 윈도우 크기 및 slow start를 고려하면서 각 대역폭에 따른 트랜잭션 시간등을 계산하였다. 그러나 이러한 여러 가지 성능 분석에 있어서 기존의 연구들은 사용자 측면에서 실제로 느끼는 대기 시간이나 반환 시간등에 대한 성능 평가 척도를 고려하지 않았다. 즉, 사용자가 실제로 느끼는 평균 대기 시간이나 평균 반환 시간 등을 원하는 값으로 제한하는 경우에 필요한 것은 얼마나 네트워크를 확장해야 하는 웹 서버는 어떻게 부하를 분산해야 하는지에 대한 계량적인 척도이다. 따라서 본 연구는 이 중에서 네트워크와 관련된 프로토콜(TCP)과 사용자 수에 따른 성능 문제에 초점을 맞춘 것으로 종단 사용자의 QoS로 평균 지연 시간과 평균 반환 시간을 가정했을때 실제 환경에 대한 이론적인 상한 값(upper bound)을 제공하는데 목적이 있다. 이를 위해 트랜잭션 시간이 서로 동일하거나 상이한 웹 환경 하에서 이러한 성능 평가 척도들을 유도하고 실제 실험 및 계산 경험을 다룬다.

전체 4장으로 구성되는 본 연구는 2장에서 TCP와 T-TCP의 트랜잭션 시간을 검토한 후 사용자 수에 따른 평균 대기 시간과 평균 반환 시간을 유도한다. 3장에서는 실제 실험 및 계산 경험을 통한 성능 평가를 다루고 마지막으로 4장에서 결론을 맺는다

2. 평균 대기 시간 및 반환 시간의 검토

이 장에서는 사용자들이 느끼는 네트워크 성능 평가 요소들 중에서 가장 중요한 요소들인 평균 대기 시간 및 평균 반환 시간을 유도하고자 한다. 먼저 기존 논문[18]에서는 TCP 위에서 연동되는 HTTP 프로토콜의 비효율성을 측정한 바 있다. 즉, HTTP의 상호 동작 시간을 분석하고, 트랙잭션당 연결 설정, 잠재적인 slow-start의 추가부담에 대한 상한 값을 계산하였다. 이렇게 얻어진 값을 최적 전송 시간과 비교하였다. 또한 시스템의 최적 성능을 고려하여 호스트는 단지 네트워크의 대역폭에 의해서만 제한되고 서버의 처리 시간은 무시하는 것으로 가정하였고 디스크 입/출력과 기타의 병목도 최소라고 가정하였다. 그러나 사용자 입장에서 중요한 성능 평가 요소인 평균 대기 시간과 평균 반환 시간은 고려하지 않았다. 우리가 이러한 시간들을 알게 되면, 즉 사용자가 원하는 최소 평균 대기 시간 또는 평균 반환 시간을 알게 되면, 그에 맞는 최소 대역폭의 크기 등을 얻을 수 있게 되므로 이를 통해 대역폭 확장 및 웹 서버의 성능 향상에 대한 구체적인 대안을 얻을 수 있을 것이다. 먼저 [18]의 용어들을 확장하여 다음의 기호들을 정의한다.

2.1 용어의 정의

R : RTT
bw : 대역폭(bandwidth)
MSS : 최대 세그먼트 크기(maximum segment size : packet size)
K : 파일내의 패킷 수(number of packets in the file)
L : RTT 동안 처리되는 패킷의 수(number of packets in RTT)
M : 최대 이용가능 윈도우 크기(max useful window size : lower bound)
S : TCP의 slow-start로 인한 round trip(upper bound)
W : 낭비 시간 (amount of wasted time)
T_f : 최소 파일 전송 시간(minimum file transmission time)
T_{min} : 최소 트랜잭션 시간(minimum transaction time)
T : TCP의 트랜잭션 시간(transaction time for TCP)
$T_{T\text{-}TCP}$: T-TCP의 트랜잭션 시간(transaction time for T-TCP)
AW : 평균 대기 시간(Average Waiting Time)
AT : 평균 반환 시간(Average Turnaround Time)
SW : 대기 시간의 표준편차(Standard deviation of Waiting time)
ST : 반환 시간의 표준 편차(Standard deviation of Turnaround time)

2.1.1 패킷 트랜잭션 시간이 전부 동일한 경우

m : 전체 사용자 수

n : 사용자의 요구 패킷 수(K의 ceiling)

t : 패킷 트랜잭션 시간(T/n 또는 $T_{T\text{-}TCP}/n$)

2.1.2 패킷 트랜잭션 시간이 서로 다른 경우

P : 전체 대역폭 라인의 수

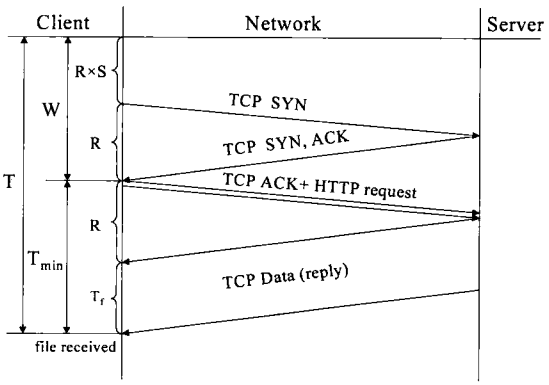
m_i : i 번째 대역폭 라인의 사용자 수 (i = 1, 2, ⋯, P)

m : 전체 사용자 수 ($m = \sum_{i=1}^{P} m_i$)

t_i : i 번째 대역폭 라인을 사용하는 패킷 트랜잭션 시간

2.2 용어의 설명

HTTP는 전체 트랜잭션에 대해 하나의 TCP 연결을 사용하며 서버 또는 클라이언트에서 FTP 프로토콜이 요구하는 것과 같은 제어-채널을 요구하지 않는다. HTTP에서 파일 전송과 관련된 흐름은 (그림 1)과 같다.



(그림 1) HTTP over TCP의 파일 전송

HTTP가 TCP와 연동하는데 있어서 발생되는 문제점은 이미 앞에서 살펴본 바가 있다. (그림 1)에서 보듯이 TCP는 임의의 HTTP request를 전송하기전에 연결을 설정해야 하고 동시에 slow-start와 같은 혼잡처리(congestion management) 메커니즘을 작동시킨다. 이제 2.1절에서 정의한 용어를 살펴보면 R, bw, MSS 등은 자명하다. K는 파일내의 패킷 수이므로 식 (1)과 같다.

$$K = \frac{filesize}{MSS} \tag{1}$$

L은 라운드 트립 타임 동안 파이프를 채우는 패킷의 수이므로 식 (2)와 같다.

$$L = \frac{bw \times R}{MSS} \tag{2}$$

(그림 1)에서, 이상적인 최적 전송(optimal transmission)을 가정해 보면 이는 TCP의 문제점들인 연결 설정 과정과 혼잡 처리과정을 제외한 것이 될 것이다. 즉, 그림에서 보듯이 HTTP request를 요청하고 대역폭(bw)에 따른 최고 전송속도로 파일을 수신하는 것이다. slow-start가 포함되지 않은 최소 파일 전송 시간(T_f)은 파일 크기를 대역폭으로 나누면 되므로 식 (3)이 된다.

$$T_f = \frac{filesize}{bw} \tag{3}$$

그러나, 실제 상황에서 TCP는 slow-start 메커니즘을 사용한다. 이 메커니즘은 초기 윈도우를 제한하는 기법으로 첫 번째 ACK를 수신할 때까지는 단 하나의 패킷만을 전송할 수 있고, 뒤이어 받게 되는 ACK에 대해서는 2의 지수로 전송 패킷 개수를 증가시킬 수 있다. 따라서 그림에서 TCP Data(reply)를 전송할때 네트워크의 대역폭이 충분하더라도 윈도우가 충분히 커질때까지는 전체 대역폭을 사용하지 못하게 되므로 실제적인 파일 전송 시간은 slow-start에 의한 영향만큼 증가된다.

M은 주어진 파일에 대한 최대 이용가능 윈도우 크기로 RTT 동안 파이프를 채우는 패킷의 수와 파일 내에 포함되는 패킷의 수중에서 작은 값을 취한다. 즉, 임의의 시간에 전송 가능한 최대 패킷 수가 된다.

$$M = \min(L, K) \tag{4}$$

S는 초기 slow-start 동안에 발생하는 라운드 트립의 개수이다. 초기 송신 윈도우는 1 MSS에서 시작한다. 그러나 대부분의 BSD 구현에서는 TCP SYN이 긍정 응답(ACK)되면 2로 증가된다. 데이터 트랜스포트를 위한 윈도우는 2에서 시작하고 각 라운드 트립에서 2배씩 증가하며 매 번의 전송량은 M보다 작다.

$$S = floor(\log_2(ceil(M/2))) \tag{5}$$

slow-start에 의한 시간의 량은 R×S로 주어진다[18]. (그림 1)에서 실제 파일 전송 시간은 최소 파일 전송 시간(T_f) + slow-start로 인한 추가 지연(R×S)이다. 따라서 R×S는 TCP Data(reply)에 포함하여 그려야 되지만 뒤에서 설명할 T_{min}과의 혼동을 피하기 위하여 그림의 시작 부분에 표시한 것에 주의하기 바란다.

W는 낭비되는 시간으로, 연결 설정을 위한 하나의 라운드 트립 타임과 slow-start를 위한 기껏해야 하나의 라운드 트립의 합이다.

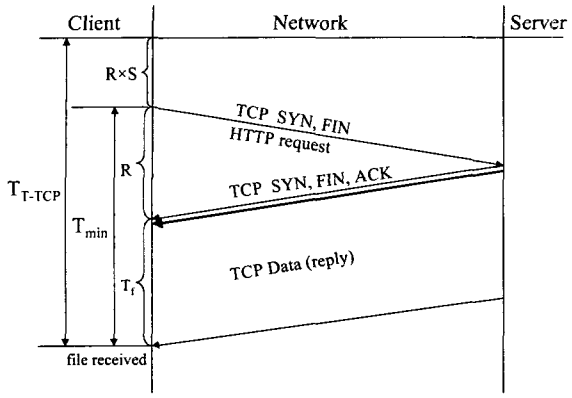
$$W = slow\ start\ time + connection\ setup\ time = R \times S + R \tag{6}$$

이상적인 최소 트랜잭션 시간(T_{min})은 HTTP request가 Server에 도달한 후 파일의 첫 번째 바이트가 Client에 도달하는데 걸리는 RTT에 최소 파일 전송 시간을 더하면 되므로 $T_{min} = R + T_f$가 된다. 따라서 TCP의 트랜잭션 시간은 (그림 1)에서 보듯이

$$T = T_{min} + W = 2R + T_f + R \times S \tag{7}$$

가 된다. 다음으로 T-TCP의 경우는 TCP에서 연결 설정 시

간(R)을 생략한 것이므로 (그림 2)처럼 표현된다.



(그림 2) HTTP over T-TCP의 파일 전송

따라서

$$T_{T-TCP} = R + T_f + R \times S \quad (8)$$

가 된다. (그림 2)에서 R×S의 위치가 앞에 나온 이유는 (그림 1)에서 설명한 이유와 같다.

2.3 가정 및 기본 개념

2.3.1 가정

(1) 웹 서버에서의 처리 방법은 패킷별로 라운드 로빈 기법을 사용한다.

(2) 사용자별로 패킷 수는 동일하다(즉, 동일한 크기의 파일을 요청한다고 가정한다).

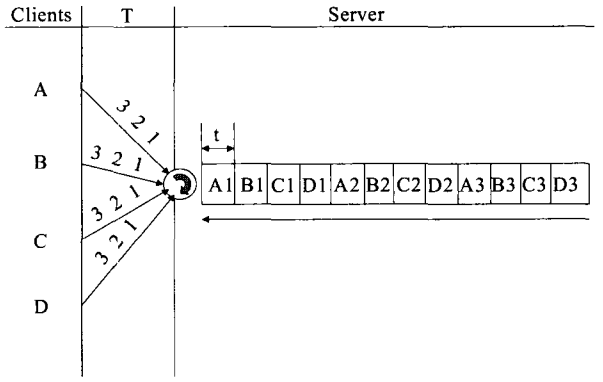
(3) 서버 처리 시간 및 요구 전송 시간은 무시한다.

2.3.2 기본 개념

앞에서 구한 전체 트랜잭션 시간(T)은 한명의 사용자가 웹 서버에 접속하여 파일을 가져올때 걸리는 전체 시간이다. 이 시간에는 임의의 개수(n) 만큼 패킷이 포함되어 있다. 가정에 의해 패킷별로 라운드 로빈 기법을 사용하므로 패킷 트랜잭션 시간은 값이 동일한 경우 t = T / n이 되고, n 값은 [filesize / MSS]가 됨은 자명하다. 이제, 4명(m)의 Client가 웹 서버에 모두 동일한 크기의 파일을 요청한다고 하면 각 Client가 기대하는 트랜잭션 시간(T)은 모두 같을 것이다. 이제, 패킷 수(n)가 3이고, 이를 패킷별로 라운드-로빈 방식으로 처리하는 경우의 예를 들면 (그림 3)과 같다. 그림에서 Client에서 Server로의 요청은 실제로는 파일을 요청하는 것이지만, 그 파일에는 패킷이 3개가 포함되므로 1, 2, 3으로 표시하였고, T는 각 Client가 기대하는 전체 트랜잭션 시간이다.

그림에서 보듯이, Server에서 Client로의 전송 순서는 A의 첫 번째 패킷(A1), B의 첫 번째 패킷(B1)… 순이 된다. 우리가 관심이 있는것은 그림의 간트 차트에 의해 계산될 수 있는 사용자들의 평균 대기 시간 및 평균 반환 시간 등이다. 그림에서 A의 대기시간은 (B1 + C1 + D1) + (B2 + C2 + D2)이고, 반환시간은 A3의 종료시점임을 알수 있다. B, C, D에 대해서도 같은 방법이 적용될 수 있다. 따라서, 평균 대기 또는 반환 시간은 사용자들의 대기 또는 반환 시간을 전체 사용자 수로 나누면 얻어질 것이다. 각 Client는 수신 완료 시점을 이론적인 트랜잭션 시간(T)으로 기대하지만 실제로 Client들에게 전송이 완료되는 시간은 서버에 접속되는 사용자의 수에 영향을 받게 될 것이다.

2.4절에서는 사용자들의 패킷 트랜잭션 시간(t)이 모두 동일한 경우의 평균 대기 시간 식 (9), 평균 반환 시간 식 (10)이 유도되고, 2.5절에서는 사용자 단위로 패킷 트랜잭션 시간이 서로 다른 경우의 평균 대기 시간 식 (11), 평균 반환 시간 식 (12)이 유도된다.



(그림 3) 사용자 수(m)가 4이고 패킷 수(n)가 3인 경우의 예

2.4 패킷 트랜잭션 시간이 전부 동일한 경우

이 경우는 내부망 서비스만 수행하거나 또는 외부망이 하나인 경우에 적용 가능하다.

[Lemma 1] 패킷 트랜잭션 시간이 전부 동일한 경우의 평균 대기 시간은

$$AW = \frac{\sum_{i=1}^{m}(m-i)t + m(n-1)(m-1)t}{m} \text{ 이다.} \quad (9)$$

<증명> 사용자 수(m), 패킷 수(n), 패킷 트랜잭션 시간(t)이 주어지는 경우, 각 사용자에 대한 대기 시간은 다음과 같다. (그림 3)에서 보듯이, 첫 번째 사용자의 대기 시간 = 0 + (사용자수 - 1) × t + (사용자 수 - 1) × t = (패킷 수 - 1) × [(사용자 수 - 1) × t], 두 번째 사용자의 대기 시간 = t + (사용자 수 - 1) × t + (사용자 수 - 1) × t = t + (패킷 수 - 1) × [(사용자 수 - 1) × t], 세 번째 사용자의 대기 시간 = 2t + (사용자 수 - 1) × t + (사용자 수 - 1) × t = 2 t + (패킷 수 - 1) × [(사용자 수 - 1) × t]이다. 이런 식으로 확장하면 m 번째 사용자의 대기 시간 = (m - 1) t + (사용자 수 - 1) × t + (사용자 수 - 1) × t = (m - 1) t + (패킷 수 - 1) × [(사용자 수 - 1) × t] 가 된다. 따라서, 이를 일반화하여 다 더하면 전체 대기 시간은

$$\sum_{i=1}^{m}(m-i)t+m(n-1)(m-1)t$$

이므로 평균 대기 시간은

$$\frac{\sum_{i=1}^{m}(m-i)t+m(n-1)(m-1)t}{m}$$ 이다.

[Lemma 2] 패킷 트랜잭션 시간이 전부 동일한 경우의 평균 반환 시간은

$$AT=[\frac{2mn-m+1}{2}]t$$ 이다. (10)

(증명) 사용자 수(m), 패킷 수(n), 패킷 트랜잭션 시간(t)이 주어지는 경우, 각 사용자에 대한 반환 시간은 다음과 같다. (그림 3)에서 첫 번째 사용자의 반환 시간 = 사용자 수 × (패킷 수 - 1) × t + t, 두 번째 사용자의 대기 시간 = 사용자 수 × (패킷 수 - 1) × t + 2t, 세 번째 사용자의 대기 시간 = 사용자 수 × (패킷 수 - 1) × t + 3t 이다. 이런 식으로 확장하면 m번째 사용자의 대기 시간 = 사용자 수 × (패킷 수 - 1) × t + mt 가 된다. 따라서, 이를 일반화하여 다 더하면 전체 반환 시간은

$$m[m(n-1)t]+\frac{m(m+1)t}{2}$$
$$=mt[m(n-1)+\frac{m+1}{2}]$$

이므로, 평균 반환 시간은

$$AT=[m(n-1)+\frac{(m+1)}{2}]t=[\frac{2mn-m+1}{2}]t$$ 이다.

[예제 1] (그림 3)에서 m = 4, n = 3, t = 2 인 경우의 평균 대기 시간과 평균 반환 시간을 구하면 다음과 같다.

$$AW=\frac{\sum_{i=1}^{m}(m-i)t+m(n-1)(m-1)t}{m}$$
$$=\frac{[(4-1)2+(4-2)2+(4-3)2+(4-4)2]+4(3-1)(4-1)2}{4}=15$$
$$AT=[\frac{2mn-m+1}{2}]t=[\frac{2\times4\times3-4+1}{2}]\times2=21$$

2.5 패킷 트랜잭션 시간이 서로 다른 경우

이 경우는 여러 개의 연결망을 갖는 네트워크에 적용할 수 있다.

[Lemma 3] 패킷 트랜잭션 시간이 상이한 경우의 평균 대기 시간은

$$AW=\frac{(n-1)[\sum_{i=1}^{P}m_i[(m_i-1)t_i+\sum_{j=1,j\neq i}^{P}m_jt_j]]}{m}+\frac{\sum_{i=1}^{P}[\sum_{j=1}^{i}m_i(m_{j-1}t_{j-1})+\sum_{j=1}^{m_i}(j-1)t_i]}{m}$$ (11)

(단, $m_0=0, t_0=0$)이다.

(증명) 전체 사용자 수(m), i번째 대역폭 라인의 사용자 수(m_i), 패킷 수(n), i번째 대역폭 라인을 사용하는 패킷 트랜잭션 시간(t_i)이 주어지는 경우, i번째 사용자의 대기 시간을 구하기 위해 먼저 모든 사용자가 마지막 패킷을 보내는 구간과 그렇지 않은 구간으로 구분한다. 그러면 마지막 패킷을 보내는 구간 이전까지 i번째 사용자의 대기 시간은 (패킷 수 - 1) × [(i번째 사용자가 속한 그룹의 나머지 사용자 수) × t_i + (i번째 그룹을 제외한 나머지 모든 그룹에 대한 패킷 트랜잭션 시간의 합)]이 되고 마지막 패킷을 보내는 구간에서 i번째 사용자가 겪는 대기 시간은 자신을 선행하는 나머지 사용자들의 패킷 트랜잭션 시간의 합이다. 이를 일반화하여 평균 대기 시간을 구하면 식 (11)이 된다.

[Lemma 4] 패킷 트랜잭션 시간이 상이한 경우의 평균 반환 시간은

$$AT=\frac{m(n-1)\sum_{i=1}^{P}m_it_i+\sum_{i=1}^{P}[\sum_{j=1}^{i}m_i(m_{j-1}t_{j-1})}{m}+\frac{\sum_{j=1}^{m_i}jt_i]}{m}$$ (12)

(단, $m_0=0, t_0=0$)이다.

(증명) 전체 사용자 수(m), i번째 라인을 사용하는 사용자 수(m_i), 패킷 수(n), i번째 라인 종류를 사용하는 패킷 트랜잭션 시간(t_i)이 주어지는 경우, i번째 사용자의 반환 시간을 구하기 위해 먼저 모든 사용자가 마지막 패킷을 보내는 구간과 그렇지 않은 구간으로 구분한다. 그러면 마지막 패킷을 보내는 구간 이전까지 i번째 사용자의 대기 시간은 (패킷 수 - 1) × [사용자 수(m_i) × 패킷 트랜잭션 시간(t_i)의 합]이 되고 마지막 패킷을 보내는 구간에서의 반환 시간은 자신을 선행하는 나머지 사용자들의 패킷 트랜잭션 시간의 합 + 자신의 패킷 트랜잭션 시간이 되므로 이를 일반화하여 평균 반환 시간을 구하면 식 (12)가 된다.

[예제 2] m = 4, m_1 = 2, m_2 = 2, t_1 = 2, t_2 = 1, n = 3인 경우의 평균 대기 시간과 평균 반환 시간은 아래와 같다((그림 3)에서 A, B를 첫 번째 그룹, C, D를 두 번째 그룹으로 간주하여 참조).

$$AW=\frac{(3-1)\times[2\times[(2-1)\times2+2\times1]}{4}+\frac{2[(2-1)\times1+2\times2]]+[2\times2\times2+2+1]}{4}$$
$$=\frac{47}{4}=11.75$$

$$AT = \frac{4\times(3-1)\times[2\times2+2\times1]}{4} + \frac{[(2+2\times2)+2\times(2\times2)+(1+2\times1)]}{4} = \frac{65}{4} = 16.25$$

2.6 서버 분산의 경우

각 대역폭 라인에 대해 웹 서버를 분산하여 서비스하는 경우에는 라인의 개수가 P이므로, 각 라인에 대해 식 (9)와 식 (10)을 개별적으로 적용하여 평균 대기 시간과 평균 반환 시간을 구하여 더한 후에 이를 P로 나누어주면 전체 평균 대기 시간과 평균 반환 시간을 구할 수 있다. 즉, 두개의 평균은 식 (13)처럼 표현된다.

$$AW = \sum_{k=1}^{P}\frac{AW_k}{P} = \sum_{k=1}^{P}\left[\frac{\sum_{i=1}^{m_k}(m_k-i)t_k + m_k(n_k-1)(m_k-1)t_k}{m_k}\right] / P$$

$$AT = \sum_{k=1}^{P}\frac{AT_k}{P} = \sum_{k=1}^{P}\frac{[m_k(n_k-1)+\frac{(m_k+1)}{2}]t_k}{P} \quad (13)$$

단, 식 (13)에서 AW_k는 k 번째 대역폭 라인의 평균 대기 시간, AT_k는 k번째 라인의 평균 반환 시간, m_k는 k번째 라인의 사용자 수, n_k는 k 번째 라인의 패킷 수, t_k는 k번째 라인의 패킷 트랜잭션 시간 등이다.

3. 성능 평가 및 분석

웹 환경에서의 실제 성능 실험은 컴퓨터 시스템의 운영 체제, 네트워크의 토폴로지, 동시 접속 인원수나 프로토콜 그리고 컴퓨터, 응용 소프트웨어 및 통신 장비 등의 성능 등에 의해 결정되기 때문에 정확한 최소 값을 측정하기가 대단히 어렵다. 본 연구는 이러한 여러 가지 요인들 중에서 프로토콜(TCP 및 T-TCP)과 사용자 수와 관련된 성능 문제에 초점을 맞춘 것으로 실제 환경에 대한 이론적인 상한 값을 제공하므로 측정값보다 항상 작은 최적 값이 나오게 된다. 3.1 절에서는 실제 실험 환경에서 계산값과 실험값을 살펴보고 3.2 절에서는 계산 경험을 보이기로 한다.

3.1 실험 환경 및 분석

이더넷 LAN 환경에서 서버(Alzza Linux 6.1 OS / Apache 1.3.x Web Server)와 클라이언트(SUN ultra-5 WS / Solaris 2.6 OS)를 10 Mbps 이더넷으로 직접 연결하고 테스트하였다. 서버에서 사용된 파일의 크기(바이트)는 3000, 6000, 12000, 20000, 50000이다. LAN에서 MSS는 1460 바이트이고 측정된 R(RTT)은 0.5ms이다. 클라이언트에서는 UNIX C 언어를 이용하여 소켓 인터페이스를 구성하고, 동시 접속 실험을 위해 사용자 수는 각각의 파일 크기에 대해 1, 5, 10 명씩을 가정하였으며 이 수만큼 쓰레드(thread)를 생성하여 처리하였다. 시간 측정은 쓰레드 시작과 끝에서 타이머를 작동 및 종료시키고 그 차이를 계산하는 프로그램[17]을 이용하였다. 실험을 위한 클라이언트 프로그램의 구성은 (그림 4)와 같다.

```
int main(int argc, char *argv[])
{
    Input the number of users;
    For each user,
        Create thread routine(thread_routine());
    Wait until all threads are terminated;
}

/* thread routine */
void *thread_routine()
{
    Start timer(t_start);
    Create socket interface;
    Open connection;
    Send HTTP request(GET file);
    Receive the file;
    Stop timer(t_stop);
    Get the transaction time(get_utime);
}
```

(그림 4) 실험에 사용된 프로그램의 구성

먼저, 2.2절에서 구한 식 (1)~식 (8)을 이용하여 계산하면 <표 1>을 얻는다. 표에서 T_{min}과 $T_{T\text{-}TCP}$가 동일하게 나타나고 있는데, 실험 환경에서 계산된 S 값이 0이기 때문이다. 물론 S가 0이 아닌 경우에는 2.2절에서도 보았듯이 $T_{T\text{-}TCP}$

〈표 1〉 실험 환경에서의 계산값

구분	Bandwidth (Mbps)	RTT (ms)	MSS (byte)	FileSize (byte)	K (pkts)	L (pkts)	T_f (ms)	M (pkts)	S (pkts)	W (ms)	T_{min} (ms)	T (ms)	$T_{T\text{-}TCP}$ (ms)
수식					(1)	(2)	(3)	(4)	(5)	(6)	RTT+T_f	(7)	(8)
계산값	10	0.5	1460	3000	2.05	0.43	2.46	0.43	0	0.5	2.96	3.46	2.96
				6000	4.11	0.43	4.92	0.43	0	0.5	5.42	5.92	5.42
				12000	8.22	0.43	9.83	0.43	0	0.5	10.33	10.83	10.33
				20000	13.70	0.43	16.83	0.43	0	0.5	16.88	17.38	16.88
				50000	34.25	0.43	40.96	0.43	0	0.5	41.46	41.96	41.46

의 값이 T_{min}보다 R×S 만큼 커지게 된다.

3.1.1 패킷 트랜잭션 시간이 동일한 경우

<표 2>에서 계산값의 T와 $T_{T\ TCP}$는 <표 1>에서 얻은 값이고, 실험값의 T와 $T_{T\ TCP}$는 (그림 4)의 프로그램을 동일한 파일 크기 및 사용자 수에 대해 10번을 반복하여 얻은 결과의 평균값이다. <표 2>에서 패킷 수(n)는 파일 크기를 MSS로 나눈 값(K)의 ceiling 값이다. 각 패킷 트랜잭션 시간(t)은 전체 트랜잭션 시간(T 또는 $T_{T\text{-}TCP}$)을 패킷 수(n)로 나눈 값이다. 2.4절의 식 (9)~식 (10)에서 n, m, t를 대입하면 각각 평균 대기시간(AW)과 평균 반환시간(AT)을 얻는다. 계산값과 실험값의 단위는 ms이다. 표에 나타난 바와 같이 파일 크기에 따라 실험값과 계산값 사이의 관계는 도출하기가 어려우며, TCP와 T-TCP의 모든 경우에 있어서 당연히 이론적인 계산값이 실험값보다 작으며, 계산값이 상한값임을 알수 있다. 실험값의 T(t)와 계산값의 그것이 서로 다른 이유는 서버 측에서 프로세스 경쟁 등의 운영 체제의 성능 요소가 포함되기 때문이다. (그림 4)의 클라이언트 의사 코드에서 실제 측정 타이머는 연결 설정과 파일 수신 종료 사이에만 셋팅되어 있다. 따라서 실험값과 계산값의 차이는 대략 서버 운영체제의 프로세스 처리 시간으로 생각될 수 있다.

3.1.2 패킷 트랜잭션 시간이 서로 다른 경우

각 파일 크기에 대해 사용자 그룹을 5명(m_1)과 10명(m_2)의 두 그룹으로 나누어서 계산하였다. <표 3>에서 계산값

〈표 2〉 패킷 트랜잭션 시간이 동일한 경우의 비교

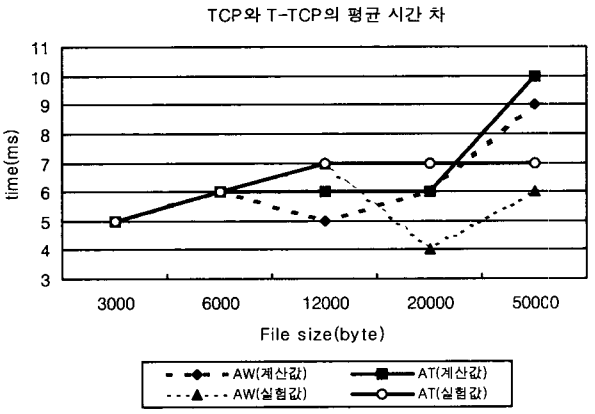
파일크기 (byte)	패킷수 (n)	사용자수 (m)	계산값						실험값					
			TCP(ms)			T-TCP(ms)			TCP(ms)			T-TCP(ms)		
			T(t)	AW	AT	$T_{T\ TCP}$(t)	AW	AT	T(t)	AW	AT	$T_{T\ TCP}$(t)	AW	AT
3000	3	1	3.46 (1.15)			2.96 (0.99)			14.30 (4.77)			14.25 (4.75)		
		5		12	15		10	13		48	62		47	61
		10		26	29		22	25		107	122		107	121
6000	5	1	5.92 (1.18)			5.42 (1.08)			18.75 (3.75)			18.25 (3.65)		
		5		21	27		19	25		68	86		66	84
		10		48	54		44	49		152	171		148	166
12000	9	1	10.83 (1.20)			10.33 (1.15)			35.63 (3.96)			35.13 (3.90)		
		5		41	52		39	49		135	170		133	168
		10		92	103		88	98		303	339		298	333
20000	14	1	17.38 (1.24)			16.88 (1.21)			89.38 (6.38)			88.88 (6.35)		
		5		67	84		65	82		345	434		343	432
		10		151	168		147	164		775	864		772	860
50000	35	1	41.96 (1.20)			41.46 (1.18)			276.25 (7.89)			275.75 (7.88)		
		5		166	208		163	204		1089	1365		1087	1363
		10		373	415		366	408		2450	2726		2447	2772

〈표 3〉 패킷 트랜잭션 시간이 서로 다른 경우의 비교

파일크기 (byte)	패킷 수 (n)	사용자 수/그룹 (m_i)	계산값						실험값					
			TCP(ms)			T TCP(ms)			TCP(ms)			T-TCP(ms)		
			$T(t_i)$	AW	AT	$T_{T\ TCP}(t_i)$	AW	AT	$T(t_i)$	AW	AT	$T_{T\ TCP}(t_i)$	AW	AT
3000	3	5	3.46 (1.15)	36	38	2.96 (0.99)	31	33	18.0(6.0)	157	168	17.5(5.8)	152	163
	3	10							13.0(4.3)			12.5(4.2)		
6000	5	5	5.92 (1.18)	70	75	5.42 (1.08)	64	69	22.0(4.4)	224	240	21.5(4.3)	218	234
	5	10							17.0(3.4)			16.5(3.3)		
12000	9	5	10.83 (1.20)	138	148	10.33 (1.15)	133	142	38.0(4.2)	471	505	37.5(4.2)	464	498
	9	10							36.0(4.0)			35.5(3.9)		
20000	14	5	17.38 (1.24)	230	246	16.9 (1.21)	224	240	64.0(4.6)	1238	1327	63.5(4.5)	1234	1320
	14	10							109.0(7.8)			108.5(7.8)		
50000	35	5	41.96 (1.20)	575	616	41.46 (1.18)	566	606	290.0(8.3)	3990	4276	289.5(8.3)	3984	4269
	35	10							292.0(8.3)			291.5(8.3)		

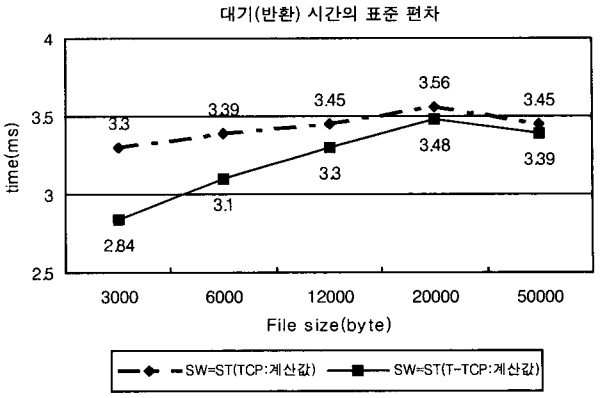
의 $T(t_i)$와 $T_{T\text{-}TCP}(t_i)$는 <표 1>에서 얻은 값이고, 실험값의 $T(t_i)$와 $T_{T\text{-}TCP}(t_i)$는 (그림 4)의 프로그램을 10번 반복 실행하여 얻은 결과의 최소값이다. 각각의 AW와 AT는 2.5절의 식 (11)~식 (12)를 적용하여 얻어졌다. 표에서 보듯이 이 경우에도 파일 크기에 따른 실험값과 계산값 사이의 관계는 도출하기가 어려우며, TCP와 T-TCP의 두 경우 모두에 있어서 계산값이 실험값보다 작음을 알 수 있다.

다음으로 TCP와 T-TCP의 성능을 비교해 본다. (그림 5)는 <표 3>의 계산값과 실험값에서 TCP의 AW, AT에서 T-TCP의 AW, AT를 뺀 값을 정리한 것이다. 그림에서 보듯이 T-TCP가 4 ms~10 ms 만큼 우수하며, 계산값의 반환 시간의 경우가 가장 좋다.



(그림 5) TCP와 T-TCP의 평균 대기(반환) 시간 차

(그림 6)은 <표 3>의 계산값에서 TCP와 T-TCP의 표준편차를 정리한 것으로 대기 시간과 반환 시간의 표준편차의 값이 TCP와 T-TCP에 있어서 모두 동일하며 T-TCP가 TCP에 비해 작은 것을 알수 있다. 그림은 생략하였지만 실험값에 대한 표준편차의 특성 역시 계산값에서와 동일하게 T-TCP가 우수하다.



(그림 6) 대기 시간 및 반환 시간의 표준편차

3.2 계산 경험 및 분석

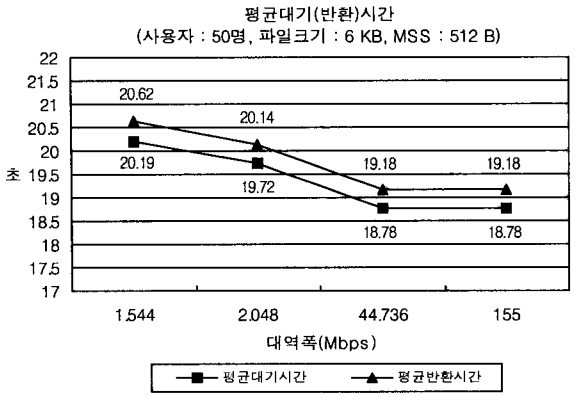
이제, 보다 다양한 계산 경험을 얻기 위한 계산 환경을 <표 4>에 나타내었다. 표에 나타난 값들은 가정 값으로 대역폭(Bandwidth)은 T1(1.544 Mbps), E1(2.048 Mbps), T3(44.736 Mbps) 및 ATM OC-3(155 Mbps)를 사용하고, TCP의 MSS는 WAN에서는 512 바이트이므로 그 값을 사용하였다. 다음으로 RTT는 WAN에 연결된 전용선에서 측정치로 주로 사용되는 값으로 0.1 ms를 사용하였다. 파일 크기는 6 KB, 12 KB, 1 MB(ATM은 제외)를 사용하였는데, 이 값은 웹 서버에 저장된 파일 크기를 의미한다. 물론 여기서 사용되는 값들은 임의로 변경시켜 계산해도 무방하다. 먼저, 각 대역폭에 대해 K, L, T_f, M, S, W, T_{min}, T, $T_{T\text{-}TCP}$를 구하면 <표 4>와 같다.

3.2.1 패킷 트랜잭션 시간이 동일한 경우

이제 예로서 <표 4>에서 각 대역폭에 대해 파일 크기가 6000 바이트인 경우에 대해 사용자 수(m)를 50 명, 패킷 수(n)를 K의 ceiling, 그리고 패킷 트랜잭션 시간(t)을 T/n으로 하여 2.4 절에서 유도한 식 (9)~식 (10)을 적용하면 (그림 7)의 평균 대기시간(AW)과 평균 반환 시간(AT)을 얻는다.

〈표 4〉 대역폭에 따른 트랜잭션 시간

Bandwidth (Mbps)	RTT (ms)	MSS (byte)	FileSize (byte)	K (pkts)	L (pkts)	T_f (sec)	M (pkts)	S (pkts)	W (sec)	T_{min} (sec)	T (sec)	$T_{T\ TCP}$ (sec)
T1 (1.544)	0.1	512	6000	11.72	37.70	0.03	11.72	2	0.3	0.13	0.43	0.33
			12000	23.44	37.70	0.06	23.44	3	0.4	0.16	0.56	0.46
			1 M	2048.00	37.70	5.56	37.70	4	0.5	5.66	6.16	6.06
E1 (2.048)	0.1	512	6000	11.72	50.00	0.02	11.72	2	0.3	0.12	0.42	0.32
			12000	23.44	50.00	0.05	3.44	3	0.4	0.15	0.55	0.45
			1 M	2048.00	50.00	4.19	50.00	4	0.5	4.29	4.79	4.69
T3 (44.736)	0.1	512	6000	11.72	1092.19	0.00	11.72	2	0.3	0.10	0.40	0.30
			12000	23.44	1092.19	0.00	23.44	3	0.4	0.10	0.50	0.40
			1 M	2048.00	1092.19	0.19	1092.19	9	1.0	0.29	1.29	1.19
ATM-OC3 (155)	0.1	512	6000	11.72	3784.18	0.00	3784.18	2	0.3	0.10	0.40	0.30
			12000	23.44	3784.18	0.00	3784.18	3	0.4	0.10	0.50	0.40



(그림 7) 평균 대기 시간 및 평균 반환 시간

3.2.2 패킷 트랜잭션 시간이 서로 다른 경우

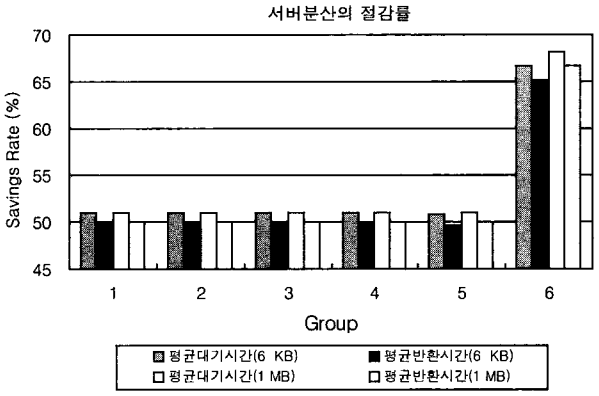
사용자 수를 10명에서 100명까지 변화시키면서 6가지 그룹을 고려하였다. 대역폭이 서로 다른 라인의 조합은 <표 4>에서 2개의 T1, T1과 E1, 2개의 E1, T1과 T3, E1과 T3를 고려하였고 다음으로 T1, E1, T3를 혼합해서 사용하는 것을 고려하였다. 각 그룹에 대한 사용자 수(m_i)는 라인이 2개인 경우는 각 절반씩을 사용하고, 3개인 경우에는 3 : 3 : 4의 비율을 사용하였다. 각 그룹의 패킷 트랜잭션 시간(t_i)은 <표 4>에서 전체 트랜잭션 시간(T)을 패킷 수(n)로 나눈 시간을 사용하였다. <표 5>는 파일 크기가 6,000바이트이고 MSS가 512바이트인 경우의 결과이다.

이제, 서버 분산의 경우 절감률을 보기 위해 다음의 Savings Rate를 정의한다.

$$\text{Savings Rate} = \frac{a(a') - b(b')}{a(a')} * 100(\%) \qquad (14)$$

여기서, a = 서버를 분산하지 않는 경우의 평균 대기 시간
a′ = 서버를 분산하지 않는 경우의 평균 반환 시간
b = 서버를 분산하는 경우의 평균 대기 시간
b′ = 서버를 분산하는 경우의 평균 반환 시간

식 (14)에서 a, a′은 각각 2.5 절의 식 (11)과 식 (12)를 이용해서 얻은 결과이고, b, b′는 각각 2.6절의 식 (13)을 이용하여 얻은 결과이다. 동일한 방법으로 파일 크기가 1 MB이고 MSS가 512바이트인 경우를 추가하여 계산한 다음, 파일 크기가 6 KB와 1 MB에 대해 Savings Rate(%)를 나타내면 (그림 8)을 얻는다.



(그림 8) 서버 분산의 효과

(그림 8)에서 서버 분산을 통한 절감률은 6 KB, 1 MB의 두 가지 경우 모두 평균 대기 시간이 평균 반환 시간보다 약간 우수하며, 전체적으로 평균 50~51%에 이르고 있음을 알 수 있다.

〈표 5〉 대역폭을 조합하는 경우의 평균 대기 시간과 평균 반환 시간

그 룹	소 속 대역폭	그룹인원수(m_i)	평균대기 시간(a)	평균반환 시간(a′)	평균 대기 시간 (서버 분산 : b)	평균 반환 시간 (서버 분산 : b′)
라인이 하나인 경 우	T1	50명	20.29	20.72	20.29	20.72
	E1	50명	19.72	20.14	19.72	20.14
	T3	50명	18.60	18.99	18.60	18.99
1	T1 * 2	각 25명	9.94	10.37	20.29(10.15)	20.72(10.36)
2	T1	25명	9.94	10.37	19.60(9.80)	20.45(10.23)
	E1	25명	9.66	10.08		
3	E1 * 2	각 25명	9.66	10.08	19.72(9.66)	20.14(10.08)
4	T1	25명	9.94	10.37	19.05(9.53)	19.87(9.94)
	T3	25명	9.11	9.50		
5	E1	25명	9.66	10.08	18.77(9.39)	19.58(9.79)
	T3	25명	9.11	9.50		
6	T1	15명	5.76	6.19	18.61(6.20)	19.85(6.62)
	E1	15명	5.64	6.05		
	T3	20명	7.21	7.61		

4. 결 론

최근 몇년 사이에 인터넷의 수요는 폭발적으로 증가되어 왔다. 그 이유중의 하나가 웹의 확산이다. 그러나 인터넷의 긍정적인 영향에도 불구하고 실제 접속 속도는 예전에 비해 상당히 떨어지고 있다. 웹의 대표적인 프로토콜인 HTTP는 하부 네트워크의 구조와 관계없이 동작할 수 있도록 설계된 프로토콜이지만 현실적으로 인터넷의 하부 네트워크는 거의 TCP/IP 프로토콜이 사용되고 있다. 따라서, HTTP는 TCP 계층과 연동되어야 하는데 HTTP와 TCP는 서로 최적으로 작동되지 않는다는 문제가 있다. 이 문제점들은 초기 연결 설정시에 수백 밀리 초의 시간 지연을 발생시키는 점과 웹 파일의 크기가 일반적으로 작기 때문에 slow start의 영향을 매우 많이 받는다는 점, 그리고 연결 해제시에 일정 시간 동안 TIME_WAIT 상태를 유지해야 하는 점등이다. 본 연구에서는 먼저 HTTP over TCP에 대한 성능 문제를 검토하고, 윈도우 크기 및 slow start를 고려하면서 각 대역폭에 따른 트랜잭션 시간 등을 계산하였다. 이를 토대로 여러 명의 사용자가 동시에 접속하는 경우의 평균 대기 시간과 평균 반환 시간을 계산하는 공식을 유도하였다. 즉, 본 연구는 TCP와 사용자 수에 따른 성능 문제에 초점을 맞춘 것으로 종단 사용자의 QoS로 평균 지연 시간과 평균 반환 시간을 가정했을때 실제 환경에 대한 이론적인 상한 값을 제시하는데 목적을 두었다. 실제 실험 및 계산 경험을 통해 공식들이 잘 작동됨을 알수 있었고, 특히 대역폭을 조합하여 웹 서버의 부하를 적절히 분산하면 약 50% 정도의 시간 절감률을 얻을 수 있음을 보였다. 본 연구에서 얻은 결과를 이용하면, 사용자가 허용하는 평균 대기 시간과 반환 시간을 충족시키기 위해 얼마나 많은 대역폭을 어떻게 분산하는 가에 대한 대안을 얻을 수 있다. 한편, T-TCP의 대기 및 반환 시간 특성이 TCP보다 좋으며 전송량과 대역폭이 작을수록 더욱 효율적임을 확인하였다. 앞으로의 연구에서는 기존의 TCP 프로토콜을 개선한 다른 프로토콜들에 대한 성능 분석과 함께 트랜잭션 시간이 확률 특성을 갖는 경우의 성능 파라메터의 유도 및 분석이 기대된다.

참 고 문 헌

[1] T. Berners Lee, R. Fielding and H. Frystyk, "Hypertext Transfer Protocol-HTTP/1.0," RFC-1945, 1995.

[2] R. Braden, "Extending TCP for Transactions-Concepts," RFC-1379, 1992.

[3] R. Braden, "T/TCP-TCP Extensions for Transactions : Functional Specification," RFC-1644, 1994.

[4] T. Faber, J. Touch and W. Yue, "Avoiding the TCP TIME_WAIT state at Busy Servers," ISI, 1997.

[5] R. Fielding, H. Frystyk and T. Berners Lee, "Hypertext Transport Protocol-HTTP/1.1," Internet Draft, 1996.

[6] J. Heidemann, K. Obraczka and J. Touch, "Modeling the Performance of HTTP Over Several Transport Protocols," IEEE/ACM Transactions on Networking, Vol.5, No.5, pp. 616-630, 1997.

[7] J. Heidemann, "Performance Interactions Between P-HTTP and TCP Implementations," ACM Computer Communications Review, pp.65-73, 1997.

[8] V. Jacobson, "Congestion Avoidance and Control," ACM Sigcomm '88, 1988.

[9] V. Jacobson and M. Karels, "Congestion Avoidance and Control," ACM CCR, Vol.18, No.4, pp.314-329, 1990.

[10] J. Postel, "Transmission Control Protocol," RFC-793/STD-007, September, 1981.

[11] J. Mogul, "The Case for Persistent-Connection HTTP," ACM Sigcomm '95, pp.299-313, 1995.

[12] J. Mogul and S. Deering, "Path MTU Discovery," RFC-1191, DECWRL, Stanford University, 1990.

[13] V. Padmanabhan and J. Mogul, "Improving HTTP Latency," Proc. of the Second International WWW Conference, 1994.

[14] E. Rescorla and A. Schiffman, "The secure Hypertext Transfer Protocol," ftp://ftp.is.co.za/internet-drafts/draft-ietf-wts-shttp-03.txt, 1996.

[15] S. E. Spero, "Next Generation Hypertext Transport Protocol," Internet Draft., 1995.

[16] R. W. Stevens, TCP/IP Illustrated : TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols. Reading : Addison-Wesley, Vol.3, 1996.

[17] R. W. Stevens, UNIX Network Programming : Appendix 4(Timer Routine), Prentice-Hall, 1991.

[18] J. Touch, J. Heidemann and K. Obraczka, "Analysis of HTTP Performance," USC/Information Sciences Institute, 1996.

[19] J. Touch, "TCP Control Block Interdependence," RFC-2140, ISI, 1997.

[20] 한국전자통신연구소, "전자거래에서 디렉토리를 위한 에이전트 연구", pp.56-108, 1997.

이 용 진

e-mail : yjlee@woosong.ac.kr

1983년 고려대학교 산업공학과 (학사, 석사)

1995년 고려대학교 전산과학과(박사)

1995년~현재 우송대학교 컴퓨터전자정보공학부 부교수

관심분야 : 네트워크 성능 평가, 인터넷 QoS, 분산 시스템