논문 03-02-19

RTO 조정을 통한 이더넷(Ethernet) 내의 웹 서버의 성능향상

Improving the Performance of Web Server in Ethernet by Controlling the RTO

金 鎭 喜*, 權 景 熙*

Jin-Hee Kim*, Kyung-Hee Kwon*

요 약

본 논문에서는 웹 서버의 부하 분산을 위해 네트워크를 LAN 내부와 LAN 외부로 나누어 각각의 웹 서버를 할당한다. LAN 내부의 노드들에 의한 접속만이 가능한 웹 서버의 경우 RTO(Retransmission Time Out : 재전송타임아웃)라는 TCP 파라미터에 의해 많은 영향을 받는다. 일반적으로 운영체제에 의해 설정된 RTO 값은 LAN에서는 지나치게 큰 값으로 네트워크와 웹 서버의 성능을 저하시킨다. 그래서 본 논문에서는 기존의 재전송 타임아웃의 알고리즘 방식을 이용하지 않고, 조정된 RTO값을 이용하여 LAN 내부의 웹 서버의 성능을 향상시킬 수 있음을 확인하였다.

Abstract

We divide networks of an organization into internal networks and external networks to distribute the load of web server, and allocate separate web server for each network. The performance of web server for internal networks can be affected seriously by RTO(Retransmission Timeout). The value of RTO set by default in the OS of an web server is so large that it degrades the performance of web server. Therefore, this paper suggests not useing but applying conventional algorithm to calculate RTO, but to apply newly controlled value of RTO, and it showed improvement of the performance of web server.

keyword : performance, RTO, RTT, 처리량(throughput), 이더넷(Ethernet)

Ⅰ. 서 론

최근에 이르러 웹은 거의 모든 종류의 데이터와 서비스를 분배하는 수단으로 활용되고 있으나 상대적으로 늘어나는 수요에 비해 성능과 신뢰도가 이를 뒷받침하지 못하고 있다. 웹 서비스의 많은 활용은 제한

* 檀國大學校 電子計算學科
(Dept. of Computer Science, Dan-Kook Univ.)
※ 이 연구는 2002 학년도 단국대학교 대학 연구비의 지원으로 연구되었습.
接受日:2003年 8月 24日, 修正完了日:2003年 11月 14日

된 대역폭과 서버 측 용량의 한계로 인하여 네트워크 혼잡과 같은 많은 문제점을 야기하고 있다. 그 중에 하나가 트래픽의 과부하를 들 수 있다. 이러한 문제점의 해결책으로 여러 가지 하드웨어, 소프트웨어적인 방법들이 소개되고 있으나 여전히 웹 서버의 트래픽 과부하시에 느린 접속 속도 및 시간 초과로 인한 재접속 등은 해결해야 할 과제이다.

웹 서버의 접속 수요의 증가에 따른 서비스 질을 높이기 위한 방법으로 기존 서버를 업그레이드하는 방법과 새로운 서버를 추가하는 방법에 이르기까지 많은 해결책들이 제시되고 있다.

현재는 네트워크 설계 시 처음부터 서버의 용도와 목적에 따라, 예를 들어 파일 서버, 메일 서버, DB서버, ftp 서버 등으로 나누어 서버를 구축하는 방법 등도 사용되고 있다.

서버를 업그레이드한다는 것은 기존 서버에 필요한 제품군의 추가가 아니라 교환의 개념이 더 강하다. 그리고 데이터가 늘어날 때마다 반복적인 작업이 이루어져야 하며 기존 서버내의 자료 이동 등의 복잡함을 안고 있다. 그래서 서버의 부하가 증가하면 서버의 개수만 늘려 부하를 분산시키는 것이 더 안정적이고 빠른 웹 서비스를 위한 분산방법으로 이용되고 있다.

기존의 서버분산 방법으로 NAT(Network Address Translator) 기반의 가상서버 방식 등을 들 수 있으나 본 논문에서는 웹 서버를 외부 접속용과 내부 접속용으로 분리시키는 방법을 제안한다. 여기서 외부는 인터넷을 의미하고 내부는 이더넷(Ethernet)으로 구성된 LAN을 의미한다. 이는 TCP 상에서 패킷 손실로 인한 데이터 재전송 시 RTO 값을 외부와 내부로 달리 적용하기 위한 것이다.

TCP는 UDP와는 달리 신뢰성을 보장하는 전송 방식으로서, 연결된 두 종단 사이에서 패킷의 손실 및 오류, 혹은 과부하로 인해 특정 시간 내에 ACK(Acknowledge)를 받지 못하면 타임아웃을 적용하고, ACK가 없는 패킷에 대하여 다시 한번 전송이 이루어진다. 이때의 타임아웃의 시간 간격 및 재전송 발생 주기 등은 네트워크상에서 전체적인 지연시간에 영향을 미칠 수 있다.

데이터 전송시 전체적인 지연시간은 전송거리에 비례한다고 할 수 있다. 즉 거리가 길면 당연히 전송시간도 길어지게 되며 거리가 짧은 내부 접속용 서버보다 외부 접속용 서버가 재전송 간격의 최대값을 더 길게 두어야한다.

이 때 RTO의 간격이 좁아서 지나치게 많은 재전송이 이루어지면 트래픽 과부하로 RTT(Round Trip Time)는 더욱 길어지고 처리량(Throughput)은 감소하는 등의 성능 저하를 가져오게 되는 것이다.

원래의 RTO를 구하는 알고리즘은 네트워크의 송신 시간과 송신에 대한 ACK의 수신 시간의 차를 RTT로 하여, RTT 평균의 두 배로 설정한다. 그리고 이때의 RTT는 재전송이 없었던 세그먼트만을 가지고 측정된다. 그러나 이러한 알고리즘은 불필요한 재전송을 유발하는 RTT의 크기를 제어하는 기능이 없다. 또한 RTT의 통계적 분산을 고려하지 않는다는 점에서 새로이 계산되는 RTT의 변화가 작다면 RTT에 두 배로 타임아웃을 설정할 필요가 없을 것이다. 그리고 이 통계적 분산이 크다는 것은 신뢰성이 떨어지는 것을 의미하며 타임아웃값이 RTT와 거의 같은 값을 가져서는 안 된다는 것을 뜻한다. 이러한 문제점을 해결하려고 나온 것이 Jacobson의 알고리즘으로 평균과 편차를 고려하여 RTO값을 계산한다[1]. 그러나 웹 서버 접속 비율이 외부 접속과 내부 접속이 비슷한 경우를 고려해 보자. 그리고 내부 접속용 서버에서 재전송이 일어날 경우 적용되는 RTO 값이 외부 접속에 대한 평균 편차를 고려하여 계산된 RTO 값이 적용된다면 지나치게 긴 지연시간으로 인하여 오히려 더 긴 RTT 값을 가져올 것이다. 따라서 재전송 타임아웃의 범위로 설정되는 기존의 알고리즘 방식이 그대로 적용될 경우 외부 접속용 서버에 적합한 최대값의 범위는 내부 접속용 서버에는 적합하지 않다.

II. 관련 연구

웹 서버의 성능 향상에 관한 기존 연구는 크게 세 가지로 이루어져 왔다.

첫째, 웹 서버 자체의 성능 향상방법으로 서버 업그레이드 및 서버 분산 방법이 있다[13][15][18].

둘째, TCP 파라미터 조정을 통한 웹 서버 성능 향상에 관한 연구가 있다[17].

셋째, 웹 클라이언트의 요구들을 종류별로 분류한 뒤 각각의 웹 문서의 접근 확률 같은 기준이 사용된 연구가 있다[16].

본 논문에서는 기존의 서버 분산 방법과 알고리즘을 사용하지 않고 외부접속용 서버와 내부접속용 서버로 분리하여 각각의 서버에 적합하게 TCP 파라미터를 조절하여 성능향상을 시도하였다. 그리고 이렇게 분리된 서버에 RTO 값을 다르게 적용하는 방법으로 연구를 진행하였으며 이러한 연구가 기존 연구에서는 전혀

이루어지지 않은 것을 확인할 수 있었다.

본 연구는 학생수 만 여명 규모의 국내 대학에 연평균 이용률이 20~30%에 불과한데도 일시적인 트래픽 과부하로 인한 낮은 응답속도에 대한 문제점을 개선시키기 위해 출발하였다. 이용률 조사 결과 외부에 비해 내부 접속률이 두 배 가까이 많은 것으로 확인되었는데도 불구하고 재전송 타임아웃으로 설정된 최소값과 최대값의 범위는 0.2 ~ 120 sec였다. 이는 LAN 환경에서 RTT 값으로 0.001sec가 적당하다고 생각한다면[1] 120sec는 상당히 큰 값으로 외부 접속에 대한 경우를 고려한 설정으로 볼 수 있다.

이는 많은 트래픽으로 인한 접속 지연과, 동일 LAN 상에 위치하는 노드에 재전송이 발생할 경우 지나치게 긴 지연으로 인하여 RTT 값이 길어질 수 있음을 의미한다. 대학과 같은 환경에서 웹 서버에 대한 내부 접속이 외부 접속보다 많은 사실을 고려한다면 서버 분산 시 외부접속 내부접속 서버로 분리하여 각각의 서버에 적합한 TCP 파라미터 값을 설정하는 것이 네트워크의 성능향상을 가져다 줄 것이다. 더욱이 내부 접속용 서버 같은 경우 같은 LAN에 위치한다면 서버에 설정되는 RTO의 값을 특정 범위 내에서 가변적인 값으로 할당되게 하지 않고 LAN의 특성을 고려하여 RTO 값을 상수화 시킬 수 있다.

Ⅲ. 네트워크 성능 분석

본 논문에서는 외부 접속용과 내부 접속용으로 서버를 분리시킨 후 이더넷에 위치한 내부접속용 서버에 다양한 RTO값을 적용해 보았다. 그리고 다양한 RTO 값의 설정에 따른 이더넷 이용률과 처리량을 확인하였다. 이더넷의 이용률은 다음의 식으로 나타낼 수 있다.

이더넷 이용률
=(Totalbyte * 8) /(Bandwidth*(startTime stoptime))
전체바이트 량 측정시간

3-1. 시뮬레이션

본 논문에서는 네트워크 시뮬레이터인 ns-2를 이용하였으며, 시뮬레이션 환경에서 다음과 같은 가정하에 연구를 진행하였다.

첫째, TCP 타입은 TCP Reno 이다. 이는 TCP의 여러가지 구현 중 대표적이며 가장 많이 이용되는 것이 Reno이기 때문이다.

둘째, ACK에 대한 손실률을 0%로 하였으며, 이는 수신측에서부터 ACK가 전송되는 동안에 error가 없음을 가정한 것이다.

셋째, 데이터 송수신 방법은 반이중(half-duplex) 으로 가정하였다.

[표1] 에서 TCP 프로토콜 Type을 클라이언트와 서버에 각각 다르게 설정되어 있다. 이는 클라이언트에서 사용하는 RTO 알고리즘과 서버에서 사용하는 RTO 알고리즘이 다르게 적용됨을 의미한다.

표 1. 시뮬레이션 모델
Table 1. Simulation model

Maximum packet size		1000 bytes
TCP type		HTTP 1.1
TCP Type	Client	TCP Reno
	Server	TCP Reno (RTO값 고정)
Bandwidth		10 Mbps
Delay		1 ms
Queue		DropTail

클라이언트는 LAN에 위치한 내부서버 접속뿐만 아니라 외부로의 접속이 있을 수 있으므로 상황에 따라 RTO 값이 다르게 적용되어야 한다. 외부로 나갈 경우에는 전송거리를 고려하여 RTO 값이 길게 설정 되어야 하며, 내부 웹 서버에 접속할 경우에는 RTO 값이 짧게 설정되어야 하므로 기존 알고리즘이 그대로 적용된다. 그러나 시뮬레이션 모델에서는 내부접속용 서버는 동일 LAN에 위치하는 클라이언트의 접속만을 받아들이므로 RTO 값이 길게 설정될 필요가 없다. 따라서 시험을 통해 LAN 이용률에 적합한 RTO 값을 얻고자 한다.

그림 1은 본 논문에서 시도한 시뮬레이션의 네트워크 토폴로지(topology)이다. 동일한 LAN 세그먼트에 7개의 노드가 위치하고 있고 7번 노드가 내부 접속용 서버이고, 9번 노드가 외부 접속용 서버이다. 10번 노드는 WAN과의 연결을 의미하는데 10번 노드를 통해 들어오는 패킷은 라우터인 8번 노드를 경유하여 외부 접속용 서버인 9번 노드로 연결이 된다.

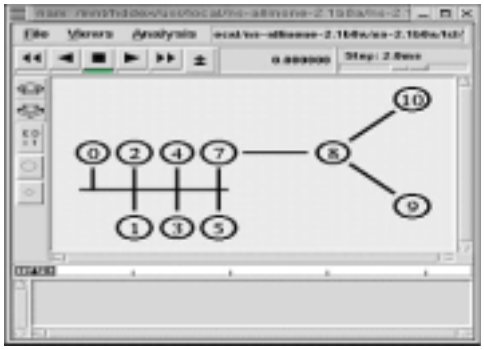

그림 1. 네트워크 토폴로지
Fig. 1 Network topology

3-2. 성능 분석

내부 접속용 서버에 테스트 간격을 0.5sec로 하여 RTO 값을 각각 0.05, 0.1, 0.2 10 sec로 설정한 후 이더넷 이용률에 따라 비교 분석하였다. 예외로 0.2sec 값의 설정연구에서 RTO의 범위를 0.2 ~ 64sec[11]로 정의하고 있는 것을 고려하여 추가한 것이다. 또한 본 논문의 시험 환경이 되었던 시뮬레이션 코드의 알고리즘을 보면 최소 최대값의 범위가 적용되는 동시에 최소값이 0.2sec 이하로 설정될 경우에는 RTO값을 0.2sec로 재설정되도록 하기 위한 것이다.

일반적으로 이더넷의 이용률이 36~38%가 되면 네트워크 재설계를 고려해 보아야 한다[3]. 따라서 본 연구에서는 이더넷 이용률을 10%에서 최대 40%까지로 하여 시험해 보았다.

그림 1에서 그림 5에 있는 Default는 TCP 코드에서 파라미터 값을 수정하기 전의 결과로 재전송 시 적용되는 최소, 최대 값의 범위가 1-100sec로 주어졌을 때이다.

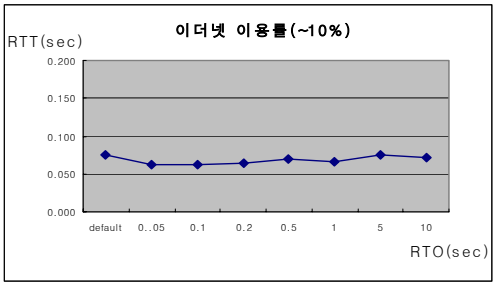

그림 2. RTO에 따른 RTT 변화
Fig. 2 Change of RTT

그림 2와 그림 3은 이더넷 이용률이 10% 이하와 20%이하인 경우로 패킷 손실로 인한 재전송이 이루어지지 않는 경우이다. 재전송 타임아웃에 관한 알고리즘이 그대로 적용되었을 경우와 재전송 타임아웃을 각각 0.05sec부터 10sec까지 고정시켜 시험한 결과이다. 실제 시험은 RTO 값을 0.001sec과 0.01sec일 경우도 해 보았으나 지나치게 빠른 재전송으로 인하여 같은 조건에서 80% 이상의 패킷 손실을 가져왔다. 따라서 테스트 범위의 최소값을 0.05sec로 한 것이다.

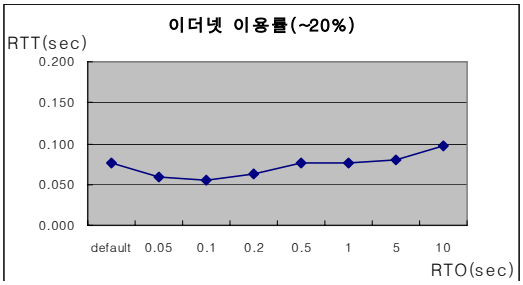

그림 3. RTO에 따른 RTT 변화
Fig. 3 Change of RTT

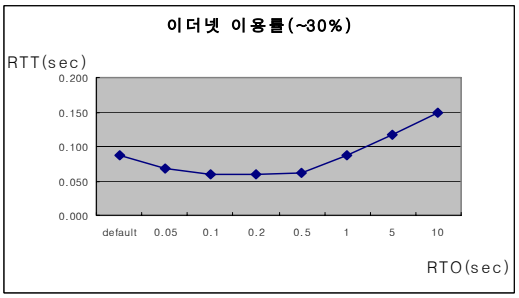

그림 4. RTO에 따른 RTT 변화
Fig. 4 Change of RTT

그림 4는 이더넷 이용률이 30% 가까이 되면서 패킷 손실로 인한 재전송과 그에 따른 RTT 값을 보여주는 예이다. RTO 값의 설정에 따라 RTT가 다르게 나타나는 것을 확인할 수 있다.

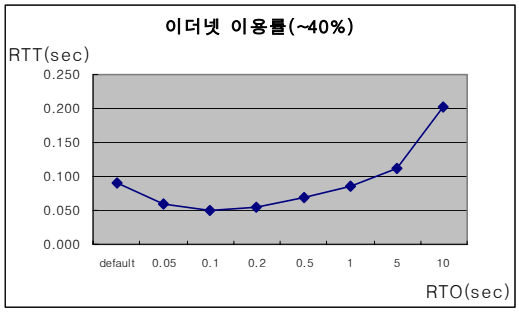

그림 5. RTO에 따른 RTT 변화
Fig. 5 Change of RTT

그림 5에서도 그림 4와 비슷한 결과를 확인할 수 있다. RTO 값이 0.05sec로 설정되었을 경우에는 0.1~0.2sec로 설정되었을 때보다 RTT가 더 길게 나타난다. 이는 RTO가 지나치게 짧게 설정되어 잦은 재전송을 유발시켜 오히려 트래픽을 가중시킴으로써 더욱 긴 RTT 값을 얻게 된다는 것을 의미한다. 반면에 5sec, 10sec의 결과를 보면 잦은 재전송으로 인한 트래픽의 과부하는 줄어들어 상대적으로 패킷 손실률은 줄었으나 RTO가 지나치게 길게 설정되어 오히려 RTT값이 더욱 길어진 것을 확인할 수 있다

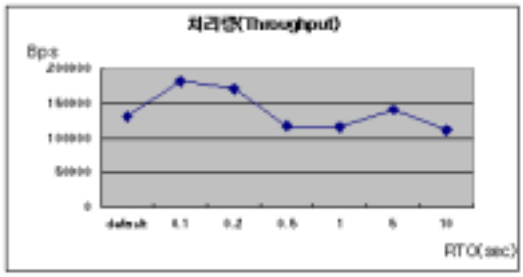

그림 6. RTO에 따른 처리량
Fig. 6. Change of throughput

그림 6은 처리량을 나타낸다. 이더넷 이용률에서 볼 수 있었듯이 RTO 값이 0.1 ~ 0.2sec로 설정돼 있을 때 처리량 역시 가장 높게 나타남을 알 수 있다. 이는 빠른 응답시간으로 인하여 주어진 시간동안에 더 많은 처리를 한 것을 의미한다.

Ⅳ.결론

본 논문에서는 외부에 비해 내부 접속률이 많은 대학과 상황을 고려하여 서버를 외부용과 내부용으로 나누었다. 그리고 각각의 서버에 적합한 TCP 파라미터 값을 구하려고 시도하였다.

내부 접속용 서버는 트래픽 과부하로 인한 패킷 손실로 재전송이 이루어질 경우 기존 RTO 알고리즘을 통한 재전송보다 RTO 값을 0.1~0.2로 고정시키는 것이 더 빠른 응답시간 즉, RTT 값은 작고 처리량은 많게 나타나는 것을 확인할 수 있었다. 이를 통해 이더넷 이용률에 따른 RTO 값 조정을 통해 내부 웹 서버의 성능 향상을 가져올 수 있다는 결론을 얻을 수 있었다.

외부 접속용 서버의 경우 인터넷을 시뮬레이션하기에 문제점이 있어, 본 논문에서와 같은 실제 환경을 구축하여 웹 서버에 설정된 RTO 값을 조정하여 실험해 보았다. 실제 환경에서는 균일한 트래픽 상황에서의 관찰이 불가능하였지만 default로 설정된 값보다 RTO 값이 커졌을 때 성능이 높아짐을 확인하였다.

참 고 문 헌

[1] Larry L.Peterson, Bruce S. Davie, Computer Networks : Computer Networks: A Systems Approach, Morgan Kaufmann, 1999.
[2] Douglas E. Comer, Computer Networks and Internets, Prentice-Hall, 2002.
[3] Priscilla Oppenheimer, Top-Down Network Design, Cisco Systems, 2003.
[4] William Stallings, Data & Computer Communications, 2001.
[5] Miller, 데이터통신과 네트워크통신, 사이텍미디어, 2001.
[6] 권경희, 웹 엔지니어링, 배움터, 2001.
[7] Beck, Harold Bohme, Linus Torvalds, LINUX KERNEL INTERNALS, Addison Wesley, 1996.
[8] Craig Hunt, TCP/IP Network administration, Addison Wesley, 2002.
[9] W. Reichard Stevens, TCP/IP Illustrated Volume1, Addison Wesley, pp 223-356, 491-498, 525-538, 2000.
[10] Gary R. Wright W. Reichard Stevens, TCP/IP Illustrated Volume2(1), Addison Wesley, 2000.
[11] W.Reichard Stevens, TCP/IP Illustrated

Volume2(2), Addison Wesley, pp 817-848, 2000.
[12] W.Reichard stevens, TCP/IP Illustrated Volume3, Addison Wesley, pp 91-158, 2000.
[13] Mark E. Crovella, Robert Frangioso and Mor Harchol-Balter, "Connection Scheduling in Web Servers", Technical Report CS 99-003, Boston University, 1999.
[14] 김은기, "TCP 혼잡제어를 위한 RTT 측정", 한국정보처리학회 논문지, VOL. 07 NO. 05 , pp 1520-1524, 2000. 05.
[15] 정진국외 3명, "다중 처리기 기반 웹 서버 구조의 실험적 성능 분석", 정보과학회논문지, VOL. 28 NO. 01, pp 0022- 0036, 2001. 03.
[16] M.F. Arlitt and C.L. Wiliamson, "Web Server Workload Characterization: The Search for Invariants", Proceeding of SIGMETRICS96, 1996.
http://www.infoage.co.kr/lantimes/9804/lecture.html
[17] 전철완, "윈도우 사이즈 조절을 통한 내부 네트워크의 성능향상", 정보처리학회 춘계학술대회 VOL. 10 NO. 01, pp 1189 -1192, 2003.
[18] 김성수, 정지영, "웹 서버 클러스터 상에서 문서 접근 확률과 문서 크기를 이용한 부하 분배□□, 정보과학회 추계학술대회 VOL. 27 NO. 02, pp 0452-0454, 2000. 10.
[19] Konstantinos Michael Pentikousis, "Error Medeling for TCP Performance Evaluation", Master Thesis, State University of New York at Stony Brook, 2000.
[20] TCP congestion control mechanism
<http://www.netmanias.com/sub/sub55/tcp.htm>
[21] Chadi Barakat, Eitan Altman, and Walid Dabbous, "On TCP Performance in a Heterogeneous Network", IEEE Communications Magazine, pp 40-46, January. 2000.
[22] 김진희, 권경희, "재전송 타임아웃 간격의 범위 조절에 의한 Web 서버의 성능향상", 정보처리학회 추계학술대회 VOL. 09 NO. 02, pp 1193-1196, 2002. 10

저 자 소 개

金鎭喜 (學生會員)

1999년 방송통신대학교 전자계산학과
2001년 단국대학교 전자계산학과 컴퓨터과학(이학석사)
2002년~단국대학교 전자계산학과 컴퓨터과학(박사과정중)
관심분야 : 컴퓨터 네트워크, TCP/IP

權景熙 (正會員)

1976년 고려대학교 물리학과(이학사)
1986년 Old Dominion Univ. Dept. of Computer Science(M.S.)
1992년 Louisiana State Univ Dept. of Computer Science(Ph.D.)
1979년 ~ 1984년 산업연구원 연구원
1993년~현재 단국대학교 부교수
관심분야 : 컴퓨터 네트워크, 알고리즘 분석 및 설계, 웹 공학