논문 2006-43TC-3-16

대역폭과 지연의 곱이 큰 네트워크를 위한 개선된 TCP 혼잡제어 메카니즘

(Enhanced TCP Congestion Control Mechanism for Networks with Large Bandwidth Delay Product)

박 태 준*, 이 재 용**, 김 병 철**

(Taejoon Park, Jaeyong Lee, and Byungchul Kim)

요 약

현재 인터넷에서 널리 사용되고 있는 TCP는 대역폭과 지연의 곱이 큰 네트워크에서 특히 초기 시동단계를 포함하여 전반적으로 효율이 낮은 문제가 있다. 본 논문은 이 문제를 해결하기 위해 지연기반 혼잡제어(DCC:Delay-based Congestion Control) 방법을 제안한다. DCC는 선형과 지수 증가구간으로 나누어진다. 선형증가 구간은 기존의 TCP 혼잡회피 기법과 유사하며, 지수증가 구간은 혼잡에 의한 지연이 없는 경우 신속한 대역 확보를 위해 사용된다. 일반 TCP에서는 slow-start와 같은 지수증가 구간에서 대역과 지연의 곱으로 결정되는 크기의 버퍼가 제공되지 않는 경우 대역이 충분함에도 불구하고 손실이 발생하여 성능을 제한할 수 있다. 따라서 DCC에서는 RTT(Round Trip Time) 상태와 예측된 버퍼크기를 이용하여 지수증가 구간의 공급초과로 인한 손실을 방지하는 메카니즘을 제안한다. 시뮬레이션 결과를 통하여 대역과 지연의 곱이 큰 네트워크에서 DCC가 TCP에서 초기 시동시간과 throughput성능을 향상시킴을 보였다.

Abstract

Traditional TCP implementations have the under-utilization problem in large bandwidth delay product networks especially during the startup phase. In this paper, we propose a delay-based congestion control(DCC) mechanism to solve the problem. DCC is subdivided into linear and exponential growth phases. When there is no queueing delay, the congestion window grows exponentially during the congestion avoidance period. Otherwise, it maintains linear increase of congestion window similar to the legacy TCP congestion avoidance algorithm. The exponential increase phase such as the slow-start period in the legacy TCP can cause serious performance degradation by packet losses in case the buffer size is insufficient for the bandwidth-delay product, even though there is sufficient bandwidth. Thus, the DCC uses the RTT(Round Trip Time) status and the estimated queue size to prevent packet losses due to excessive transmission during the exponential growth phase. The simulation results show that the DCC algorithm significantly improves the TCP startup time and the throughput performance of TCP in large bandwidth delay product networks.

Keywords : Congestion control, Highspeed TCP, large bandwidth-delay product networks

I. 서 론

인터넷에서 신뢰성 있는 전달 프로토콜로서 전통적으로 TCP가 가장 널리 사용되고 있다. 최근 다양한 형태의 대규모 프로젝트를 위한 그리드 네트워크 등의 등장에 따라 대역과 지연의 곱이 큰(HBDP:High Bandwidth-delay product) 네트워크의 수요가 증가하고 있다[1]. 그러나 전통적으로 널리 사용되는 TCP를 HBDP 네트워크에서 대량의 데이터를 전달하기 위해 사용하면 제공된 대역을 모두 사용하기 위해 과도하게 긴 시간이 필요하여 혼잡윈도우 증가속도로 인한 대역 효율 문제가 발생하며, 또한 현재의 통신장비 기술로서 만족할 수 없을 정도로 낮은 데이터 손실률을 요구하게

* 정회원, 한국전자통신연구원 (ETRI)
** 정회원, 충남대학교 전기정보통신공학부 (Department of Information and Communication Engineering, Chungnam National University)
접수일자: 2005년12월27일, 수정완료일: 2006년3월14일

된다.

본 논문에서는 HBDP 네트워크에서 신속한 대역확보가 필요한 경우를 위해, TCP의 slow start(SS)가 발생하는 제한적인 경우에만 사용하던 지수증가 구간을 혼잡회피 구간에 적용하는 수정된 혼잡제어 방법을 제안한다. 혼잡회피 구간에서 지수증가 구간을 사용하기 위해서는, 선형증가 구간의 손실에 비해 파급효과가 큰 지수증가 구간의 패킷 손실을 방지하는 것이 중요하다. 혼잡에 의한 손실은 버퍼 저장에 의한 지연이 먼저 발생한다. 따라서 현재의 RTT와 최소 RTT간의 관계로부터 손실을 유발할 수 있는 지연이 발생하는지 여부를 판단할 수 있다. 본 논문에서는 이와 같이 지연을 이용한 손실 예측으로 혼잡회피구간에서 선형증가 또는 지수증가를 선택하는 메카니즘을 제안한다. 기존 네트워크에서 사용되는 경우와 호환성을 보장하기 위해 혼잡윈도우의 크기가 기존 네트워크에서 가능한 최대크기 이상에서만 이 알고리즘이 동작하도록 한다. HBDP 네트워크에서 제안된 방법이 TCP-Reno에 비해 초기 시동 성능과 오류 발생 후 가용대역 회복 특성이 개선됨을 시뮬레이션 결과로 보였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 본 논문과 관련된 기존 연구에 대해 설명하고, Ⅲ장에서 제안된 알고리즘을 설명한다. Ⅳ장에서 성능분석, Ⅴ장에서 시뮬레이션 결과를 보이며, Ⅵ장에서 본 논문의 결론을 맺는다.

Ⅱ. 관련연구

RFC 793[2]및 RFC 2581[3]과 호환성을 유지하는 전통적인 TCP는 처음 연결이 설정되면 패킷손실을 검출할 때까지 ACK 없이 송신할 수 있는 패킷의 수를 나타내는 혼잡윈도우를 지수적으로 증가시키는 slow start구간이 시작된다. slow start 구간에서 패킷 손실이 발생하면 최대가용대역의 초과로 혼잡이 발생했다고 판단하여 혼잡윈도우 크기를 반으로 축소하고 혼잡회피 구간을 시작한다. 혼잡회피 구간은 혼잡윈도우 만큼의 ACK를 수신하면, 혼잡윈도우가 1 증가하고, 손실이 발생하면 반으로 감소하는 즉, 선형적으로 증가하고, 배수적으로 감소하는 AIMD(Additive Increase Multiplicative Decreas)특성을 나타낸다. 그러나 AIMD 특성의 혼잡제어 알고리즘으로 HBDP 네트워크의 전송효율을 높이기 위해 전송로를 패킷으로 가득 채우기 위해서는 지나치게 긴 시간과 낮은 손실률이 필요하다. 일반적으로 알려진 예로, 패킷길이가 1500바이트, RTT가 100ms인 TCP 연결이 정상상태에서 10 Gbps 성능에 도달하기 위해서는 평균 혼잡윈도우 크기가 83,333 세그먼트가 되어야 하며, 패킷 손실은 매 5,000,000,000 패킷마다 1번 이하의 손실이 발생해야 한다. 즉, 가용대역을 모두 사용하기 위해서는 평균 비트오류율이 ($2x10^{-14}$)이하를 만족해야 하는 것으로 현재의 네트워크 기술로는 수용할 수 없는 조건이다. 이 문제를 해결하기 위해 HighSpeed TCP 등 다양한 연구가 수행되고 있다[4][5].

Sally Floyd가 제안하였던 HighSpeed TCP[4]는 전통적인 TCP의 AIMD 알고리즘에서 혼잡윈도우의 증가인자의 범위를 1(전통적인 TCP)에서 73패킷까지, 감소 인자의 범위는 0.5(전통적인 TCP)에서 0.09까지로 수정하였다. 이 효과로 현재의 혼잡윈도우가 클수록 증가하는 크기가 커지고, 손실에 의한 혼잡윈도우 감소량이 작아지므로, 가용대역확보에 필요한 시간이 단축되고, 손실에 의한 회복 속도가 빨라진다. Tom Kelly가 제안한 Scalable TCP[5]는 손실이 발생하면 감소인자의 크기를 0.125로 한다. 그러나 증가 방법은 ACK수신마다 현재의 혼잡윈도우와 무관하게 0.01 증가한다. 따라서 혼잡윈도우가 클수록 손실복구가 신속이 이루어진다.

HighSpeed TCP와 Scalable TCP 모두 현재의 혼잡윈도우 크기가 클수록 증가속도가 더 빠르므로, 혼잡윈도우가 작은 연결에 대해 심각한 공평성 문제가 발생한다. 본 논문에서 제안한 알고리즘은 공평성에 유리한 선형증가 구간과 신속한 대역확보가 가능한 지수증가 구간을 적절히 지연특성에 따라 조합하여 사용하며, 지수증가 구간에서도 현재의 혼잡윈도우 크기와 무관하게 동일한 기간에 대해 동일한 크기가 증가하므로 공평성 확보에 유리하다.

Ⅲ. TCP 혼잡제어 개선 알고리즘 제안

전통적인 TCP의 CA구간은 혼잡윈도우가 선형적으로만 증가하며, Timeout과 같은 제한적인 경우에만 SS로의 전환이 가능하다. 이러한 전통적인 AIMD기반 혼잡제어방법이 HBDP 네트워크에 적용되는 경우, 대역확보가 적절히 이루어지지 않아 효율이 심각하게 낮은 문제가 있다.

본 논문은 신속한 가용대역 확보에 유리한 지수적인 혼잡윈도우 증가방법과 안정적인 대역활용을 위한 선형적인 증가방법을 적절히 조합하여 효과적인 대역활용이

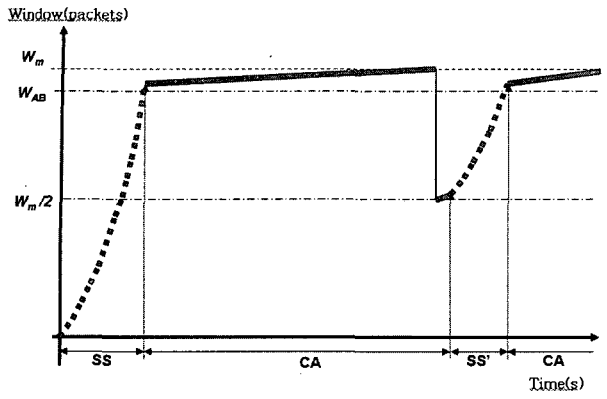

그림 1. 윈도우 크기 변화 예
Fig. 1. Example of window size evolution.

가능한 방법을 제안한다. 지수증가 방법을 사용하기 위해 종단간 지연시간을 이용한 지수증가 중단시점 결정방법, 버퍼크기 예측기법, 지수증가 시작점 결정방법 등을 제안한다.

제안한 TCP가 CA구간과 SS구간의 조합으로 구성된다는 점은 전통적인 방법과 동일하다. 그러나 제안된 방법은 CA구간에 SS'구간이 추가 되었다. SS'구간은 본 논문에서 제안한 방안으로 전통적인 TCP의 SS와 유사하나 CA구간에서 손실이 없이 신속한 대역확보가 필요한 경우에 사용한다. 이 구간은 손실이 발생하기 이전에 종료되며, CA구간의 선형증가 중에도 필요에 따라 지수증가 구간인 SS'구간이 시작될 수 있으며, 이때 SS'구간의 지수증가는 시작점을 기준으로 이루어진다. 그림 1은 본 논문에서 제안한 방법을 연결설정 후 최초 SS구간부터 하나의 혼잡손실주기(TDP:Triple Duplicate ACK Period)에 대해 CA구간, SS'구간의 조합까지를 표현한 그림이다. 혼잡윈도우 크기가 지수적으로 증가하는 구간에서의 손실은 성능에 치명적인 문제를 유발할 수 있으므로 지연 특성을 이용하여 손실이 발생하기 이전에 선형증가 구간으로 전환을 결정한다. SS'구간에서 CA구간으로 전환되면 가용한 크기의 대역을 충분히 확보한 상태이므로 혼잡윈도우를 급속히 증가시킬 필요는 없다. 혼잡윈도우가 선형적으로 증가하는 CA구간에서 지연특성과 현재의 혼잡윈도우 크기에 따라 HBDP네트워크에서 신속한 대역확보가 필요한 경우 SS'구간의 시작 여부를 결정한다. 이때 순간적으로 과도한 증가를 방지하기 위해 항상 지수증가를 일으키는 윈도우는 SS'구간이 시작된 혼잡윈도우를 1로 매핑하여 기준으로 삼는다.

제안된 방법은 HBDP 네트워크를 적용 대상으로 하며, HSTCP와 같이 일정기준 이하의 혼잡윈도우에서는 기존의 TCP 방법이 운용되어 기존 네트워크에 대한 호환성을 유지한다. 다음 절에서 구간별 동작에 대해 자세히 설명한다.

1. 버퍼크기 예측에 의한 SS 제어

최초 SS구간은 가용한 대역을 신속히 확보하기 위해 혼잡윈도우를 지수적으로 증가한다. SS구간에서는 ACK를 수신하면 2개의 패킷을 송신하게 되므로, cross traffic 등의 영향을 무시할 정도로 가용대역이 충분히 큰 경우, 네트워크 전달경로의 병목노드에서 패킷의 1/2은 TCP의 self-clocking 율에 따라 저장없이 전달되고, 1/2은 저장 후 전달된다. 이때 버퍼에 저장될 양이 버퍼를 초과하는 경우에 손실이 발생한다. 이러한 문제로 발생하는 초기시동 성능문제를 해결하려는 다양한 노력이 있었다[6].

본 논문은 SS 구간에서 지연증가 없는 손실이 발생하는 경우 버퍼부족에 의한 손실로 판단할 수 있으며, 이때의 버퍼크기는 손실이 발생한 혼잡윈도우의 1/2로 판단할 수 있는점을 이용한다. 예측한 버퍼크기가 대역과 지연의 곱 크기보다 작으며, 이후 각 라운드에 대해 연속적인 slow start 횟수를 버퍼크기로 제한하여, SS구간에서 버퍼 크기에 의한 패킷 손실을 방지할 수 있다. 본 논문에서 제안한 버퍼크기 예측에 의한 SS횟수 제한 방법은 기존의 네트워크에 비해 더 큰 버퍼가 필요한 HBDP 네트워크 등의 SS구간에서 우수한 특성을 나타낸다.

제안된 버퍼크기 예측방안을 적용하기 위해서는 SS중에 발생하는 손실의 원인을 명확하게 판단할 수 있어야 한다. 본 논문은 버퍼크기 부족에 의한 손실과 혼잡 또는 무작위(Random)에 의한 손실을 지연과 손실유형으로 구분한다. 혼잡에 의한 손실은 저장에 의한 지연으로 RTT가 증가하는 것을 확인할 수 있다. 지연이 없는 손실이 발생한 경우, 손실된 패킷 이후 동일 라운드의 나머지 패킷이 모두 손실되면 버퍼부족으로 판단할 수 있다. 라운드의 마지막 패킷은 다음 라운드에서 판단하도록 한다.

2. 혼잡회피구간(CA)

CA구간은 전통적인 TCP에서 SS로 증가하던 혼잡윈도우가 ssthresh에 도달하거나, 패킷 손실을 검출한 후 나타나는 선형증가 구간으로 연결시간의 대부분을 차지한다. 앞서 언급한 바와 같이 HBDP 네트워크에서 기존 방법의 선형증가로 가용한 대역을 확보하는 것은 적절

치 않다. 본 논문은 CA구간에서도 HBDP 네트워크에서 신속한 대역확보가 필요한 경우, 조건이 만족되면 SS′ 구간으로 변경을 결정한다.

TCP연결의 RTT 증가는 버퍼저장에 의한 지연의 증가를 의미하며, 버퍼저장 지연의 증가는 가용대역에 비해 많은 데이터가 유입되고 있다는 의미이다. 그러나 반대로 최소의 RTT를 유지하는 경우, 현재 가용대역에 비해 적은 데이터가 유입되어 버퍼저장이 발생하지 않고 있다는 의미이다. 따라서 최소의 RTT가 유지되는 동안 혼잡윈도우의 증가를 지수적으로 수행하는 SS′구간으로 전환하게 되면 신속한 가용대역 확보를 할 수 있다. CA구간에서 SS′구간으로 전환을 위해서는 기본적으로 식(1)에서 *WaitTime*으로 나타내어지는 시간이 경과해야 한다.

$$WaitTime = max(SendTime, log_2(cwnd)) \quad (1)$$

식(1)의 *WaitTime*은 *SendTime*과 현재의 혼잡윈도우까지 지수적으로 증가하는데 걸리는 시간 중에서 더 큰 값으로 나타낸다. 여기서 *SendTime*은 CA구간에서 버퍼 넘침에 의해 손실이 발생하여 혼잡윈도우의 크기가 축소된 경우 버퍼에 쌓여있는 데이터를 모두 소모하는데 걸리는 시간으로 RTT가 최소 RTT까지 감소하기 위해 걸리는 시간이다.

그림 1의 W_{AB}점과 같이 RTT가 최소 RTT보다 큰 값을 가지게 되면, 유휴 대역을 모두 사용하여 병목대역에서 버퍼저장이 발생하는 상태이므로 패킷 손실이 발생할 때 까지 혼잡윈도우를 선형적으로 증가시킨다. 이때의 선형증가는 가용대역을 모두 사용하고 있는 상황이므로 대역확보의 목적은 없으며, 네트워크의 혼잡여부를 패킷 손실로 판단하는 기법의 장점을 유지하며 다른 연결과 공평성을 확보하기 위한 것이다. 이때의 선형증가율은 전통적인 방법에 비해 충분히 작아도 문제가 없으므로, 본 논문은 전통적인 방법의 라운드당 증가율인 1보다 작은 0.1을 사용한다.

3. 지수증가구간(SS, SS')

SS와 SS′구간은 가용한 대역을 신속히 확보하기 위해 혼잡윈도우를 지수적으로 증가하는 구간이다. 본 논문은 신속한 가용대역 활용을 위해 지수증가와 선형증가의 적절한 조합으로 동작한다. SS가 연결이 설정된 초기 또는 timeout으로 다시 시작하는 경우에 사용되는 것과 달리 SS′구간은 혼잡윈도우가 큰 값을 가지는 상태에서 선형증가하던 혼잡윈도우를 지수적으로 증가시키는 구간이다. 따라서 최초 연결이 시작된 경우에 사용하는 SS와 같이 수신하는 모든 ACK마다 2개의 패킷을 보내는 방식은 과도하게 패킷이 증가하여 패킷 손실을 유발한다.

본 논문에서는 SS′구간이 시작된 때의 혼잡윈도우를 cwnd_base라 할 때, 라운드별로 cwnd_base만큼 ACK를 수신한 이후에만 ACK수신마다 2개의 패킷을 보내는 방식의 지수증가를 시작한다. 최초구간이나 timeout 등의 경우와 같이 시작점 혼잡윈도우가 1인 경우는 기존과 동일하게 동작하며, HBDP네트워크에서 정해진 조건을 만족하여 CA구간에서 SS′구간으로 변경된 경우에는 그때의 혼잡윈도우가 cwnd_base가 되며, ACK 수신 횟수가 cwnd_base가 되면 지수증가가 시작된다. 예를 들면, cwnd_base=K인 경우, 첫 라운드의 혼잡윈도우는 cnwd_base+1, 둘째는 cnwd_base+2, 셋째는 cnwd_base+4, 넷째는 cnwd_base+8 등과 같다. 이런 증가 방법은 현재의 혼잡윈도우가 충분히 큰 경우에도, 초기의 과도한 증가를 방지할 수 있다. 또한 기존의 HighSpeed TCP와 Scalable TCP와 다르게 현재의 크기와 무관하게 동일한 기간에 대해 동일한 크기가 증가하므로 공평성 확보에 좋다. 식(2)는 제안된 방법으로 각 라운드에서 발생하는 혼잡윈도우 증가량을 나타낸 식이다.

앞서 살펴본 바와 같이 지수증가 구간의 패킷 손실은 성능 저하를 크게 가져올 수 있으므로 지연특성을 이용하여 지수증가 구간의 손실을 방지하기위해 RTT가 최소 RTT에 비해 미리 정의된 범위이상 증가하면 SS′에서 다시 선형증가의 CA구간으로 변경한다. 이를 위해 혼잡윈도우 증가는 TCP-Vegas[7]와 같이 한 라운드 증가 후, 다음의 한 라운드동안 RTT 변화를 확인한 후 지수증가를 계속 유지할 것인지, 선형증가로 전환할 것인지를 결정한다. RTT의 증가는 가용대역보다 트래픽이 더 많아 연결 경로에서 버퍼에 데이터가 축적되고 있다는 의미이다. 축적되는 데이터의 양이 흡수할 수 있는 버퍼크기를 초과하는 순간 손실이 발생한다. 이 상황이 혼잡윈도우가 지수적으로 증가하는 구간에서 발생하면 대량손실이 발생할 수도 있다는 점은 이미 잘 알려져 있다[6]. 따라서 손실이 발생하기 전에 즉, 버퍼에 데이터가 축적되기 시작하는 시점에 선형증가로 변경하여 이러한 문제를 최소화 한다.

$$Incr = \lfloor((max(cwnd(i) - cwndbase, 0) + 1)/\lfloor cwnd(i) \rfloor)\rfloor \quad (2)$$

```
When an ACK for a new packet arrives;
 If(cwnd > low_window){
    If (state == SS'){
      If (ACK_count≧cwnd_base)cwnd=cwnd +1;
      If ( RTT > (BaseRTT) ) {
        wait_time = log2(cwnd);
        state = CA;}
    }
    else If (state == CA){
      cwnd = cwnd +0.1/cwnd;
      wait_time --;
      If ( (RTT == BaseRTT)&(wait_time == 0))
        state = SS';
    }
 }
When triple-duplicate ACK arrive;
  cwnd = cwnd / 2;
  wait_time = log2(cwnd);
```

그림 2. 혼잡윈도우 갱신방법의 pseudo-code
Fig. 2. Pseudo-code for congestion window update mechanism.

그림 2는 제안된 방법의 의사프로그램이다. HS-TCP와 같이 혼잡윈도우 크기가 *low_window*이상인 경우에만 제안된 알고리즘이 동작하며, *low_window* 보다 작은 경우는 기존 TCP와 동일하게 동작하여 공평성을 유지한다.

Ⅳ. 성능 분석

정상상태(steady state) 성능 분석을 위해 CA와 SS 구간이 주기적으로 나타나는 주기적 모델에 대한 성능을 분석한다. 그림 3은 정상상태의 한 혼잡손실주기를 나타낸 그림으로서, SS'구간 시간 S와 CA구간 시간 C로 구성되며, 시간 C는 손실발생 이전의 CA구간인 C'와 손실 후의 CA구간인 C''의 합이다. C'구간에서 W_{AB}를 초과하는 패킷이 버퍼에 저장되며, C''구간에서는 $W_m/2$과 버퍼에 저장되었던 패킷이 전달되어 C구간 동안 W_{AB}의 전송률이 유지된다.

손실확률 p인 연결의 평균전송률 $X(p)$는 식(3)과 같이 평균전송시간 $E[X]$에 대한 평균전송량 $E[Y]$의 비로 표현할 수 있다.

$$X(p) = \frac{E[Y]}{E[X]} \tag{3}$$

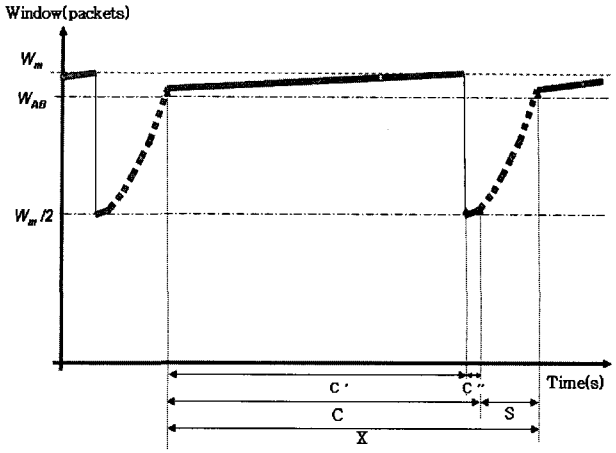

그림 3. 정상상태의 혼잡손실주기
Fig. 3. Congestion loss period in steady state.

이때 정상상태의 평균전송시간 $E[X]$는 CA구간, $E[C]$,와 SS'구간, $E[S]$,로 이루어진 정상상태 한주기로서 식 (4)로 나타낼 수 있다.

$$E[X] = RTT \cdot (E[C] + E[S]) \tag{4}$$

정상상태의 평균전송량 $E[Y]$는 CA구간 전송량인 $E[Y_C]$과 SS구간 전송량인 $E[Y_S]$의 합으로서 다음과 같이 표현할 수 있다.

$$\begin{aligned} E[Y] &= E[Y_C] + E[Y_S] \\ &= E[C]\,W_{AB} + \frac{W_m}{2}(k+1) + 2\sum_{i=0}^{k} 2^i \\ &= 1/p \end{aligned} \tag{5}$$

여기서 W_{AB}는 패킷수로 나타낸 가용대역의 크기를 나타내며, $W_{AB} = \frac{Available\,Bandwidth}{MSS} \times RTT$ 로 나타낼 수 있으며, k는 식 (6)을 만족하는 값으로서, $E[S]$는 앞 절의 SS구간 알고리즘에 의해 $(k+1)$ 라운드이다.

$$\frac{W_m}{2} + 2^{k-1} < W_{AB} \le \frac{W_m}{2} + 2^k \le W_m \tag{6}$$

정상상태의 평균 CA구간 크기인 $E(C)$를 구하기 위해 특성을 알고 있는 식 (7)의 $E(Y)$를 이용한다.

$$\begin{aligned} E[Y] &= E[C]\,W_{AB} + \frac{W_m}{2}(k+1) + (2^{k+1}-1) \\ &= E[C]\,W_{AB} + (W_{AB} - 2^k)(k+1) + (2^{k+1}-1) \\ &= E[C]\,W_{AB} + (k+1)\,W_{AB} - (k-1)2^k - 1 \end{aligned} \tag{7}$$

$E[C]$를 구하기 위해 식 (7)을 정리하면,

$$E[C] \approx \frac{1/p-(k+1)W_{AB}+(k-1)2^k+1}{W_{AB}} = \frac{1}{pW_{AB}}-(k+1)+\frac{(k-1)2^k+1}{W_{AB}} \quad (8)$$

와 같다.

성능을 표현하는 손실확률 p일때 연결의 평균전송률 $X(p)$는 식(3), (4), (5) 그리고 (8)에 의해 다음의 식 (9)와 같이 나타낼 수 있다.

$$X(p) \approx \frac{1/p}{RTT(E[C]+E[S])} = \frac{W_{AB}}{RTT\{1+p((k-1)2^k+1)\}} \quad (9)$$

식 (6)에서 $2^{k+1} \le W_m$이고 $k=\lceil log_2(W_m)-1 \rceil$ 이므로 식(9)는 식(10a)과 같이 나타낼 수 있다. 이때 p가 충분히 작으면 식(10a)는 식(10b)와 같이 표현할 수 있다.

$$X_{DCC}(p) \approx \frac{W_{AB}}{RTT(1+p((k-1)W_m/2+1))} \quad (10a)$$

$$\approx \frac{W_{AB}}{RTT} \qquad ,pW_m \ll 1 \quad (10b)$$

$$X_{Reno}(p) = \frac{1}{RTT}\sqrt{\frac{3}{2p}} \quad (11)$$

식(11)은 기존의 AIMD기반 TCP중 가장 대표적인 TCP-reno의 손실률 p에 대한 전송률 성능을 나타낸 식이다[8]. 이 식을 통해 기존의 TCP가 가용대역의 크기가 아닌 손실률로 평균전송률이 제한되어 대역 활용 효율에 문제가 있음을 알 수 있다. 제안한 혼잡제어 방법은 p와 W_m의 곱이 1보다 충분히 작으면 평균전송률 특성이 식(10b)과 같이 가용대역의 크기로 결정되어 우수한 대역 효율을 나타낼 수 있음을 알 수 있다. 식 (10)의 특성은 다음절의 시뮬레이션 결과로 확인할 수 있다.

V. 시뮬레이션 결과

본 논문에서 제안한 방법의 특성을 확인하기 위해 패킷 손실률에 따른 특성과 가용한 대역을 확보하기 위해 필요한 시간 등에 대하여 시뮬레이션을 하였다. 손실이 성능에 미치는 영향을 확인하기 위해 임의손실이 없거나 충분히 적은 네트워크와 임의손실이 큰 네트워크에 대해 시뮬레이션 결과를 비교하였다. 가용대역 확보특성은 시동구간의 혼잡윈도우 변화로 확인하였다.

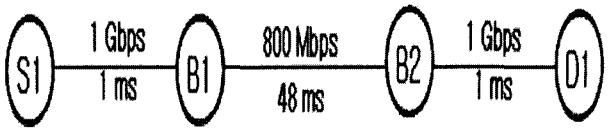

그림 4. 시뮬레이션 네트워크 구성도
Fig. 4. Simulation network topology.

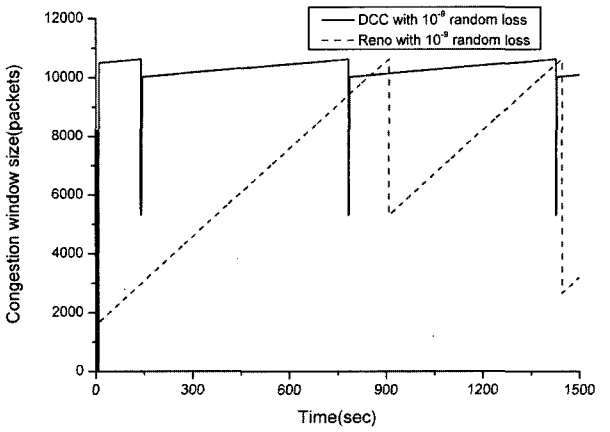

그림 5. 10^{-9} 의 임의손실 연결의 혼잡윈도우 변화
Fig. 5. Congestion window behavior of TCP-DCC and TCP-Reno mechanisms with random loss of 10^{-9}.

본 논문의 모든 시뮬레이션 결과는 ns-2[9]를 사용한 결과로서, 단일 병목 링크를 갖는 네트워크를 대상으로 하였다. 시뮬레이션 네트워크 구성도는 그림 4와 같다. 송수신 노드는 1Gbps로 접속되며, 병목 노드의 대역폭은 800Mbps, RTT는 100ms로 하였다.

그림 5는 시동단계와 정상상태의 혼잡윈도우 변화를 관찰할 수 있는 최초 1500초간의 혼잡윈도우를 나타낸다. 왕복지연 100ms, 병목노드의 버퍼크기와 대역이 1000패킷과 800Mbps, 그리고 단대단 경로의 임의 손실률이 10^{-9} 으로 $1/W_{AB}$ 에 비해 충분히 작은 네트워크를 대상으로 하였다. 제안된 방법은 10초 이내에 가용대역을 충분히 활용할 수 있으며, 정상상태에서 지속적으로 가용대역을 충분히 활용할 수 있다. 그러나 기존의 방법은 800여초가 경과된 후 가용대역을 활용할 수 있으며, 이후 가용한 대역을 모두 활용하기 위해서는 알려진 바와 같이 대역과 지연의 곱으로 결정되는 크기의 버퍼가 제공되어야 한다. 단대단 경로의 임의 손실률 p가 10^{-9} 으로 $1/W_{AB}$ 에 비해 충분히 작으면 식(10a)로 해석이 가능함을 그림 5를 통해 확인할 수 있다. 기존의 방법은 식(11)과 같이 손실률로 성능이 제한되므로,

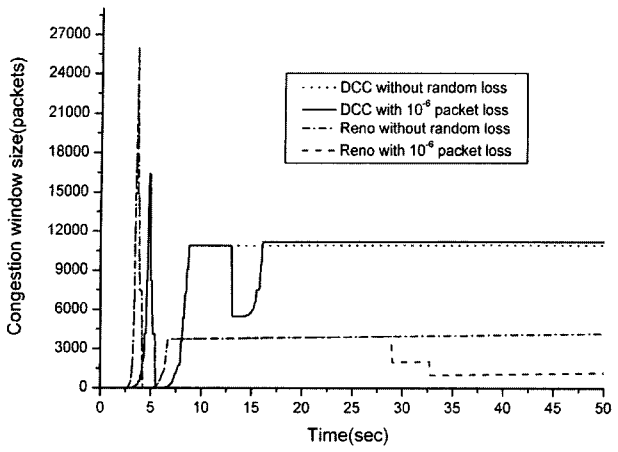

그림 6. 시동기간 혼잡윈도우 변화
Fig. 6. Congestion window behavior during startup phase.

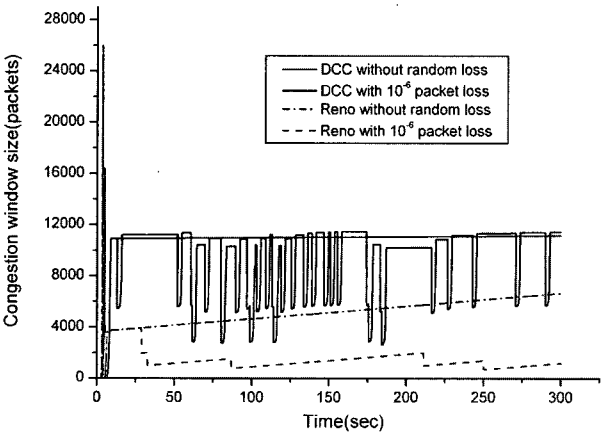

그림 7. 혼잡윈도우 변화
Fig. 7. Congestion window behavior

제공되는 대역의 크기와 무관하게 손실률에 의하여 최대성능의 한계가 결정된다. 그러나 제안된 방법은 식 (10a)와 같이 성능이 가용대역의 크기로 결정되므로 구조적으로 충분한 대역활용이 가능하다.

그림 6은 시동단계의 혼잡윈도우 변화를 관찰할 수 있는 최초 50초간의 혼잡윈도우를 나타낸다. 이 그림은 제안된 방법과 TCP-reno를 사용할 때 그리고 임의손실이 없는 이상적인 경우와 임의손실률이 10^{-6} 으로 비교적 큰 경우를 비교하였다. 단대단의 왕복지연이 100ms이고, 병목노드의 버퍼크기와 대역이 5000패킷과 800Mbps인 네트워크를 대상으로 하였다. DCC는 본 논문의 버퍼크기 예측 기법을 적용하여 혼잡윈도우가 10초 이전에 가용대역을 충분히 활용할 수 있는 크기로 증가한다. 임의 손실이 있는 연결의 경우 최초 13초경 임의손실이 발생하고, 이때 혼잡윈도우를 1/2로 축소하나 약 3초 이내에 회복된다. 반면 TCP-reno는 가용대역을 충분히 활용하는 구간이 관찰하는 최초 50초 이내

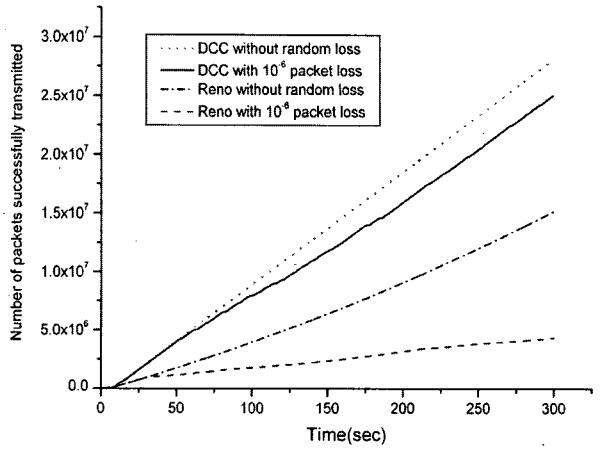

그림 8. 성공적으로 전달된 누적패킷 수
Fig. 8. Cumulative number of packets successfully transmitted.

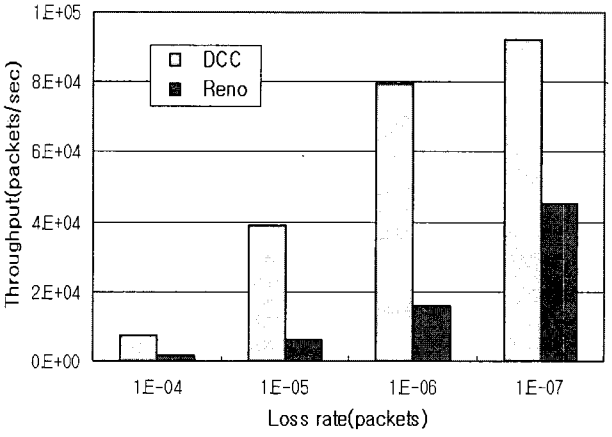

그림 9. 손실률에 따른 전송률
Fig. 9. Time averaged throughput versus loss rate.

에서는 나타나지 않으며, 약 29초에서 발생하는 임의손실로 가용대역의 크기와 무관하게 혼잡윈도우가 낮아져 대역 활용 효율이 급격히 저하되고, 가용대역 확보 또한 더 어렵게 된다.

그림 7은 그림 6의 시동단계 이후 안정상태의 혼잡윈도우 변화를 관찰하기 위한 최초 300초간의 혼잡윈도우를 나타낸다. 제안된 방법은 임의 손실이 발생하면 공평성을 위한 구간을 거쳐 가용한 대역을 확인하면 신속히 손실이 없는 경우에 따르는 특성을 확인할 수 있다. 그러나 TCP-reno는 임의손실이 없는 경우에도 300초까지 가용대역을 모두 확보하지 못한 상황이며, 임의손실이 발생하면 송신률이 더 축소되어 성능이 더 낮아지게 된다.

그림 8은 그림 7의 혼잡윈도우 변화에 대해 성공적으로 전달된 누적패킷수를 나타낸 그림이다. 임의 손실이 없는 경우, 제안된 방법이 기존 방법에 비해 성공적으

로 전달된 패킷수가 많다. 임의 손실률이 10^{-6}인 경우, 제안된 방법에 비해 기존 방법에서 성공적으로 전달되는 패킷량이 더 크게 감소되어 모든 경우에 대해 제안된 TCP-DCC가 TCP-reno에 비해 우수함을 확인할 수 있다.

그림 5는 손실확률 p가 1/WAB 에 비해 충분히 작아 식(10b)로 해석이 가능한 경우였다. 그러나 손실확률 p가 1/WAB 에 비해 충분히 작지 않으면, 손실에 따라 SS 구간이 발생하는 빈도가 증가하여 전체적인 성능에 미치는 영향도 증가한다. 그림 9는 손실확률이 1/WAB 에 비해 충분히 작지 않아 식(10a)에 의한 해석이 필요한 경우로서, 10-4에서 10-7 까지 손실률이 변화할 때 DCC와 기존방법의 성능을 비교한다. 그림에서 손실률이 증가함에 따라 DCC에 비해 Reno의 성능이 더 큰 폭으로 저하함을 알 수 있다. 손실확률과 더불어 가용대역의 크기가 성능에 반영되는 식(10a)와 반영되지 않는 식(11)의 차이와 같이, 손실률 증가에 따른 성능저하 정도가 DCC에 비해 TCP-reno가 더 크다는 것을 알 수 있다.

VI. 결론 및 향후 연구

본 논문에서는 기존 네트워크에서 사용되는 TCP 혼잡제어 방법이 대역과 지연의 곱이 큰 네트워크에서 사용할 때 발생하는 문제를 해결하는 방법을 제안하였다. 제안된 방법은 기존에 사용하던 Slow start의 지수적인 혼잡윈도우 증가방법과 혼잡회피의 선형적인 혼잡윈도우 증가방법을 새로운 방법으로 조합하여 신속한 대역확보에 의한 대역효율 문제를 개선하였다. 또한 버퍼크기 예측으로 초기시동 성능문제를 해결하였으며, 네트워크 경로의 상태를 반영하는 지연정보를 이용한 두 방법 간 전환으로 네트워크에 심각한 문제가 될 수 있는 지수 증가구간의 손실을 방지하여 가용 대역폭으로 짧은 시간 내에 수렴하여 사용할 수 있음을 보였다.

참 고 문 헌

[1] W.Allcock, et al, "GridFTP: Protocol Extensions to FTP for the Grid," GFD-R.020, 2004.
[2] J. Postel, "RFC 793: Transmission control protocol," Sep. 1981.
[3] M. Allman, V. Paxson, and W. Stevens, "RFC 2581: TCP Congestion Control," Apr. 1999.
[4] S. Floyd, "RFC 3649: HighSpeed TCP for large congestion windows," Dec. 2003.
[5] T. Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," ACM SIGCOMM, vol. 33, issue 2, pp. 83-91, Apr. 2003.
[6] S. Floyd, "RFC 3742: Limited Slow-Start for TCP with Large Congestion Windows," Mar. 2004.
[7] L. Brakmo and L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," IEEE JSAC, vol.13, pp.1465-1480, 1995.
[8] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," ACM CCR, vol. 27, no. 3, pp. 67-82, Jul. 1997.
[9] The network simulator ns-2. http://www.isi.edu/nsnam/ns/

저 자 소 개

박 태 준(정회원)
1992년 경북대학교 전자공학과 학사
1994년 경북대학교 전자공학과 석사
2001년~현재 충남대학교 정보통신공학과 박사수료
1994년~현재 한국전자통신연구원 선임연구원
<주관심분야: 인터넷, 데이터 통신, 초고속 통신>

김 병 철(종신회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1996년 한국과학기술원 전기 및 전자공학과 박사
1993년~1999년 삼성전자 CDMA 개발팀
1999년~현재 충남대학교 정보통신공학부 부교수
<주관심분야: 이동인터넷, 이동통신 네트워크, 데이터통신>

이 재 용(종신회원)-교신저자
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1995년 한국과학기술원 전기 및 전자공학과 박사
1990년~1995년 디지콤 정보통신연구소 선임연구원
1995년~현재 충남대학교 정보통신공학부 부교수
<주관심분야 : 초고속통신, 인터넷, 네트워크 성능분석>