패킷 손실률에 기반한 효율적인 TCP Buffer Tuning 알고리즘

류 기 철† · 김 동 균††

요 약

기존 TCP 기술은 송·수신측에 각각 고정된 크기의 버퍼를 할당하기 때문에 높은 대역폭(High-Bandwidth) 및 큰 전송지연(High Delay)을 가진 통신에는 적합하지 못하다. 따라서 종단간의 TCP 처리량을 개선하기 위해 통신망 상황에 따라 자동으로 TCP 버퍼를 조절하려는 시도가 있어왔다. ATBT(Automatic TCP Buffer Tuning)에서 송신측은 현재의 혼잡 제어 윈도우(CWND)의 값에 따라 송신 버퍼 크기를 조절하고 수신측은 운영체제가 정해둔 최대 크기의 TCP 버퍼 값으로 수신 버퍼 크기를 고정한다. DRS(Dynamic Right Sizing) 에서는 이전에 수신한 TCP 데이터의 두 배를 현재 송신할 TCP 데이터라고 예측함으써, TCP 수신측은 단순히 이에 따라 수신 버퍼 크기를 동적으로 변화시킨다. 그렇지만, TCP 세그먼트의 손실 가능성으로 인해 정확히 두 배로 버퍼 크기를 변화시킬 필요는 없다. 따라서 우리가 제안한 패킷 손실률에 기반한 효율적인 TCP 버퍼 조절 알고리즘(TBT-PLR:TCP Buffer Tuning Algorithm based on Packet Loss Ratio)은 TCP 송신측에는 ATBT 방법을 TCP 수신측에는 TBT-PLR 방법을 적용하였다. 실제 TCP 성능을 테스트하기 위해서 리눅스 커널 2.4.18을 수정하여 구현하였으며 기존의 고정된 크기의 TCP 버퍼를 가진 경우와 버퍼 크기가 동적으로 변하는 TBT-PLR을 적용한 경우를 비교하였다. 결과적으로, TCP 연결들간의 균형있는 메모리 사용으로 인해 성능 향상을 얻을 수 있었다.

An Efficient TCP Buffer Tuning Algorithm based on Packet Loss Ratio(TBT-PLR)

Gi-Chul Yoo† · Dong-kyun Kim††

ABSTRACT

The existing TCP(Transmission Control Protocol) is known to be unsuitable for a network with the characteristics of high BDP(Bandwidth-Delay Product) because of the fixed small or large buffer size at the TCP sender and receiver. Thus, some trial cases of adjusting the buffer sizes automatically with respect to network condition have been proposed to improve the end-to-end TCP throughput. ATBT(Automatic TCP Buffer Tuning) attempts to assure the buffer size of TCP sender according to its current congestion window size but the ATBT assumes that the buffer size of TCP receiver is maximum value that operating system defines. In DRS(Dynamic Right Sizing), by estimating the TCP arrival data of two times the amount TCP data received previously, the TCP receiver simply reserves the buffer size for the next arrival, accordingly. However, we do not need to reserve exactly two times of buffer size because of the possibility of TCP segment loss. We propose an efficient TCP buffer tuning technique(called TBT-PLR: TCP buffer tuning algorithm based on packet loss ratio) since we adopt the ATBT mechanism and the TBT-PLR mechanism for the TCP sender and the TCP receiver, respectively. For the purpose of testing the actual TCP performance, we implemented our TBT-PLR by modifying the linux kernel version 2.4.18 and evaluated the TCP performance by comparing TBT-PLR with the TCP schemes of the fixed buffer size. As a result, more balanced usage among TCP connections was obtained.

키워드 : TCP, 버퍼 할당(Buffer Tuning), 패킷 손실(Packet Loss), ATBT, DRS, SABT, TBT-PLR

1. 서 론

BDP(Bandwidth-delay Product)가 높은 GRID와 같은 고속 데이터 통신망의 등장에도 불구하고 종단간 데이터 전송의 성능향상을 얻을 수가 없어서 응용프로그램, 운영체제, 송·수신 측의 디스크 혹은 네트워크 어댑터(Adaptor), 네트워크 스위치와 라우터 등에서의 문제점을 파악하고 그 해결책을 얻으려는 노력을 많이 기울여 왔다. 하지만 성능 향상을 얻지 못하는 원인이 불분명하여 최근에는 응용프로그램과 통신 서비스에 가장 밀접한 관련이 있는 TCP 버퍼 부분에 대한 연구가 진행되고 있다. TCP 성능향상을 위한 노력으로는 TCP 제어 알고리즘을 수정하는 방법과 TCP 버퍼를 조절하는 방법으로 크게 나눌 수 있다.

TCP 제어 알고리즘을 수정하는 방법은 TCP 혼잡제어(Congestion control) 방법 혹은 흐름제어(Flow control)

† 준 회 원 : LG전자 DM연구소 연구원
†† 정 회 원 : 경북대학교 컴퓨터공학과 전임강사
논문접수 : 2004년 6월 19일, 심사완료 : 2004년 9월 13일

방법을 수정하여 실제 종단간 TCP 성능향상을 얻고자 한다. 예를 들면 슬로우 스타트 시 세그먼트 전송 수를 처음부터 크게 늘려 데이터 전송을 하거나, 혼잡회피(Congestion avoidance) 단계에서 선형적으로 CWND(Congestion window size)를 증가시키지 않고 증가량을 달리하여 많은 세그먼트를 전송할 수 있게 한다. 그리고 임계값(Threshold) 등을 조절하여 슬로우 스타트 단계에서 지수적으로 증가하는 CWND의 한계 값을 높게 설정해 더 많은 양의 데이터 전송을 가능하게 한다. 하지만 이 방법은 TCP 제어 알고리즘 자체에 대한 복잡한 수정이 필요하여 모든 시스템의 커널 수준의 수정뿐만 아니라 통신망에 위치한 다른 시스템에 탑재되어 있는 다른 TCP와의 혼잡제어 방법간의 공평성(Fairness) 문제도 더 심도 있게 고려해야 한다.

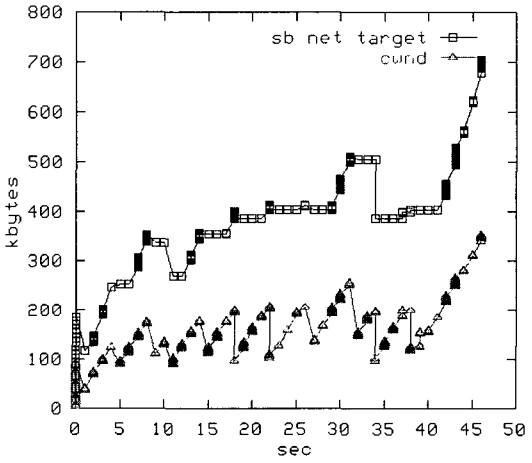
TCP 버퍼를 조절하는 방법은 TCP 제어 알고리즘을 수정하지 않고 기존의 TCP 기술을 그대로 수용한다. 하지만 실제 TCP를 구동하는데 있어 변경 가능한 파라메터를 네트워크 상황에 맞게 조절함으로써 종단간 성능향상을 얻고자 한다. 사실상 BDP가 큼에도 불구하고 송신 측에서 전송하는 데이터 양이 적어서 종단간 성능향상을 얻기가 힘들다. 그 해결책으로 단순하게 디폴트 버퍼 크기를 크게 했을 경우, 단일 시스템 위에서 TCP를 이용하고 있는 응용프로그램 수가 많다고 할 때 결국 버퍼에 대한 경쟁이 생겨 이를 해결해 줄 수 있는 방법이 필요하며 TCP 연결설정 수락을 시킬 수 있는 응용프로그램의 수를 늘려 주는 게 성능향상 측도로 사용될 수 있다. 물론 TCP 연결설정에 앞서 통신망의 상황을 검사(예, ping과 같은 프로그램 사용)하여 최적의 버퍼 크기를 구할 수 있겠지만 통신망 상황이 시간에 따라 계속 변하는 상황에서 연결 설정시에 고정시킨다는 것은 무리가 따르기 때문에 자동화된 버퍼 할당 기술이 필요하다.

2. 관련 연구

2.1 ATBT(Automatic TCP Buffer Tuning)

TCP 버퍼를 조절하는 방법은 크게 Hand tuning 방법과 Auto tuning 방법으로 나눌 수 있다. 먼저 기존의 Hand tuning 방법은 TCP 연결설정 수락을 하기 전에 통신망의 상황을 검사하여 송·수신 측 버퍼 크기를 결정한다. 통신망 상황은 시간에 따라 계속 변하는데 비해 TCP 버퍼 크기는 고정되어 있다는 문제점이 있다. 그리고, Hand tuning 방법을 사용하기 위해 전문가나 시스템 운영자 정도의 전문적인 지식과 능력이 요구된다는 문제점도 있다. 따라서 Hand tuning 방법에서 발생하는 이러한 문제를 해결하고자 Auto tuning 방법이 제안되었다. Auto tuning 방법은 각 TCP 연결마다 어떠한 환경에라도 최상의 수행 능력을 줄 수 있고, 더욱 효과적으로 시스템 메모리를 사용하면서 메모리 낭비를 줄이고자 한다. 대표적인 Auto Tuning 방법인 ATBT 방법은 동적으로 변화하는 네트워크 상황과 시스템 메모리의 유효성을 기반으로 동작한다. ATBT는 크게 송신측과 수신측으로 나눠서 생각할 수 있다. 수신측 버퍼 크기는 운영체제의 설정된 최대치로 결정된다. 반면에 송신측 버퍼 크기는 네트워크의 혼잡 상황과 메모리 할당의 공정성을 고려하여 동적으로 변화시킨다. 이를 위해 송신측에서는 네트워크에 기반한 버퍼 할당 방법과 공정한 메모리 공유 방법을 적용한다.

먼저, 네트워크에 기반한 버퍼 할당 방법에서 송신 버퍼는 적당한 혼잡제어 윈도우 크기를 산정하기 위한 CWND 값을 사용한다. 그리고 sb_net_target이라는 변수는 현재의 송신 버퍼 크기를 나타낸다. ATBT는 CWND 값에 영향을 받는데 만약 CWND가 (sb_net_target/2)보다 큰 경우에는 sb_net_target을 증가시키게 되고 CWND가 (sb_net_target/4)보다 작을 경우에는 감소시킨다. 결과적으로, 현재의 송신 버퍼 크기를 나타내는 변수인 sb_net_target는 CWND의 2배와 4배 사이에 존재하는 것을 (그림 1)에서 볼 수 있다.

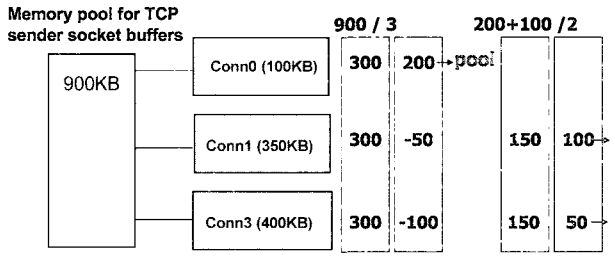


(그림 1) sb_net_target과 CWND의 over time

(그림 1)에서와 같이 ATBT에서는 송신 버퍼 크기를 sb_net_target의 크기에 맞게 조절하여 성능 향상을 나타내고자 한다. 따라서 ATBT의 송신측은 응용프로그램으로부터 송신할 데이터를 미리 송신버퍼로 옮겨와서 수신 측에서 데이터를 요구할 때 보다 빨리 데이터를 전송할 수 있다는 장점을 가지고 있다[1].

다음으로, ATBT 송신측에서 제공하는 공정한 메모리 공유 방법은 각 TCP 연결에 할당되는 송신측 버퍼 크기가 현재 전송되는 CWND의 값에 따라 결정된다. 모든 TCP 연결들의 전체 요구되는 버퍼 크기가 송신 버퍼 크기보다 클 경우에는 송신 버퍼는 각 TCP 연결에 최대-최소 공평성 정책(MAX-MIN Fairness Policy)에 따라 할당이 된다. 우선적으로 동등한 크기의 버퍼 공간을 각 TCP

연결마다 공평하게 할당하는데 이를 hiwat_fair_share 라는 변수로 나타낸다. sb_net_target이 hiwat_fair_share 보다 작을 경우에는 (큰 공간의 버퍼를 요구하지 않는 연결이 존재할 경우) 사용되지 않는 버퍼 공간을 "Shared Pool"에 넘겨주게 된다. 반대로, sb_net_target 이 hiwat_fair_share 보다 큰 경우에는 이 "Shared Pool"에 넘겨진 버퍼 공간을 할당된 크기보다 더 큰 버퍼 공간을 요구하는 연결들에게 동등하게 재할당해 주는 방법을 제공한다. 이러한 방법을 tcp_slowtime()에 의해 주기적으로 계산된다.



(그림 2) ATBT의 Fair Share Algorithm

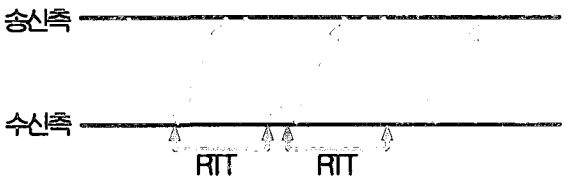
예를 들어, (그림 2)에서 세 개의 TCP 연결 (Conn0, Conn1, Conn2)이 활성화되어 있고 각 연결이 요구하는 버퍼 크기가 100, 350, 400이며 송신버퍼 크기가 900이라고 할 때 각 연결에 할당되는 버퍼 크기는 300이 된다. 그럴 경우, Conn0는 잔여량 200을 "Shared Pool"에 보내게 되고, Conn1과 Conn2처럼 할당된 크기보다 더 많이 요구하는 연결들은 "Shared Pool"의 부분을 공평하게 재할당해 준다. ATBT에서 송신 TCP가 갑자기 CWND 값을 변경할 경우 할당된 버퍼 크기는 때때로 연결이 실제로 요구하는 양보다 더 작은 양으로 설정될 수 있으며 버퍼 크기 할당이 너무 자주 변경될 수 있는 문제점을 가지고 있다. 또한 통신망 대역폭이 클 때 CWND의 값이 심하게 진동하는 현상(Oscillation)을 초래할 수 있다.

2.2 DRS(Dynamic Right Sizing)

TCP는 인터넷에서 유비쿼터스 전송 프로토콜로서의 입지를 굳히고 있을 뿐만 아니라 계산적인 그리드나 데이터 그리드, 엑세스 그리드와 같은 인프라 스트럭처에서 중요한 이슈가 된다. 그러나 고성능의 계산적인 그리드와 그 외 대역폭과 관련된 응용에서 TCP와 관련한 흐름 제어와 혼잡 제어문제가 있다. 이를 해결하기 위해 관련 연구가들은 수동적으로 최적의 버퍼 사이즈를 유지하려는 노력을 해왔다. 그래서 그리드 컴퓨팅 환경 아래의 WAN (Wide-area Network)에서 어느 정도의 성능 향상은 얻을 수 있었다. 하지만 이러한 Tuning 프로세스가 주어진 하나의 연결마다 RTT(Round-trip Time)와 bottleneck link의 대역폭을 계산해야 하므로 사용자나 보통의 개발자가 이용하기에는 어렵다는 문제점이 있다. 그리고 TCP에서는 EWND/RTT의 비율로 데이터를 전송한다. 여기서 EWND(Effective window size)는 FWND(Flow window size)와 CWND(Congestion window size)의 최소값으로 결정된다. 현재에는 CWND가 네트워크 상황의 변화에 따라 동적으로 변화하는데 비해 FWND는 고정적인 경향을 보인다. 대부분의 운영체제(Operating System)는 FWND≈64KB로 설정한다. 그러나 BDP의 범위가 몇 바이트(56Kbps × 5ms = 36bytes)부터 몇 메가바이트(622Mbps × 100ms = 7.8MB)라고 할 때 시스템은 몇 바이트인 경우에는 할당된 메모리의 99%(36B/64KB=0.05%)를 낭비하게 되고 몇 메가바이트인 경우에는 네트워크 대역폭의 99%(64KB/7.8MB=0.8%)를 낭비하게 된다. 전송 윈도우(EWND)는 혼잡 제어 윈도우(CWND)에 제한을 받는 것이 이상적이다. FWND ≥ CWND 경우는 네트워크 이용률은 최대가 되고 FWND == CWND인 경우는 메모리 사용률이 가장 효율적이다. 따라서 네트워크 대역폭이나 메모리 자원의 낭비를 줄여 성능을 향상시키기 위해 동적으로 FWND를 변화하여야 한다[2].

따라서 DRS는 TCP 수신측 버퍼 크기를 자동으로 조절하고자 한다. 다시 말해 DRS는 수신측에서 송신측 CWND 크기를 예측하여 측정치에 따라 동적으로 수신측의 윈도우 알림(Window advertisement) 크기를 변화시키는데 연결이 설정되는 순간만 동적으로 버퍼를 조절하는 것이 아니라 연결의 생명주기(life-time) 동안 계속적으로 변화한다. 여기서 수신측의 윈도우 알림 크기는 수신측의 남아있는 버퍼 크기를 나타낸다[5].

DRS에서 수신측의 윈도우 알림 크기는 수신측의 가용한 대역폭과 RTT에 의해 계산된다. RTT는 원래 송신측이 데이터를 보내고 그 데이터에 대한 ACK를 수신한 시간으로 계산하지만 수신 측에서는 네트워크에서 패킷을 보내고 ACK를 받는 동작의 트래픽을 조사해 보지 않고는 알 수 없다. 즉, 수신측 RTT를 측정하기 위해서 (그림 3)과 같이 수신측에서 하나의 패킷에 대한 ACK를 보낸 후에 그 ACK에 대해 데이터가 도착한 시간으로 결정한다.



(그림 3) 수신측 RTT 예측

수신측의 평균 가용 대역폭은 수신측 RTT 시간 동안 수신된 바이트 수로 구할 수 있다. 하지만 평균 가용 대역폭을 구하는 샘플링 간격이 짧으면 오버헤드가 증가하고 성능이 저하되어 스케줄링과 버퍼링의 영향으로 정확하지 않은 예측이 발생할 수 있다. 반면에 샘플링 간격이

너무 길면 가용한 대역폭의 변화에 따른 응답이 늦어진다. 따라서 충분한 정확도를 가질 수 있는 샘플링 간격을 적절히 선택해야 하는 문제가 있다. 평균 가용 대역폭과 수신측 RTT의 곱을 BDP(Bandwidth Delay Product)의 값으로 결정한다. 따라서 슬로우 스타트 동안에 수신측 윈도우 알림의 크기는 2배의 BDP가 되고 혼잡 회피 구간에서는 선형적으로 증가하게 된다. DRS의 성능은 평균 가용 대역폭과 수신측 RTT에 대한 정확한 예측에 기반한다[3].

2.3 SABT(Scalable Automatic Buffer Tuning)

SABT는 TCP 송신 버퍼 크기를 자동으로 조절하고자 하며 특히 송신 호스트가 많은 TCP 연결설정 요구를 동시에 수락했을 경우 송신 버퍼 크기를 각 연결의 네트워크 상황에 따라 설정한다. ATBT에서는 각 TCP 연결에 할당되는 버퍼 크기가 현재 사용되는 CWND의 값에 따라 결정된다. 모든 TCP 연결들의 전체 요구되는 버퍼 크기가 송신 버퍼 크기보다 클 경우 송신 버퍼는 각 TCP 연결에 최대-최소 공평성 정책(MAX-MIN Fairness Policy)에 따라 할당이 된다. 즉, 우선적으로 버퍼를 동등한 크기로 모든 TCP 연결에 공평하게 할당하며 큰 버퍼를 요구하지 않는 연결이 존재할 경우에 사용되지 않는 부분을 더 큰 버퍼 공간을 요구하는 연결에게 재할당해 주는 과정을 거친다[4].

만약 ATBT에서 송신 TCP의 CWND 값이 변경된 경우 할당된 버퍼 크기는 때때로 연결이 실제로 요구하는 양보다 더 작은 양으로 설정될 수 있으며 버퍼 크기 할당이 너무 자주 변경되는 문제점을 가지고 있다. 또한 통신망 대역폭이 클 때 CWND의 값이 심하게 진동하는 현상(Oscillation)이 크게 일어나 결국 할당된 버퍼 크기도 역시 Oscillation을 초래할 수 있다. 반면 SABT는 잔여 버퍼 크기를 연결별로 요구되는 양이 다를 경우 요구량에 비례해서 재 할당한다. 예를 들어 세 개의 TCP 연결 (T1, T2, T3)이 활성화되어 있고 각 연결마다 CWND로부터 계산된 버퍼 크기가 20, 200, 800이라 가정하고 송신 버퍼의 전체 크기가 300이라 가정하면 송신 호스트는 먼저 100씩 각 연결에 할당할 것이며 T1이 100만큼을 요구하지 않기 때문에 80에 해당하는 잔여 버퍼를 T2와 T3에 공평하게 40씩 할당하여 결국 T2와 T3 각 모두 140씩 할당 받게 되어 있다. 하지만 실제 잔여 버퍼를 할당함에 있어 요구되는 양에 비례해서 할당해야 한다는 문제가 발생하게 된다. SABT는 그 요구되는 양에 따라 재할당하는 방법을 제공한다. 따라서 각 TCP 연결이 실제 요구하는 버퍼크기를 정확히 예측하는 것이 중요하며 SABT에서는 예측된 처리율에 따라 버퍼를 할당한다.

송신 호스트는 먼저 세 가지 파라메터(p-RED 알고리즘이 패킷 패기 확률, rtt-평균 RTT 값, rto-평균 재전송 타임아웃 값)로부터 각 연결마다 기대되는 TCP 처리율을 예측한다. ATBT와 달리 CWND 값으로부터 요구되는 버퍼 크기를 계산하는 것이 아니라 예측된 TCP 처리율 식 (1)로부터 버퍼 크기를 계산한다.

$$\lambda = \max\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right)}\right) \quad (1)$$

(Wmax는 수신측의 버퍼 크기, b는 Delayed ACK 사용여부에 따른 값 변화)

따라서, SABT에서는 TCP 연결 i의 예측된 처리율을 Pi로 표기했을 때 요구되는 버퍼 크기 Bi는 Pi × RTTi로 나타낼 수 있으며 앞서 설명한 대로 이러한 계산된 버퍼 요구에 따라 최대-최소 공평성 정책(MAX-MIN Fairness Policy)을 사용해서 버퍼를 할당한다.

3. 패킷 손실률에 기반한 TCP Buffer Tuning 알고리즘 (TBT-PLR)

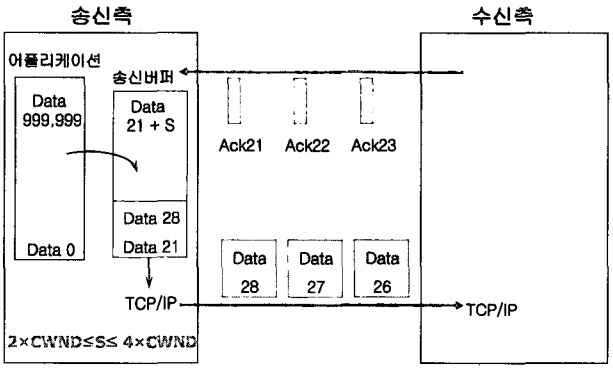
실제 네트워크에서는 많은 TCP 연결이 발생하여 각 연결마다 필요로 하는 버퍼를 할당 받기 위해 경쟁을 해야 하는 상황이 발생한다. 제한된 수신 버퍼를 공정하게 할당함으로써 성능 향상을 얻고자 패킷 손실률에 기반한 TCP 버퍼 조절 알고리즘인 TBT-PLR을 제안한다. TBT-PLR은 결국 TCP 송신측과 수신측에서 제한된 버퍼를 효율적으로 사용하도록 한다.

3.1 송신측 버퍼 할당 메카니즘

TCP 송신측은 각 연결마다 최대 전송률을 얻기 위해 ATBT의 버퍼 할당 방법을 사용해 동적으로 송신측 버퍼 크기를 변화시킨다. (그림 4)에서 보듯이, 현재 TCP 송신측의 CWND가 8이라고 한다면 순차번호 21~28까지의 세그멘트들이 송신 측으로 보내진다. 이때 TCP 송신측은 추가적으로 송신측 버퍼 크기를 S만큼 증가시키는데 S의 값은 3×8 세그멘트로 결정함으로써 $2\times CWND \le S \le 4\times CWND$에 존재하게 된다. TCP 송신측은 다음 전송을 시작하기 전에 먼저 어플리케이션으로부터 세그멘트 29~52까지를 할당된 송신측 버퍼로 가져온다.

만약 세그멘트 21~28이 패킷 손실이 없이 수신측에 전달이 되고 TCP 수신측으로부터 ACK 패킷을 송신측이 받게 된다면 이전의 CWND 값의 두 배인 16에 해당하는 세그멘트 29~44를 슬로우 스타트 동안에 전송하게 된다. 다시 말하면, TCP 송신측은 송신측 버퍼 크기를 2×CWND와 4×CWND 사이를 유지하도록 한다. CWND는 각 TCP 연결마다 송신측 버퍼 크기를 결정하는 요소로 사용되고 이전 CWND의 두 배를 먼저 할당한다. 따라서 TCP 송신

측은 이미 보내진 세그멘트에 대한 ACK가 오기 전에 다음 전송을 위한 충분한 크기의 버퍼를 할당받게 되고 각 연결에 할당하고 남은 버퍼 공간은 TCP 연결의 수만큼 동등하게 나눠 재할당을 하게 된다.



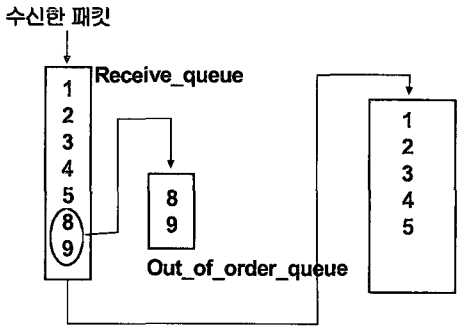
(그림 4) ATBT 송신 측 Buffer Tuning

3.2 수신측 버퍼할당 메카니즘

TCP 수신측은 패킷 손실률에 기반한 효율적인 버퍼 조절 알고리즘인 TBT-PLR을 적용한다. TCP 연결의 네트워크 상황에 따라 할당할 수신 버퍼 크기를 다르게 함으로써 많은 연결이 발생하는 경우에도 연결마다 공평하게 버퍼를 할당하였다. ATBT는 가용한 연결의 수나 수신측 버퍼에 관해 고려하지 않고 단지 TCP 수신측은 운영체제에서 정의한 최대크기의 고정된 버퍼를 가진다. 반면에 DRS는 수신측이 송신측의 CWND 크기를 예측하게 하고 수신측의 윈도우 알림(window advertisement)의 크기인 FWND를 동적으로 변화시키는데 사용한다. 연결이 설정되는 순간만 동적으로 버퍼를 조절하는 것이 아니라 연결의 생명주기(life-time) 동안에 계속적으로 변화한다. 그렇지만 DRS는 수신측에 많은 TCP 연결이 발생해 제한된 버퍼크기를 할당받기 위해 경쟁하는 상황을 고려하지 않아서 높은 전송률의 연결이 먼저 성립된 경우에는 수신측 버퍼를 독점하는 문제가 발생하게 된다.

기존의 방법인 SABT에서는 연결마다 공정한 버퍼 사용을 위해 할당할 버퍼 크기를 결정할 때 네트워크 상황과 관련한 세 가지 파라메터(p-RED 알고리즘이 패킷 패기 확률, rtt-평균 RTT 값, rto-평균 재전송 타임아웃 값)를 이용해 복잡한 계산과정을 거친다. 하지만, SABT는 수신측을 고려하지 않고 송신측만을 대상으로 하였으며 복잡한 계산 과정을 통해 처리율을 예측한다. 그리고 정확한 값을 얻기 위해 많은 비용을 치러야 한다.

따라서 수신측에서 제한된 버퍼를 가능한 많은 연결들이 효율적으로 사용할 수 있도록 각 연결마다 패킷 손실률에 기반하여 버퍼를 조절하는 TBT-PLR 알고리즘을 제안하였다. 패킷 손실이 발생하면 TCP 송신측은 타임아웃(time-out)이나 빠른 재전송(fast retransmit) 알고리즘으로 인해 CWND 크기를 줄이므로, TCP 수신측에서 전송할 세그멘트의 수가 감소하는 상황을 고려해 필요 이상으로 버퍼가 할당되는 상황을 막아 또 다른 TCP 연결이 성립되었을 때 할당할 수 있도록 한다. 특히 실제 네트워크 상황을 예측하는데 있어 보다 단순한 방법으로 구현을 하고자 하는데 초점을 맞추었다.



(그림 5) Receive_Queue와 Out_of_Order_Queue

(그림 5)에서 보듯이, TCP 송신측에서 세그멘트 1~9를 TCP 수신측으로 전송했지만 세그멘트 6, 7은 손실이 발생해 결국 Receive_queue에는 세그멘트 6, 7을 제외한 세그멘트 1~9가 도착했다. 이때 세그멘트 8, 9는 수신 버퍼를 통해 상위계층으로 전달되기 전에 out_of_order_queue에서 세그멘트 6, 7이 재전송되기를 기다리게 된다. 따라서 TCP 송신측에 있는 out_of_order_queue를 이용해 네트워크 상황을 예측할 수 있다. 리눅스 커널에 사용된 구현을 보면, 수신된 패킷은 receive_queue에 위치하게 되고 이때 만약 비순차적인(out of order) 패킷이 존재한다면 out_of_order_queue로 옮겨진다. 만약 out_of _order_queue에 패킷들이 많이 존재한다면 네트워크 트래픽이 매우 혼잡한 상황이라 볼 수 있다. 따라서, TCP 송신측은 보다 많은 데이터를 전송하지 못하므로 작은 크기의 TCP 수신 버퍼를 할당하는 것이 합당하다. 반대로 비순차적인 패킷이 작다면 TCP 수신측에 큰 버퍼 공간을 할당한다.

$$\alpha = (rcv\_Q - out\_Q) / rcv\_Q \qquad (2)$$

식 (2)는 receive_queue와 out_of_order_queue의 크기를 이용해 최적의 버퍼 크기를 결정하는 요소인 α를 계산한다. 여기서, rcv_Q와 out_Q는 receive_queue와 out_of_order_queue의 크기를 각각 나타낸다. 따라서 α는 수신한 패킷 중에 상위 계층이나 어플리케이션으로 전달하게 될 총 패킷의 비율이라 할 수 있다.

수신된 패킷 가운데 out_of_order_queue로 옮겨질 패킷이 지속적으로 존재하기 때문에 제안한 방법은 다음 패킷의 수신을 위해 TCP 수신측 버퍼공간을 단순히 두 배로 결정하지 않고 α의 값에 따라 결정한다.

```
INITIAL STATE :
    rcvmem = current receiver's default window size;
    receive buffer size = 3× rcvmem
CHANGE STATE :
if the remaining size of receive buffer  ≤ 3 × rcvmem
then
    new receive buffer size = min(current receive buffer size × α +
    current receive buffer size, total remaining receive buffer size);
endif
```

(알고리즘 1) TBT-PLR

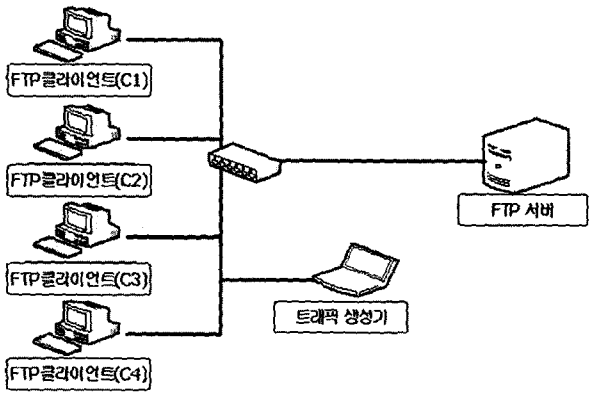
위의 (알고리즘 1)에서 보듯이, 초기에 연결이 성립하면 TCP 수신측 모든 연결마다 3×rcvmem 크기의 버퍼를 할당한다. 리눅스 커널에서 rcvmem은 하나의 MSS(maximum segment size)를 나타낸다. TCP 혼잡 제어 알고리즘에 따라 TCP 송신측에서는 초기에 하나의 세그멘트가 수신측으로 전달되며 이에 대한 ACK가 TCP 송신측에 도착하면 이후로는 이전의 두 배의 세그멘트가 전달된다. 따라서, 버퍼의 부족으로 인한 장애가 없이 원활하게 데이터가 전송되도록 초기에 3×rcvmem 크기의 버퍼를 할당한다. 데이터를 전송하는 동안에, TCP 수신측 버퍼 공간은 실제 네트워크 상황에 따라 지속적으로 변화한다. 만약 패킷 손실로 인한 타임아웃(time-out)이 발생한다면 CWND가 하나의 세그멘트 값으로 변경된다. 따라서 각 연결들은 최소 버퍼 공간인 3×rcvmem 보다는 큰 버퍼 공간을 할당받는다. 또한 만약 수신버퍼의 잔여공간이 3×rcvmem 보다 작다면 수신 버퍼는 현재 수신 버퍼×α+현재 수신 버퍼와 전체 TCP 수신측 버퍼의 잔여 공간의 최소값으로 변경한다. 더욱이, 제안한 TBT-PLR 알고리즘은 할당되지 않은 잔여 버퍼공간은 다음 연결이 발생했을 때 할당하게 되므로 고정된 수신 버퍼공간을 가진 경우보다 가용한 연결의 수가 증가한다.

4. 실 험

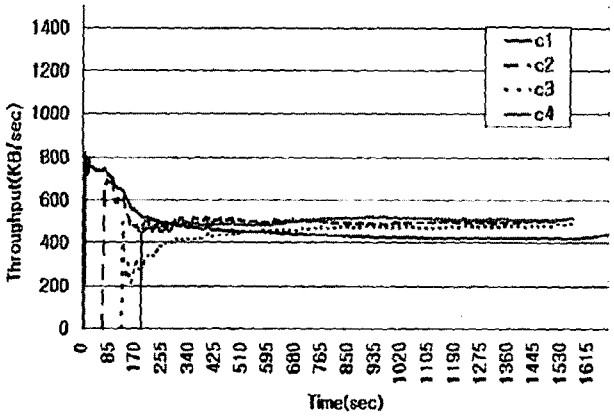
4.1 실험 환경

리눅스 커널 2.4.18을 수정하여 TBT-PLR 방법을 적용하였다. 제안한 TBT-PLR 방법의 성능을 실험하기 위해 리눅스 커널 2.4.18에서 고정된 TCP 수신측 버퍼의 크기가 큰 값으로 정의한 경우와 작은 값으로 정의한 두 가지 경우를 비교하였다.

실험 환경은 (그림 6)과 같이 이더넷 링크로 연결된 4개의 FTP 클라이언트(C1~C4)와 TBT-PLR 방법이 구현된 FTP 서버, 그리고 하나의 트래픽 생성기로 구성하였다. 단순한 실험에서 많은 TCP 연결들을 발생하기 어려우므로 전체 가용한 TCP 수신측 버퍼 크기를 제한하였고 네트워크 트래픽을 발생하기 위한 트래픽 생성기를 추가하였다. FTP 클라이언트 C1부터 C4까지 1분 간격으로 FTP 서버와 연결을 시작하여 각 연결마다 700메가바이트의 데이터를 전송하였다. 이 실험을 통해 측정된 전송률과 전송시간은 연결의 공평성(fairness)을 통해 향상된 성능을 보이는데 이용하였다.



(그림 6) 실험 환경

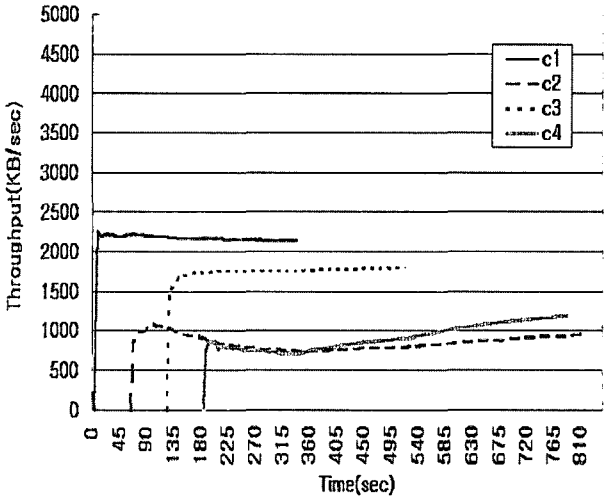


(그림 7) 디폴트 버퍼 크기 : 작은 버퍼

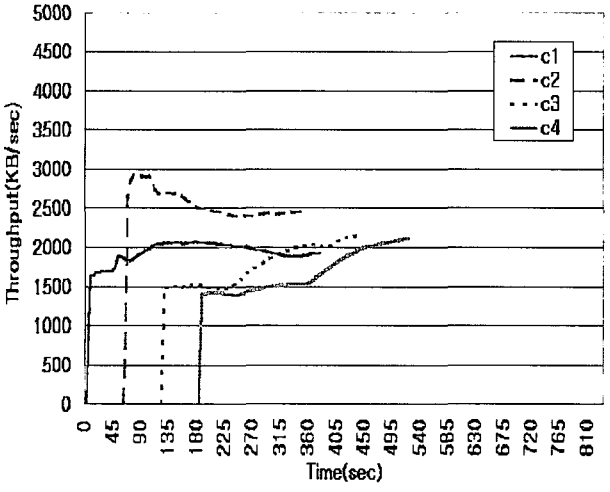
(그림 7)은 각 TCP 연결마다 고정된 버퍼 크기인 128킬로바이트의 작은 버퍼공간을 정의한 경우이다. 버퍼 공간이 매우 작기 때문에 각 연결은 가용한 대역폭을 충분히 사용할 수 없다. 첫 연결이 성립하고 초기 5초의 시간동안 상대적으로 높은 대략 초당 800킬로바이트의 전송률을 보였지만 급속하게 처리율이 감소하여 대략 초당 450킬로바이트를 유지하면서 지속적으로 작은 변동을 보이고 있다. 매 1분마다 연결이 하나씩 증가하면서 초기 처리도 점차 낮아지고 모든 연결은 초당 500킬로바이트의 전송률로 근접하는 경향을 보이고 있다.

(그림 8)의 실험은 서버에서 하나의 연결마다 할당하는 TCP 수신 버퍼 크기를 1028바이트로 설정한 경우이다. 이전 (그림 7)의 작은 버퍼를 사용한 경우와는 달리 각 연결은 가용한 대역폭을 충분히 사용할 수 있다. C1은 처음 5초 때 2260KB/sec를 나타내는데 (그림 7)의

작은 버퍼를 할당 받은 경우의 823KB/sec에 비교하면 버퍼가 큰 경우에 초기 전송률도 크다는 사실을 알 수 있다. C1과 C3는 초기 전송률을 전송이 끝날 때까지 유지하고 C2와 C4는 초기 전송률이 감소하는 경향을 보이다가 C1의 전송이 마치는 시간대에서 증가하는 경향을 보인다. 초기에 큰 버퍼를 할당하므로 초기에 나타나는 전송률이 마지막까지 거의 유지되는 경향을 보이고 있다.



(그림 8) 디폴트 버퍼 크기 : 큰 버퍼



(그림 9) 동적인 버퍼 크기 : TBT-PLR

(그림 9)는 제안한 방법 TBT-PLR에 의해 패킷 손실에 따라 동적으로 수신측 TCP 버퍼 크기를 변화시킨다. C2를 제외하고는 초기 전송률이 갈수록 증가하는 경향을 보이고 있다. C2의 경우는 초기에 높은 전송률을 보이다가 떨어지는 경향을 보이지만 C1, C3, C4는 (그림 8)에 비해 전반적으로 높은 처리율을 나타내고 시간이 경과함에 따라 전송률이 높아지는 경향을 나타낸다. 연결이 성립하고 버퍼크기가 증가하거나 유지하는 경향을 보이므로 초기에 큰 버퍼를 할당한 경우에 비해 증가하는 경향을 보이고 있다. (그림 8)의 고정된 큰 버퍼를 할당받은 경우와 비교했을 때 C4가 전송을 마치는 시간은 13분 30초인데 비해 제안한 방법에 의한 경우의 C4는 8분 40초에 전송을 마쳐 나은 성능을 보이고 있다. 그리고 각 연결의 전송이 마치는 시점을 통해 이전의 경우들보다 전체적으로 나은 연결의 공평성(fairness)을 확인할 수 있다.

5. 결 론

실험을 통해 TBT-PLR 방법은 패킷 손실률을 이용해 예측한 네트워크 상황에 따라 가용한 TCP 수신측 버퍼 크기를 변화시킴으로써 모든 TCP 연결에 있어 공평성과 효율적인 메모리 사용을 보여주었다. 본 논문에서 제안한 패킷 손실률에 기반한 효율적인 TCP Buffer Tuning 알고리즘인 TBT-PLR은 BDP 계산이나 추가적인 동작에 대한 오버헤드를 줄이면서 버퍼사용의 공평성과 전송률을 높이고자 하였다. TCP 송신측 버퍼 크기를 2배에서 4배 크기로 확보해 두는 ATBT를 송신측에 적용하였고 수신측에서는 TCP 송신측의 전체 가용한 버퍼를 패킷 손실에 따라 비례적으로 할당하였다. TBP-PLR 알고리즘은 망이 고속화 되어 높은 BDP가 요구되고 연결의 수가 증가하는 그리드(Grid) 컴퓨팅 환경이나 슈퍼컴퓨팅 환경에서 동작하는 응용프로그램에 적용할 수 있다.

참 고 문 헌

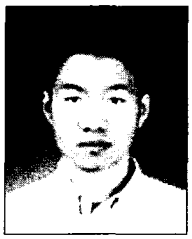
[1] J. Semke, J. Mahdavi and M. Mathis, "Automatic TCP Buffer Tuning," ACM SIGCOMM 1998, Vol.28, No.4, 1998.

[2] Eric Weigle and Wu-chun Feng, "Dynamic Right-Sizing:A Simulation Study," IEEE ICCCN, 2001.

[3] M. Fisk and W. Feng. "Dynamic Right Sizing in TCP," In Proceeding of the Los Alamos Computer Science Institute Symposium, LAUR 01-5460, Oct 2001.

[4] Takahiro Matsuo, Go Hasegawa and Masayuki Murata, "Scalable Automatic Buffer Tuning to Provide High Performance and Fair Service for TCP Connection," In Proceedings of IEEE INET 2000, July, 2000.

[5] Eric Weigle and W. Feng, "A Comparison of TCP Automatic Tuning Techniques for Distributed Computing" 11th IEEE International Symposium on High Performance Distributed Computing HPDC11 2002(HPDC'02), July, 2002.



류 기 철

e-mail : gcyoo@lge.com
2003년 영남대학교 컴퓨터공학과(학사)
2005년 경북대학교 컴퓨터공학과 (공학석사)
2005년~현재 LG전자 Digital Media 연구소 연구원
관심분야 : 초고속 인터넷, Mobile Ad Hoc Network, 센서 네트워크 등

김 동 균

e-mail : dongkyun@knu.ac.kr
1994년 경북대학교 컴퓨터공학과(학사)
1996년 서울대학교 컴퓨터공학과(공학석사)
2001년 서울대학교 전기·컴퓨터공학부 (공학박사)
1999년 미국 Georgia Institute of Technology, 방문 연구원
2002년 미국 University of California at Santa Cruz, Post-Doc. 연구원
2003년~현재 경북대학교 컴퓨터공학과 전임강사
관심분야 : 이동인터넷, 초고속 인터넷, Mobile Ad Hoc Network, 무선 LAN 등