논문 2005-42TC-12-20

고정대역 네트워크에서 혼잡윈도우 제한에 의한 TCP 성능개선

(Improving TCP Performance by Limiting Congestion Window in Fixed Bandwidth Networks)

박 태 준*, 이 재 용**, 김 병 철**

(Taejoon Park, Jaeyong Lee, and Byungchul Kim)

요 약

본 논문은 고정대역 네트워크에서 최대 TCP 혼잡윈도우를 제한하여, 버퍼 크기와 무관하게 안정적인 성능과 전송률을 제공할 수 있는 혼잡회피 알고리즘을 제안한다. 현재는 AIMD(Additive Increase, Multiplicative Decrease) 기반의 혼잡제어 방법이 가장 널리 사용되고 있다. 그러나 AIMD 기반의 TCP 혼잡제어 방법은 고정대역 네트워크에서 불필요하게 성능을 저하시키는 문제가 있다는 것이 여러 연구결과를 통해 알려져 있다. 또한 TCP의 톱니파형 데이터율로 안정적인 데이터 전송률이 필요한 응용에 적용하기 어렵다. 제안된 알고리즘은 필요에 따라 공평성을 유지하며 혼잡에 의한 손실을 방지하기 위해 혼잡윈도우의 크기를 제한한다. 최대 혼잡윈도우의 크기를 병목노드에서 데이터 축적을 방지하는 지연 정보로 결정하여, 별도의 버퍼와 윈도우 제어절차 없이 연결의 성능과 전송율이 안정적이도록 한다. 다양한 경우에 대한 시뮬레이션을 통해 호환성, 정상상태의 성능, 정상상태 손실 패킷 수, 그리고 혼잡윈도우의 분산 등으로 특성을 검증하였다. 제안된 방법은 송신단의 간단한 수정으로 적용이 가능하며, 네트워크 라우터와 사용자 프로그램의 수정이 불필요하여 확산이 용이한 장점을 가지며, 고정대역 네트워크로 볼 수 있는 국내 초고속인터넷 접속망에 적용하면 성능개선을 얻을 수 있다.

Abstract

This paper proposes a congestion avoidance algorithm which provides stable throughput and transmission rate regardless of buffer size by limiting the TCP congestion window in fixed bandwidth networks. Additive Increase, Multiplicative Decrease (AIMD) is the most commonly used congestion control algorithm. But, the AIMD-based TCP congestion control method causes unnecessary packet losses and retransmissions from the congestion window increment for available bandwidth verification when used in fixed bandwidth networks. In addition, the saw tooth variation of TCP throughput is inappropriate to be adopted for the applications that require low bandwidth variation. We present an algorithm in which congestion window can be limited under appropriate circumstances to avoid congestion losses while still addressing fairness issues. The maximum congestion window is determined from delay information to avoid queueing at the bottleneck node, hence stabilizes the throughput and the transmission rate of the connection without buffer and window control process. Simulations have performed to verify compatibility, steady state throughput, steady state packet loss count, and the variance of congestion window. The proposed algorithm can be easily adopted to the sender and is easy to deploy avoiding changes in network routers and user programs. The proposed algorithm can be applied to enhance the performance of the high-speed access network which is one of the fixed bandwidth networks.

Keywords : TCP congestion control, congestion window limit

I. 서 론

네트워크 기술의 발달로 가입자에게 제공되는 네트워크 서비스가 급속히 고급화되고 있다. 과거에는 네트워크에 대한 대부분의 투자가 규모를 확장하기위한 것

* 정회원, 한국전자통신연구원 (ETRI)
** 정회원, 충남대학교 전기정보통신공학부 (Department of Information and Communication Engineering, Chungnam National University)
접수일자: 2005년8월19일, 수정완료일: 2005년12월6일

이었으나, 점차 부가가치를 높이기 위한 방향으로 변화하고 있다. 통신 사업자가 제공하는 네트워크 서비스의 부가가치를 높이기 위해 서비스 품질을 예측 가능한 수준으로 제공할 수 있는 기술에 대한 연구가 진행되고 있다. 주문형 대역서비스(Bandwidth on demand)를 위한 대역제한(Rate Limiting) 기능은 이미 신규 투입되는 네트워크 장비에 구현되어 있으며, 제한된 서비스 영역 내에서 연결별로 대역을 보장하는 기능이 곧 네트워크에 적용될 예정이다[1]. 홈네트워크 등 최근에 구성된 대부분의 가입자망은 점대점 통신이 가능한 이더넷 스위치로 구성된다. 이더넷 스위치를 사용하는 소규모 LAN은 점대점 연결에 대해 고정대역을 사용하는 효과가 있다. 이와 같은 네트워크 기술의 변화로 조만간 다양한 형태의 품질 보장형 네트워크 서비스가 일반화될 것으로 판단된다.

네트워크 환경의 변화에도 불구하고 네트워크 트래픽의 대부분을 전송하는 TCP는 여전히 공유 미디어 환경에 최적화된 AIMD기반의 혼잡제어 방법을 사용하고 있다. AIMD기반의 혼잡제어 방법으로 가용한 대역을 효과적으로 활용하기 위해서는 대역과 지연의 관계로 결정되는 적절한 크기의 버퍼 제어가 필요하다. 버퍼가 적정 크기보다 작으면 대역효율이 떨어지고, 크면 버퍼 지연의 증가로 혼잡제어가 적절히 이루어지지 못하여 집중적인 패킷 손실이 발생할 수 있다[2]. 이러한 문제를 개선하기위해 호스트의 버퍼를 제어하는 방법에 대한 연구가 활발히 진행되었다[3][4]. 그러나 네트워크 라우터 등 중간 노드의 버퍼를 제어하는 것은 어렵다. 이와 같은 버퍼제어의 어려움으로 오히려 버퍼를 없애는 것이 좋겠다는 의견도 있다[5].

또한 AIMD기반의 혼잡제어 방법은 가용 대역을 확인하기 위해 지속적으로 전송률을 증가시켜 혼잡에 의한 패킷 손실을 유도한다. 손실이 발생하면 가용대역을 초과한 패킷에 의해 네트워크에 혼잡이 발생했다고 판단하여 혼잡윈도우의 크기를 반으로 축소한다. 이러한 주기적인 패킷 손실과 재전송으로, 데이터 발생률이 톱니파형을 나타내어 안정적인 데이터 전송률이 필요한 응용에는 적용하기 어렵다. 이러한 특성이 고정대역을 보장해 준다고 하여도 변화하지는 않는다. 따라서 변화하는 네트워크의 특성을 효과적으로 활용하기 위해서는 TCP의 혼잡제어 방법도 적절하게 변화되어야 한다.

최근 연구에서 품질보장 서비스로 대역이 보장된 연결에서 혼잡윈도우의 크기를 제한하여 AIMD의 문제점을 해결하는 방안이 논의되었다[6][7]. 이 방법은 AIMD 특유의 주기적인 손실에 의한 톱니파형을 방지하여, 안정된 전송률을 유지할 수 있으며, 버퍼크기에 영향을 받지 않는 예측 가능한 성능을 얻을 수 있다. 그러나 기존 연구는 별도의 제어기능이 필요하여 사용이 제한적이며, 최대 혼잡윈도우 결정 방법이 네트워크 경로의 병목 대역이 아닌 호스트의 버퍼 크기로 결정되어 효율을 보장할 수 없다.

본 논문에서 제안하는 TCP-CWL(Congestion Window Limit)은 고정대역 네트워크에서 TCP의 데이터 발생률을 결정하는 혼잡윈도우 크기를 병목구간의 버퍼에 데이터 축적이 일어나지 않으면서 대역을 최대한 활용할 수 있는 크기로 제한하여 성능과 특성을 개선하는 방법이다. TCP-CWL은 가용한 최대대역을 알기 위한 별도의 제어기능이 필요치 않고, 송신단의 수정만으로 적용이 가능하다. 또한 기존의 네트워크에서도 안정된 고정대역을 사용하는 상황에서는 TCP-CWL이 동작하며, 대역을 공유하는 상황으로 변화하면 즉시 AIMD로 운용된다. 따라서 TCP-CWL은 고정대역 서비스의 사용여부와 무관하게 기존 네트워크에 즉시 적용이 가능하다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 혼잡윈도우 제한을 이용한 혼잡제어에 대한 관련연구와 적용분야에 대해서 설명하며, 제 Ⅲ장에서는 본 논문이 제안하는 TCP-CWL알고리즘에 대해 설명하였다. 제 Ⅳ장에서는 이를 시뮬레이션을 통해 기능동작을 확인하고, 기존의 혼잡제어 방법을 사용하는 경우와 성능을 비교 하였고 마지막으로 제 Ⅴ장에서 결론 및 향후 연구 계획을 제시한다.

Ⅱ. 관련연구 및 적용분야

1. 관련연구

가. TCP RUDE[6]

AIMD기반의 TCP 혼잡제어 방법을 그리드와 같이 제한된 용도에 사용하는 경우 네트워크 전달 성능이 불필요하게 손상될 수 있다. 최근 고성능 컴퓨팅과 공동작업을 위해 대용량 데이터를 공유하는 그리드가 늘고 있다. 이들은 대부분 TCP/IP를 사용한다. 이러한 네트워크는 대역을 공유하지 않으며 일정 수준의 품질을 보장받는 전용의 네트워크를 사용한다. 또한 장시간 연결을 유지하며, 대용량의 데이터 전달을 위해 사용한다. 이런 네트워크에서는 기존의 TCP혼잡제어 방법이 패킷손실 유발과 손실에 대한 과민반응으로 오히려 성능

을 저하시킬 수 있다. TCP RUDE는 이러한 문제점 때문에 고정대역이 보장된 네트워크에서는 TCP혼잡제어 기능을 정지하고, 사용자에게 미리 알려진 가용 대역만을 사용할 수 있도록 TCP수준에서 별도의 소켓을 제공한다. 이 방법은 연결 기반으로 혼잡제어기능을 활성 또는 비활성 할 수 있다. 그러나 이 방법은 별도의 소켓을 사용해야 하므로 기존 응용프로그램의 사용에 제한이 있고, 기존의 혼잡제어방법을 사용하는 트래픽과 네트워크를 공유할 수 없으며, 혼잡제어기능을 비활성화하는 경우 예상치 못한 네트워크 상황의 변화에 취약한 문제가 있다. 따라서 일반적인 용도로는 사용할 수 없고, 제한적인 용도로만 사용이 가능하다.

나. TCP Clamping[7]

본 논문에서 제안하는 바와 유사하게 혼잡윈도우의 크기를 손실이 발생하지 않는 크기로 제한하여 AIMD 특유의 문제를 피하는 방안을 제시한다. TCP Clamping의 최대 혼잡윈도우 크기는 호스트의 버퍼 메모리를 기준으로 결정하며, 별도의 절차로 최대 혼잡윈도우의 크기를 설정해야 한다. 그러나 최대 혼잡윈도우의 크기는 대역효율에 직접적인 영향을 주는 요소로서, 병목구간의 최대 대역을 기준으로 하지 않고 호스트의 버퍼 메모리를 기준으로 결정하는 경우 대역효율을 충분히 확보하지 못할 수 있다. 따라서 새로운 혼잡제어 방법은 효과적인 적용과 확산을 위해 별도의 제어절차 없이 연결에 대한 적용이 가능하고, 대역 효율성을 위하여 병목구간의 최대 가용대역에 의해 최대 혼잡윈도우의 크기가 결정되어야 한다.

2. 고정대역 네트워크에 대한 적용분야

품질보장 서비스의 가장 기본적인 형태인 고정대역 서비스는 새로운 장비로만 가능한 것은 아니며, 다음과 같은 기존의 네트워크 환경에서도 가능하다.

가. Home Network

댁내망 또는 소규모 사무실 등 스위치기반의 이더넷으로 구성된 소규모 네트워크는 대부분의 연결시간동안 점대점 연결에 대해 배타적으로 제공되는 고정대역을 사용하게 된다. 데이터 백업, 스트리밍 데이터 송수신 등 대부분의 장시간 유지되는 연결은 일대일 통신이 이루어지며 이때 대부분의 시간동안 고정대역 서비스의 특성에 의해 운용된다. 경우에 따라서는 단시간통신(short lived TCP)이 현재 연결된 포트 또는 대역을 공유할 수 있다. 따라서 이때는 두 연결이 기존의 방법과 같은 특성의 공평성을 유지해야 한다.

나. 인터넷 서비스

사업자의 마케팅 전략에 따라 다양한 형태의 고정대역 인터넷 서비스가 가능하다. 인터넷 전용선과 같이 계약에 의해 가입자에게 특정 점대점 연결에 대해 배타적인 대역이 제공되는 서비스와 인터넷 초고속가입자망 서비스처럼 계약에 의해 대역제한(Rate Limiting) 기능으로 제한된 접속 대역이 제공하는 경우 등이 있다. 또한 최근 BcN 기술에서 MPLS/RSVP 등을 이용한 단대단 품질보장 네트워크 서비스의 개발이 진행 중이다. 이러한 서비스는 특정 도메인 내에서는 고정대역이 제공되나, 서비스 도메인을 벗어나면 최선형 구간이 포함되므로, 최선형 구간에서는 공평성이 유지되어야 한다.

다. Streaming Video over TCP

일반적으로 멀티미디어 스트리밍 서비스를 위해 TCP를 이용하는 것은 부적합하다고 알려져 있다. 그 이유는 첫째, 패킷 재전송은 수용할 수 없는 크기의 단대단 지연을 유발할 수 있다는 점과, 둘째, TCP의 AIMD기반의 혼잡제어 알고리즘으로 혼잡윈도우의 크기가 톱니파 모양으로 나타나 안정적인 데이터율의 서비스에 부적합하다는 점이다[8][9].

첫째 문제점의 패킷손실에 의한 재전송으로 1 왕복시간(RTT:Round Trip Time)이 추가된 패킷의 지연은 원거리간 순수 대화형 응용과 같은 제한된 경우에만 문제가 될 수 있다[8]. 오히려 고품질 서비스를 위해 제한된 범위의 지연이 추가되더라도 손실에 의한 품질저하를 방지하기 위해 재전송이 필요할 수 있다. 따라서 둘째 문제인 안정된 전달특성이 제공되면 멀티미디어 서비스를 위해 TCP를 사용하는 것이 가능하다.

Ⅲ. 혼잡윈도우 제한 알고리즘

그림 1은 TCP-CWL의 혼잡윈도우를 결정하는 pseudo-code이다. 이 방법은 고정 대역 네트워크의 정상상태 TCP 연결에 대해 RTT정보를 이용하여 최대 혼잡윈도우의 크기를 병목구간의 대역을 기준으로 결정한다. 혼잡윈도우의 크기는 바이트 단위로 관리되나, 패킷은 MSS(Maximum Segment Size)단위로 결정되므로, 혼잡윈도우의 크기는 MSS단위의 세그먼트로 표현된 TCP_cwnd로 제한된다.

```
If (NormalRxAck == 1) {
  If ((CA==1) & (cwnd > 2 )) { // BaseRTT 결정과정
    If (RTT≤BaseRTT) & (BaseRTT > 0) {
      BaseRTT = RTT;
      BaseRTT_Time = now;}
    If ((now- BaseRTT_Time) > (BaseRTT*α) ) {
      BaseRTTavail = 1;
    } else {  BaseRTTavail = 0; }
    // cwnd_cwl 결정과정
    If((BaseRTTavail==1) {
      if (RTT≤BaseRTT*β)&(cwnd > cwnd_cwl)) {
        cwnd_cwl = cwnd;
        cwnd_cwl_Time = now;  }
      else { cwnd_cwl = 0; }
      If( (now-cwnd_cwl_Time) > (BaseRTT*δ)
                          & (cwnd_cwl > 0) {
        cwndAvail = 1;
        cwndAvail_Time = now;
      } else {cwndAvail = 0;}}
  } else {              //CWL 관련변수 초기화
    BaseRTTavail = 0;
    cwndAvail = 0;
    BaseRTT = 0;
    cwnd_cwl = 0; }
// 패킷 손실에 대한 cwnd_cwl 갱신여부 결정
else if ( PackDrop==1) & (DropOption == 1) {
  if ((now-cwndAvail_Time) > (RTT*λ) & (cwndAvail ==1)){
    // cwnd_cwl 갱신위한 CWL 관련변수 초기화
    BaseRTTavail = 0;
    cwndAvail = 0;
    BaseRTT = RTT*10;
    cwnd_cwl = 0;  }
}
TCP_cwnd = ⌊cwnd_cwl / MSS⌋;
```

그림 1. 제안된 방법의 pseudo-code
Fig. 1. pseudo-code of our mechanism.

1. BaseRTT결정

RTT는 TCP 연결의 지연특성을 나타내는 기본변수로 네트워크 경로의 데이터축적(Queueing) 여부를 판단하는 중요한 정보로 활용할 수 있다. 네트워크의 가용대역폭보다 작거나 같은 크기의 부하가 가해지면 전달경로에는 데이터 축적에 의한 지연이 없으므로 순수하게 전달에 필요한 지연만이 존재한다. 이때의 RTT를 데이터 축적이 없는 순수한 전달지연을 의미하는 BaseRTT로 한다. 가용 대역폭보다 큰 부하가 가해지면 전송하지 못하고 남은 트래픽은 버퍼에 축적되며, 축적된 트래픽의 전달에 필요한 시간만큼의 추가지연이 발생하게 된다. BaseRTT결정에 반영하는 RTT는 delayed ACK에 의한 영향을 고려하여 혼잡윈도우의 크기가 2 이상인 경우로 한다.

네트워크 상황으로 측정한 RTT를 BaseRTT로 결정하기 위해서는 신중한 판단이 필요하다. 변화중인 비정상상태의 BaseRTT를 네트워크 상황을 판단하는 근거로 사용하면 오류가 발생할 수 있다. 따라서 갱신된 BaseRTT를 BaseRTT검증시간(BaseRTT*α)이상 유지하며 정상상태의 값임을 검증한 후 네트워크 상황 판단에 이용한다. 여기서 (BaseRTT*α)로 표현되는 BaseRTT검증시간은 최소 지연을 의미하는 BaseRTT의 α배 시간으로, 가장 빠른 라운드 주기의 α배를 나타낸다. BaseRTT_Time은 BaseRTT가 마지막으로 갱신된 시간을 나타내며, BaseRTT검증시간의 경과를 확인하기 위해 사용하는 변수이다. BaseRTTavail 신호는 BaseRTT의 유효 여부는 나타내는 신호로서, 검증된 경우 '1'로 나타낸다. 여기서 α는 Triple Duplicate ACK 수신 주기에 따라 결정한다.

2. cwnd_cwl 결정

검증된 BaseRTT가 결정되면, 데이터가 버퍼에 축적되지 않는 최대 혼잡윈도우 크기인 cwnd_cwl을 결정하는 절차를 시작한다. 데이터 축적이 없는 최소의 RTT에서 지연이 증가하기 시작하는 시점이 데이터 축적이 시작되는 혼잡윈도우의 크기라고 판단할 수 있다. 정상적으로 수신되는 ACK마다 RTT를 BaseRTT와 비교한다. RTT와 BaseRTT의 차가 미리 정의된 범위 이내로 일치하고, 현재의 혼잡윈도우 크기가 cwnd_cwl보다 크면, cwnd_cwl을 현재의 혼잡윈도우 크기로 갱신한다. RTT가 BaseRTT와 보다 작으면 1절의 BaseRTT결정 과정을 다시 시작한다.

cwnd_cwl의 유효 여부를 판단하는 방법은 BaseRTT를 검증하는 과정과 유사하다. 최종 cwnd_cwl이 결정된 후 cwnd_cwl결정시간(BaseRTT*δ)이상 유지되면 유효하다고 판단한다. 여기서 (BaseRTT*δ)로 표현되는 결정시간은 최소 지연을 의미하는 BaseRTT의 δ배 시간을 의미한다. Cwnd_cwl_Time은 cwnd_cwl이 최종 갱신되었던 시간을 나타내고, cwndAvail은 cwnd_cwl 의 유효여부를 나타내는 신호로 사용한다.

BaseRTTavail과 cwndAvail이 모두 '1'이 되면 TCP 연결의 최대 cwnd는 cwnd_cwl을 세그멘트로 표현한 TCP_cwnd 로 제한된다. 여기서 δ는 Triple Duplicate ACK 수신에 의한 패킷 손실 확인에 걸리는 시간으로 결정한다.

```
If (NormalRxAck == 1) {
  If ((BaseRTTavail==1)&(CwndAvail==1)) {
    If (RTT≤BaseRTT*β) { // cwnd_cwl 증가
      If (cwlCnt < cwlCntMax) {
        cwlCnt= cwlCnt+1;
      } else {
        cwlCnt= 0;
        cwl_Incr +=MSS;   }
    } else {          // cwnd_cwl 증가 후 원위치
      If ((cwlCnt-1 == cwlCntMax) & (cwl_Incr > 0)) {
        cwl_Incr= cwl_Incr - 2*MSS;
        AddOneSig= 1;  }
      If ((cwlCnt ≥ cwlCntMax) & (AddOneSig=1)) {
        cwlCnt= 0;
        cwl_Incr +=MSS;     }
    }
  } else { // cwnd_cwl 증가관련 변수 초기화
        cwlCnt= 0;
        cwl_Incr= 0;   }
}
TCP_cwnd= ⌊(cwnd_cwl+cwl_Incr) / MSS⌋;
```

그림 2. cwnd_cwl 갱신방법의 pseudo-code
Fig. 2. pseudo-code of cwnd_cwl update mechanism.

3. cwnd_cwl 갱신

네트워크 경로를 공유하여 가용대역이 줄어들거나, 공유하던 연결이 해제되어 가용대역이 늘어나는 경우, 이러한 변화를 반영할 필요가 있을 때 cwnd_cwl 갱신 기능이 필요하다.

가용대역이 줄어드는 경우를 위한 cwnd_cwl 갱신은 관련 변수를 모두 초기화하고, 1절의 BaseRTT결정과정부터 시작한다. 대역 감소가 발생할 때마다 cwnd_cwl를 갱신하게 되면, 매번 cwnd_cwl결정과정을 거치게 되어 고유의 효율을 떨어뜨릴 수 있다. 가용대역이 축소된 경우 cwnd_cwl를 갱신하지 않으면, cwnd_cwl보다 작은 혼잡윈도우에서 손실이 발생하므로 기존의 AIMD로 동작한다. 따라서 심각한 문제는 발생하지는 않는다. 오히려 경우에 따라서는 가용대역 축소 상황에서는 cwnd_cwl를 갱신하지 않는 것이 효과적이다. 그러므로 cwnd_cwl를 갱신기능을 필요에 따라 선택할 수 있도록 그림 1과 같이 패킷 손실이 발생하고 DropOption 변수가 '1'인 경우에만 대역축소에 대한 cwnd_cwl 갱신기능이 동작토록 한다.

가용대역이 축소되는 경우와는 달리 확대되는 경우에 cwnd_cwl을 갱신하지 않으면 대역효율에 문제가 발생한다. 혼잡윈도우의 크기가 cwnd_cwl로 제한되므로 확대된 대역을 활용할 수 없기 때문이다. 따라서 가용대역의 증가 여부를 수시로 확인하고, 필요에 따라 갱신할 수 있어야 한다. 그림 2는 대역이 증가하는 상황에서 cwnd_cwl을 갱신하는 방법을 나타내는 pseudo-code이다. TCP_cwnd를 적용하여 cwlCntMax 라운드 시간이상동안 지연이 증가하지 않으면 가용대역의 확대여부를 검사하기위해, TCP_cwnd를 cwnd_cwl이 MSS만큼 증가된 크기에서 RTT를 확인한다. RTT변화가 정의된 영역 이내이면 cwnd_cwl을 갱신하여 추가대역을 확보하고, 정의된 크기 이상으로 증가한 경우에는 TCP_cwnd를 1라운드동안 최종 확인된 cwnd_cwl보다 작은 (cwnd_cwl-2*MSS)로 줄인 후 다시 cwnd_cwl로 돌아간다. 이 과정의 반복으로 대역이 증가하는 경우에도 적응이 가능하다.

가용대역이 줄어든 경우의 cwnd_cwl 갱신은 선택적으로 적용되는 방법으로서 가용대역 축소는 패킷 손실로 즉시 확인할 수 있다. 패킷 손실이 발생하면 호환성을 유지를 위해 기존의 AIMD 방법과 동일하게 현재의 혼잡윈도우 크기를 반으로 줄인다. DropOption이 '1'인 경우는 대역축소에 대한 갱신기능을 선택한 경우이므로 cwl 관련 모든 변수를 초기화하고 1절의 BaseRTT결정단계를 시작한다.

Ⅳ. 시뮬레이션 및 성능평가

본 논문에서 제안한 TCP-CWL은 고정대역 또는 대역제한 기능이 제공되는 네트워크 구간을 대상으로 한다. 모든 노드의 특성이 일시에 변화되기는 어려우므로 기존 TCP연결과 호환성을 유지하는 것은 여전히 중요하다. 본 절은 먼저 단대단 고정대역 네트워크에 대한 시뮬레이션으로 제안된 알고리즘의 특성을 확인한 후, 일반적인 네트워크 구성에 대한 다양한 시뮬레이션 시나리오에 의해 제안된 알고리즘의 동작과 TCP-Reno와 공평성을 시험하여 호환성을 유지함을 확인한다. 또한 시뮬레이션을 통해 본 논문이 제안한 TCP-CWL 트래픽과 대표적인 AIMD기반 혼잡제어 방법을 사용하는 TCP-Reno 트래픽의 특성을 비교한다. 본 논문의 모든 시뮬레이션은 ns-2를 사용하였다[10].

단대단 고정대역 네트워크에 대해 제안된 알고리즘을 검증하기 위한 시뮬레이션 네트워크 구성도는 그림

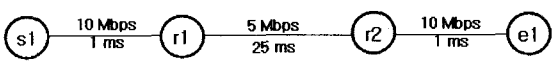


그림 3. 시뮬레이션 네트워크 구성도
Fig. 3. Simulation network topology.

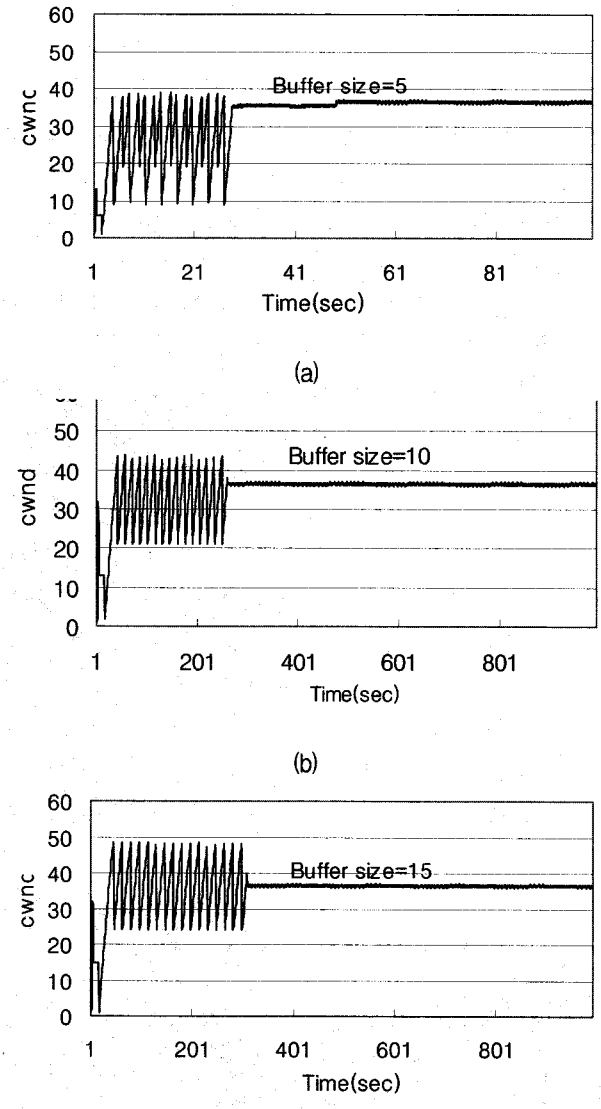


그림 4. 혼잡윈도우의 크기, (a) 버퍼크기=5 와 (b) 버퍼크기=10 과 (b) 버퍼크기=15

Fig. 4. Congestion window size, (a) buffer size=5, (b) buffer size=10 and (c) buffer size=15.

3과 같다.

그림 4의 (a), (b), (c)는 단대단 고정대역 네트워크에 대한 시뮬레이션 결과로 그림 3의 네트워크에 대해 병목노드의 버퍼 크기를 5, 10, 15로 변경하여 시뮬레이션을 하였다. 초기의 가용대역 확인을 위한 구간이 경과하면, 혼잡윈도우는 병목노드의 버퍼 크기와 무관하게 일정한 크기로 수렴한다. 따라서 병목노드의 버퍼 크기와 무관하게 손실이 발생하지 않는 가용대역을 확보함을 알 수 있다.

그림 5는 대역보장 네트워크와 최선형 네트워크가 섞여있는 경우에 기존 네트워크에 대한 호환성을 시험

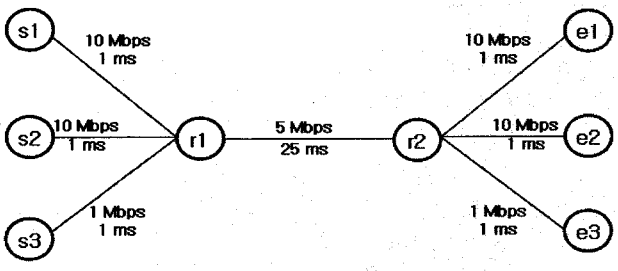


그림 5. 공평성 시험을 위한 시뮬레이션 네트워크 구성도

Fig. 5. Simulation network topology for fairness test.

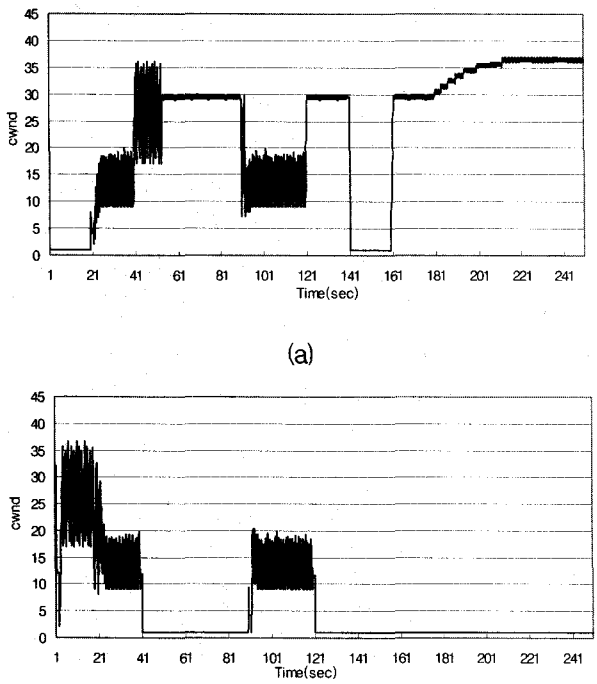
표 1. 시뮬레이션 시나리오

Table 1. Simulation scenario.

구간	시간(sec)	s1 → e1 TCP-CWL	s2 → e2 TCP-Reno	s3 → e3 UDP(1Mbps)
1	0 - 20	Off	On	On
2	20 - 40	On	On	On
3	40 - 90	On	Off	On
4	90 - 120	On	On	On
5	120 - 140	On	Off	On
6	140 - 160	Off	Off	On
7	160 - 180	On	Off	On
8	180 - 250	On	Off	Off

하기 위한 네트워크 구성도 이다. 이를 위해 다양한 시뮬레이션 시나리오에 의해 제안된 알고리즘의 동작과 TCP-Reno와 호환성을 유지함을 확인한다. 노드 s1에서 노드 e1으로는 본 논문에서 제안한 TCP-CWL을 사용하는 FTP 프레픽이 전달된다. 노드 s2에서 노드 e2로는 기존의 TCP와 호환성을 시험하기 위한 배경 트래픽으로서 TCP-Reno를 사용한 FTP 트래픽이 전달된다. 노드 s3에서 노드 e3으로는 병목 링크의 가용대역 변화에 따른 현상을 시험하기 위해 배경 트래픽으로서 1Mbps의 UDP 트래픽을 연결하였다. TCP와 UDP 패킷의 크기는 1000바이트이고, UDP패킷은 1Mbps/CBR로 발생한다.

표 1은 그림 5의 네트워크를 이용한 시뮬레이션 시나리오이다. 그림 6은 표1의 시나리오에 따른 시뮬레이션 결과로 (a)는 TCP-CWL의 혼잡윈도우 크기변화를 나타내며, (b)는 TCP-Reno의 혼잡윈도우 크기변화를 나타낸다. 시나리오 구간 1에서 TCP-Reno가 대역을 독점적으로 사용하여 정상상태에 도달하였다. 구간 2는 TCP-Reno 가 선점한 대역을 TCP-CWL이 추가되어 대역을 공유하는 상황으로서, TCP-CWL은 대역제한 기능이 동작하지 않고 TCP-Reno와 동일하게 동작한다. 따라서 두 종류의 트래픽이 대역을 공유할 때 공평성에 문제가 없다. 구간 3은 TCP-CWL 트래픽이 대역



(a)

(b)

그림 6. 혼잡윈도우 크기, (a) TCP-CWL과 (b) TCP-Reno

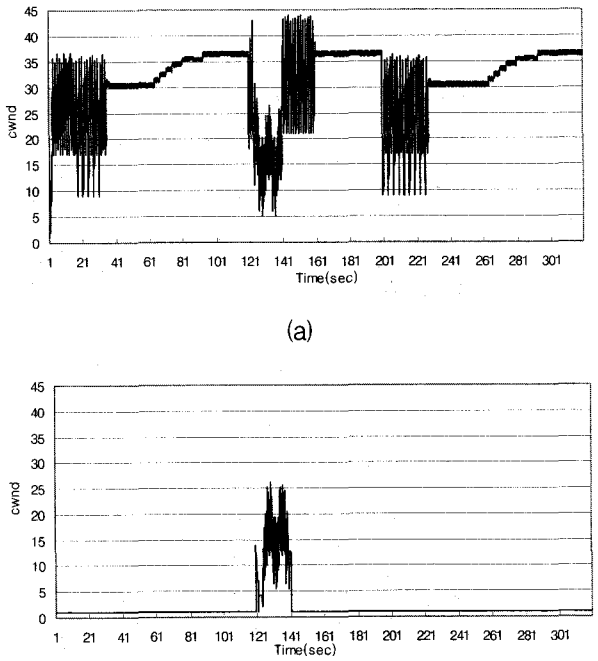
Fig. 6. Congestion window size, (a) TCP-CWL and (b) TCP-Reno.

표 2. 시뮬레이션 시나리오

Table 2. Simulation scenario.

구간	시간(sec)	s1 → e1 TCP-CWL	s2 → e2 TCP-Reno	s3 → e3 UDP(1Mbps)
1	0 - 60	On	Off	On
2	60 - 120	On	Off	Off
3	120 - 140	On	On	Off
4	140 - 200	On	Off	Off
5	200 - 260	On	Off	On
6	260 - 320	On	Off	Off

을 독점하는 상황으로서 제안된 알고리즘을 거쳐 정상상태의 TCP-CWL 기능이 동작한다. 이때 혼잡윈도우의 크기가 손실이 발생하지 않는 크기로 제한되어 안정된 상태로 유지됨을 확인할 수 있다. 구간 4는 구간 2의 반대상황으로 정상상태의 TCP-CWL이 대역을 선점한 상황에서 TCP-Reno 트래픽이 추가되어 대역을 공유해야 하는 상황이다. TCP-CWL의 대역제한 기능이 동작치 않을 때와 같이 TCP-Reno와 동일한 동작으로 공평성에 문제가 없다. 구간 5의 결과는 가용대역이 감소하더라도 CWL관련 변수를 갱신하지 않는 선택사양을 택한 경우로서, 대역을 공유하는 상황이 끝나면 이전의 정상상태에서 결정되었던 CWL관련 변수가 유효하여



(a)

(b)

그림 7. 혼잡윈도우의 크기, (a) TCP-CWL 과 (b) TCP-Reno

Fig. 7. Congestion window size, (a) TCP-CWL and (b) TCP-Reno.

즉시 정상상태로 복구한다. 가용대역이 감소할 때 CWL 관련 변수를 갱신하는 경우는 그림 7에서 설명한다. 구간 6은 모든 TCP트래픽이 없는 상태이며, 구간 7은 TCP-CWL 트래픽만 존재하는 상태로서, 트래픽이 중단되어도 연결이 해제되지 않으면 CWL관련 변수도 유효하여 구간 5의 경우와 같이 즉시 정상상태가 된다. 구간 8은 가용 대역이 4Mbps에서 5Mbps로 커진 상황에서 cwnd_cwl 갱신으로 확보된 대역을 정상적으로 활용할 수 있음을 보인다.

표 2는 가용대역이 감소하는 경우에도 cwnd_cwl을 갱신하는 경우의 혼잡윈도우 변화를 관찰하기 위한 시뮬레이션 시나리오이다.

그림 7은 표2의 시나리오에 따른 시뮬레이션 결과로 (a)는 TCP-CWL의 혼잡윈도우 크기변화를 나타내며, (b)는 TCP-Reno의 혼잡윈도우 크기변화를 나타낸 그림이다. 구간 1은 TCP-CWL이 대역을 독점적으로 사용하여 정상상태에 도달한 경우이다. 구간 2는 가용대역이 4Mbps에서 5Mbps로 증가하였으며, cwnd_cwl도 갱신되어 확보된 대역을 정상적으로 활용함을 보인다. 구간 3은 TCP-Reno와 TCP-CWL이 대역을 공유하는 상황으로서 두 트래픽이 대역을 공평하게 사용함을 확

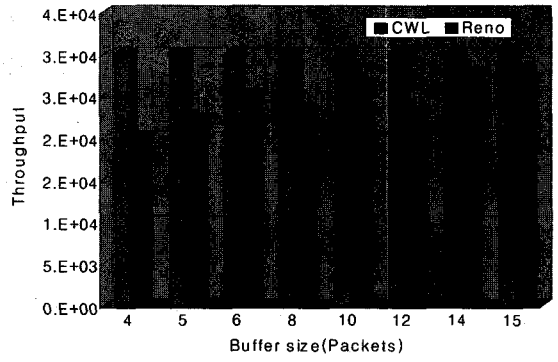


그림 8. 정상상태 전송률
Fig. 8. Steady state throughput.

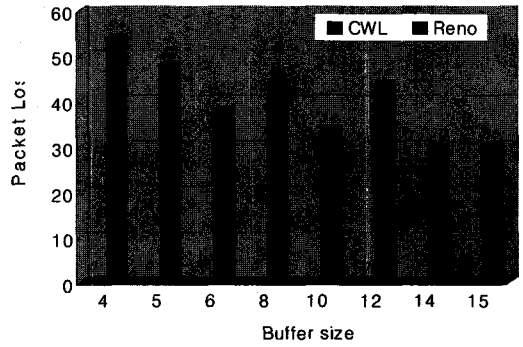


그림 9. 정상상태 패킷 손실 수
Fig. 9. Packet loss count at steady state.

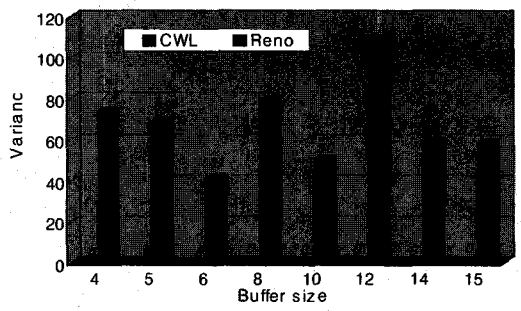


그림 10. 혼잡윈도우 크기의 분산값
Fig. 10. Variance of congestion window size.

인한다. 이때 TCP-CWL은 혼잡윈도우 제한 기능이 동작치 않으며 기존의 TCP-Reno와 동일하게 동작한다. 구간 4는 TCP-CWL이 대역을 독점적으로 사용하는 상황으로 초기 CWL관련 변수 결정과정을 포함한다. 가용대역 감소에 대한 갱신기능이 추가되어 그림 4와 달리 CWL관련 모든 변수가 초기화되므로 초기 결정과정을 포함한다. 구간 5는 여전히 TCP-CWL이 대역을 독점적으로 사용하나 가용대역이 감소하여 CWL관련 모든 변수가 초기화되는 과정을 포함한다. 구간 6은 구간 2와 동일하게 가용 대역이 4Mbps에서 5Mbps로 커진 상황에서 cwnd_cwl 갱신으로 확보된 대역을 정상적으로 활용할 수 있음을 보인다.

그림 8, 9, 10 은 제안된 방안과 TCP-Reno의 특성을 시뮬레이션 결과로 비교하였다. 시뮬레이션 환경은 그림 3의 네트워크 구성에서 노드 s1과 노드 s2에 대해 병목대역을 독점하도록 서로 다른 시간에 트래픽을 가하였으며, 각각의 연결이 설정된 후 500sec 경과 후 500sec간의 정상상태 데이터를 수집했다. 먼저 그림 8은 TCP-CWL과 TCP-Reno의 병목 노드 버퍼 크기에 대한 정상상태 성능을 비교한 시뮬레이션 결과이다. 알려진 바와 같이 TCP-Reno는 버퍼 제어가 성능에 영향을 미침을 알 수 있다. TCP-CWL은 버퍼 크기와 무관하게 TCP-Reno보다 우수한 성능을 나타낸다.

TCP-Reno의 AIMD기반의 혼잡제어 방법은 주기적으로 혼잡에 의한 손실이 발생한다. 그림 9는 Reno의 병목 노드 버퍼 크기에 대한 혼잡에 의한 패킷손실의 시뮬레이션 결과이다. 그러나 TCP-CWL은 버퍼 크기와 무관하게 혼잡에 의한 손실은 발생하지 않는다.그림 10은 Reno의 병목 노드 버퍼 크기에 대한 혼잡윈도우 크기의 분산값을 나타낸 그림이다. 혼잡윈도우의 크기 변화는 송신단의 입장에서는 가용대역의 변화와 같다. 따라서 그림과 같은 변화율은 안정된 대역이 필요한 서비스에 적합지 못한 특성이다. 그러나 TCP-CWL은 분산값이 0에 근접하여 버퍼 크기와 무관하게 혼잡윈도우의 크기가 안정적인 값을 나타낸다.

V. 결론 및 향후 연구

고정대역이 보장되는 네트워크를 효과적으로 활용하기 위해 TCP의 혼잡윈도우(cwnd)를 제한하여 정상상태의 TCP 특성을 개선하는 방법이 논의되고 있다. 그러나 TCP RUDE와 TCP Clamping 등의 기존의 연구는 별도의 제어기능이 필요하거나 특정 네트워크에만 적용가능하다는 문제가 있다.

본 논문에서는 TCP의 데이터 발생률을 결정하는 혼잡윈도우 크기를 버퍼에 데이터 축적이 일어나지 않고, 병목구간의 대역을 최대한 활용할 수 있는 크기로 제한하는 혼잡제어방법을 제안하였다. 제안된 방법은 TCP 연결 고유의 지연정보인 RTT를 이용하므로 별도의 제어기능이 불필요하며, 네트워크 상황에 따라 동작 상태가 결정되므로 네트워크의 종류와 무관하게 적용할 수 있다.

시뮬레이션을 통해 대역제한 또는 고정대역 네트워

크에서 혼잡윈도우의 제한으로 손실 없이 고정 대역을 유지하고, 최선형 데이터와 대역을 공유하는 경우에도 호환성을 유지함을 확인할 수 있었으며, 가용대역의 변화도 적절히 반영됨을 확인할 수 있었다. 또한 제안된 방법의 성능은 정상상태에서 버퍼 크기와 무관하게 TCP-Reno 에 비해 우수함을 알 수 있었다. 또한 제안된 방법의 성능이 항상 일정하여 예측이 가능하고, 구조적으로 발생하는 손실과 혼잡윈도우 크기의 분산이 0 또는 0에 근사하여 안정된 전달특성으로 이 나타남을 확인할 수 있다. 기존 AIMD기반의 TCP는 고정대역이 제공되어도 그 장점을 활용할 수 없었으나, 본 논문의 TCP-CWL의 적용으로 멀티미디어 서비스를 비롯한 다양한 응용에 적용할 수 있다.

본 연구 결과를 바탕으로 진행할 향후 연구과제로는 본 논문에서 제안한 방법을 고속의 네트워크에서 적용할 수 있는 방법에 대한 연구와 cwnd_cwl 결정과정에 사용되는 적정 파라미터에 대한 연구가 수행되어져야 할 것이다.

참 고 문 헌

[1] Sung Back Hong, "QoS-based manageable NGN architecture," ITU-T NGN Technical Workshop, pp. 209-223, March 2005.

[2] A. Gurtov, "TCP performance in presence of congestion and corruption losses", Master's Theses, Department of Computer Science, University of Helsinki , December 2000.

[3] J. Semke, J. Mahdavi and M. Mathis, "Automatic TCP buffer tuning," ACM SIGCOMM'98, vol. 28, no.4, 1998.

[4] Eric Weigle and W. Feng, "A comparison of TCP automatic tuning techniques for distributed computing," HPDC'02, July 2002.

[5] G. Appenzeller, I. Keslassy, and N. Mckeown, "Sizing router buffers," ACM SIGCOMM'04, 2004.

[6] Eric Weigle and Wu-chun Feng, "Enhancing TCP performance for dedicated clusters and grids (Rude TCP)," [LA-UR 03-3822, LA-CC 03-058], [Online] Available: http://csag.ucsd.edu/individual/ehw/research.html

[7] Andrea Di Donato, Yee-Ting Li, Frank Saka and Peter Clarke, "Using QoS for high throughput TCP transport over fat long pipes," PFLDNet'04, Feb. 2004.

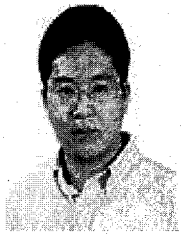
[8] C. Krasic, K. Li and J. Walpole, "The case for streaming multimedia with TCP", IDMS'01, pp213-218, Sep. 2001.

[9] Y. Zhang and D. Loguinov, "Oscillations and buffer overflows in video streaming under non-negligible delay," ACM NOSSDAV'04, June 2004.

[10] he network simulator ns-2. [Online]. Available: 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(정회원)
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
<주관심 분야: 이동인터넷, 이동통신 네트워크, 데이터통신>

이 재 용(정회원)-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>