종단간 네트워크 시스템에서 승인 압축 비율 제어를 이용한 TCP 성능 개선

김광준* · 윤찬호** · 김천석***

The Performance Improvement using Rate Control in End-to-End Network Systems

Gwang-Jum Kim* · Chan-Ho Yoon** · Chun-Suk Kim***

요 약

본 논문에서는 종단간 네트워크 시스템에서 전송 율 기반하의 혼잡 제어를 제안하여 양방향 TCP 연결의 성능을 개선한다. TCP의 패킷과 승인들은 종단 시스템의 공통 버퍼를 공유하기 때문에 소스 노드에 패킷과 승인이 집단적으로 도착되는 승인 압축 결과를 초래함으로서 불공성과 처리 율이 감소한다. 양방향 트래픽에 의한 처리율 감소는 매우 중요하다. TCP 비율제어를 이용한 백그라운드 트래픽이 2.5Mbps에서 5.0Mbps로 대역폭이 높아짐으로서 중간 노드의 혼잡이 어느 정도 해소되어 2.5Mbps에서 나타낸 것보다 왕복 지연 시간에 대한 처리 율이 적어짐으로서 연결의 효율성이 증가한다. 또한 백그라운드 트래픽을 7.5Mbps로 설정한 지터 처리 율에 대한 시뮬레이션 결과는 혼잡을 제어함으로서 2.5Mbps나 5.0Mbps에 발생한 지터 처리 율보다 개선됨으로서 제안된 비율 제어 방식의 연결 효율성의 성능 개선이 이루어짐을 알 수 있다.

ABSTRACT

In this paper, we extend the performance of bidirectional TCP connection over end-to-end network that uses transfer rate-based flow and congestion control. The sharing of a common buffer by TCP packets and acknowledgement has been known to result in an effect called ack compression, where acks of a connection arrive at the source bunched together, resulting in unfairness and degraded throughput. The degradation in throughput due to bidirectional traffic can be significant. Even in the simple case of symmetrical connections with adequate window size, the connection efficiency is improved about 20% for three levels of background traffic 2.Mbps,5.0Mbps and 7.5Mbps. Otherwise, the throughput of jitter is reduced about 50% because round trip delay time is smaller between source node and destination node. Also, we show that throughput curve is improved with connection rate algorithm which is proposed for TCP congetion avoidance as a function of aggressiveness threshold for three levels of background traffic 2.5Mbps, 5Mbps and 7.5Mbps. By analyzing the periodic bursty behavior of the source IP queue, we derive estimates for the maximum queue size and arrive at a simple predictor for the degraded throughput, applicable for relatively general situations.

키워드

Transfer rate-based Congestion Control, Ack Compression, Throughput Rate, Connection Efficiency

* 여수대학교 컴퓨터공학과
** 조선대학교 컴퓨터공학과
*** 여수대학교 전자통신공학과
접수일자 : 2005. 1. 12

Ⅰ. 서 론

컴퓨터 네트워크에서 사용되는 메시지 교환에 관한 규칙을 정의 하는 프로토콜들의 모음을 의미하는 전송제어 프로토콜 (TCP : Transmission Control Protocol)은 오늘날의 트랜스포트 계층 프로토콜에서 가장 광범위하게 사용되어 왔다. TCP의 가장 중요한 성분은 혼합 제어와 복구를 수행하기 위해 사용된 알고리즘으로서 지속적으로 확장[1,2] 연구되어 왔다. TCP는 신뢰성을 갖춘 연결 중심의 전이중 바이트 스트림 전송 계층 프로토콜로서 흐름 및 혼잡 제어를 지원하는 종단간 시스템 프로토콜이다. 컴퓨터 네트워크의 집합체인 인터넷과 같은 최선 노력(Best Effort) 네트워크 관리는 제어나 자원 예약의 개념이 존재하지 않으므로 네트워크가 부여되는 로드, 즉 네트워크 내부에 존재하는 전체 패킷 수를 제어하지 않기 때문에 네트워크 부하가 많은 최선 노력 네트워크는 혼잡이 발생한다. 네트워크가 혼잡 상태라면 호스트와 라우터 인터페이스의 버퍼가 오버플로우되면서 패킷 손실이 발생한다. 네트워크 종단점은 혼잡 제어를 통해 혼잡에 신속하게 대응함으로써 네트워크의 신뢰성을 보장하고 혼잡으로 인한 시스템 장애를 미연에 예방함으로서 원활한 정보 전송을 추구한다[7,8,9].

TCP는 응답신호가 회신하는 데 걸리는 시간을 계속 측정한다. TCP는 이러한 지연 평균(Round Trip Time, RTT) 및 이 평균을 기준으로 한 예상 지연 편차(Delay Variation)를 관리한다. 현재 지연이 평균에 비해 예상 편차보다 몇 배 이상 길어지면 TCP는 패킷 손실을 해결하기 위해 혼잡 회피를 적용한다. 이러한 알고리즘은 실제 모든 유,무선 네트워크에 적용되며 손상에 의한 패킷 손실은 매우 적기 때문에 패킷 손실은 발신지와 수신지 사이의 네트워크 중 어딘가에 정체가 발생했음을 의미한다는 것을 전제로 한다.

본 논문에서는 유선 네트워크에서 양방향 트래픽이 존재하는 곳에서 비동기 전송 모드 네트워크의 동적인 TCP 연결을 분석하며, 양방향 트래픽을 네트워크 경로를 통해 동일한 종단 노드 쌍 사이의 반대 방향에서 데이터를 전송하는 두 개 또는 그 이상의 TCP 연결로부터 생긴 트래픽 패턴을 사용한다. 한 방향에서 연결에 의해 전송된 TCP 세그먼트들은 반대 방향에서 연결에 대한 승인을 가지고 있는 동일한 물리적 경로를 공유하고, 패킷 승인들은 종단 시스템의 공통 버퍼를 공유할 뿐만 아니라 네트워크의 교환기와 라우터들을 공유한다. 이러한 공유는 승인 압축이라고 하는 결과를 초래하며, 승인 압축은 소스가 집단적으로 도착할 때 승인한다. 유선 네트워크상에서 감소된 처리 율을 향상시키기 위해 윈도우 기반 혼잡 회피 알고리즘 대신에 전송 율 기반 제어 알고리즘을 제안한다. TCP가 전송 율 기반 제어된 채널을 통해 운용될 때 비동기 전송 모드 네트워크에서 가변 비트 율에 의해 네트워크 노드에서 볼 수 있는 트래픽의 버스트성은 전송 율 기반 제어가 없는 채널을 통해 운용되는 것과 비교하고 유선 네트워크 노드에 의해 도입된 양방향 TCP 트래픽의 연결의 효율성으로 인해 전송 율 기반 제어된 유선 네트워크 환경에서 성능 처리 율이 향상됨을 입증하고자 하며, 혼잡제어 성능 평가 방식이 전송 율 기반하의 네트워크 트래픽의 물리적 모델링으로부터 얻은 시뮬레이션 결과를 통해 개선됨을 나타낸다.

Ⅱ. TCP 트래픽 모델

2.1 네트워크 모델

네트워크 배치 구조는 무선 네트워크에 의해 서비스 할 수 있는 가변 비트 율(VBR: Variable Bit Rate) 서비스와 같이 순차적으로 전송되는 비율 제어된 네트워크를 통해 통신하는 두 개의 종단 노드 i 와 j 를 그림 1과 같이 가장 간단한 배치 구조로 두 개의 한쪽방향 TCP 연결의 쌍으로 구성하고, 연결은 i 에서 j 로의 데이터 패킷 전송과 i 에서 j 로 다른 데이터 패킷이 전송된다는 것을 가정한다. 종단 시스템 효율성을 분석하기 위해 노드의 전송 율은 안정된 처리율 간격을 유지하며, 네트워크 각 방향의 노드 사이에 TCP 연결에 대한 대역폭이 고정되어 동작되고, 네트워크의 종단 노드의 경로를 통해 지연이 발생한다. 이것은 VBR 서비스[14,16]가 ATM 교환기내의 한정된 큐의 크기를 통해 전송하고 비율 제어된 네트워크를 통해 일정한 비율로 대역폭을 할당하기 때문에 발생한다.

한쪽 방향의 TCP 연결에 의해 전송된 데이터 세그먼트는 반대방향의 연결 전송로를 이용하여 전송 세그먼트에 대한 승인 세그먼트는 ATM 가상 채널을 공유한다. 종단 노드 시스템에서 지연에 대한 분석을 위해 그림 1의 네트워크 구조를 보다 단순한 배치 구조로 바꾸면 그림 2와 같다. 종단 노드들은 전형적인 점대점 링크에 의해 서로 연결되고 각각의 노드는 가상채널을 포함하고 있다. 종단 노드 사이의 TCP 연결 쌍은 네트워크에서 관측된 세그먼트의 간격을 통해 일정한 전송 율로 전송되며 네트워크 망의 부하 상태에 따른 전송지연이 생긴다. 종단노드 i에서 j까지의 TCP 연결 전송 데이터를 연결 i로 정의하고 j에서 i로의 TCP 연결

을 연결 j로 정의한다.

그림 1. 네트워크에서 양방향 트래픽 모델
Fig. 1 Model of two-way traffic in network

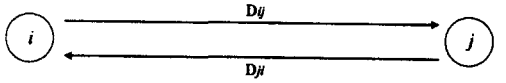


그림 2. 양방향 트래픽에 대한 단순화된 모델
Fig. 2 Simplified model for two-way traffic

종단 노드 i에서 j까지 전송된 TCP 세그먼트에 의해 발생되는 지연을 D_{ij}로 하고, 노드 j에서 i까지의 세그먼트에 의해 발생된 지연은 D_{ji}로 한다. 효율적인 분석을 위해 전송된 TCP 세그먼트의 크기는 일정하다고 가정하며, 대칭적인 네트워크 배치 구조에서 각각의 방향에서 종단 노드의 전송 율은 동일한 것으로 간주하며 전송 율은 초당 네트워크 전송로를 통해 전송된 TCP의 세그먼트의 수로 정의하며 ρ로 나타낸다.

종단 노드의 모델은 응용 계층, TCP 계층, IP 계층, 물리 계층의 4계층으로 구성되고, TCP 와 IP 계층 프로토콜은 비율 제어된 ATM 계층을 통해 운용되며, 적응 계층은 세그먼트의 분할과 재조립을 수행하고 셀 전송 율 제어 절차는 IP계층[18,19] 아래에서 이루어진다. 각각의 노드 내에는 두 개의 응용이 존재하는데 그 노드에서 발생하는 TCP 연결에 대한 데이터 소스로서 동작하는 송신 응용과 반대쪽 노드로부터 도착하는 데이터에 대한 상환으로서 동작하는 수신응용으로 구성되어 있다. 송신 종단 노드는 TCP 연결을 위해 발생된 데이터를 전송하고 수신 종단 노드는 송신 노드로부터 전송된 모든 데이터를 즉시 수신한다. 분리된 TCP 연결을 통해 통신하는 두 개의 응용이 있을 때 양방향 TCP 연결을 통해 데이터를 송신하고 수신하는 노드에서 송신응용과 수신응용에 대해 분석이 동일하게 적용된다. 공동 처리는 노드내의 모든 TCP 프로세스를 취급하고, TCP 프로세스는 네트워크로부터 데이터 세그먼트를 수신하고 송신 응용으로 데이터 세그먼트를 전달한다. 각각의 수신된 세그먼트에 대해 승인이 발생하고 이러한 승인은 출력 큐에 추가된다. 또한 TCP 프로세스는 반대편 노드로부터 승인이 수신될 때마다 전송 응용으로부터 한 개 이상의 데이터 세그먼트를 출력 큐에 큐잉함으로서 반대편 노드로 데이터 전송한다. TCP 처리는 윈도우[10,18] 크기가 최대 크기에 도달할 때까지 승인을 매번 수신할 때마다 윈도우를 증가시킴으로서 슬로우 스타트와 혼잡 회피 단계 동안에 송신 TCP의 윈도우 증가를 조절한다.

TCP 연결이 혼잡 회피 단계에서 거의 모든 시간을 소비하면 패킷이 손실되어 동적 연결과 승인 압축의 결과로 이어지지 않는다. 그러므로 본 논문에서 수행하고자 하는 분석은 승인 압축으로 인한 긴 주기를 통한 안정된 데이터의 흐름을 유지하는 충분한 버퍼가 있는 처리율이 민감한 환경에 적용할 수 있으므로 패킷 손실을 복구하는 TCP 메카니즘이다.

2.2 TCP 트래픽 승인 압축

순방향 연결의 세그먼트와 역방향 연결의 승인은 출력 링크의 전송 율로 서비스되는 IP 계층 공용 FIFO 큐를 공유한다. 이 공용 큐는 역방향 연결의 승인이 순방향 연결의 데이터 세그먼트 뒤의 큐에서 대기하는 동안 집단화되는 승인 압축 발생에 중요하다. 그러나 역방향 연결의 승인과 순방향 연결의 데이터 세그먼트가 공용 ATM 가상 채널을 공유하면 IP 서비스율과 링크 상에서 전송 율이 같다. IP 큐 밖의 서비스율이 높아지면 IP 계층의 IP 큐는 감소시킬 수 있으나 데이터가 큐 되는 ATM 계층 또는 적응 계층에 똑같은 집단화 결과가 발생한다. TCP 처리는 노드 i에 승인이 도착할 때마다 발생되며, 새로운 데이터 세그먼트에서 승인 결과의 처리는 노드 j에 전송되어지기 위해 IP 큐에 부가된다. 연결 i가 슬로우 스타트 단계에 있다면 그 노드의 윈도우는 하나씩 증가하고 부가적인 세그먼트는 IP 큐에 적재되고 나중의 세그먼트는 IP 큐의 이전의 세그먼트 뒤로 즉시 큐 된다. 그 결과로서 두개의 세그먼트는 연결 j을 위한 승인이 사이에 끼이지 않고 전송되어진다. 반대편 노드로부터 도착한 집단화된 승인에 대한 응답으로 IP 큐에 추가된 데이터 세그먼트는 승인이 사이에 끼이지 않고 집단적으로 전송된다. 이러한 동작은 각각의 연결의 전체적인 윈도우가 항상 단일집단으로 전송되어지도록 한다. IP 계층에 대해서 가정된 기능은 단순하며, 입력되는 트래픽에 대해 IP 계층은 하위 계층으로부터 TCP 처리까지 순방향 데이터이다. 그러므로 TCP 처리시간은 매우 짧은 것으로 추정할 수 있으며, IP 계층에서 입력되는 트래픽은 큐잉 처리과정을 필요로 하지 않는다. 이와는 반대로 출력되어지는 트래픽에 대해서는 링크 상에서 전송하기 위해서 큐가 설정되어야 하며 최악의 경우

전체적인 세그먼트의 윈도우가 반대편 종단으로부터 승인이 집단화 때문에 빠르게 연속되도록 출력 큐에 부가될 수도 있다. 그러므로 소스 노드로부터 패킷의 손실을 피하기 위해서는 IP 큐가 송신 TCP의 최대 윈도우 크기와 동일한 크기를 가져야 한다.

알 수 없는 상태에서 네트워크 전송을 시작하는 경우 TCP는 성공적인 전송과 정체 예방을 위해 네트워크 상태를 점검해야 한다. 슬로우 스타트 알고리즘은 전송을 시작할 때 또는 재전송 타이머가 감지한 손실을 복구한 후에 그와 같은 목적으로 사용된다. 그림 3은 슬로우 스타트 과정에서 새 데이터에 대해 응답하는 각각의 수신 응답에 대해 송신 노드 측에서 최대 세그먼트 크기만큼 정체 윈도우가 증가한다. 따라서 라운드 지연 시간마다 정체 윈도우가 두 배로 증가하게 되는데 정체 윈도우가 초기에 설정된 임계 값에 도달하거나 정체가 발생하면 슬로우 스타트가 종료되고 그 이후에는 혼잡 회피 알고리즘이 적용된다. 혼잡 회피에서 정체 윈도우는 왕복 지연 시간마다 세그먼트가 하나씩 증가하며, 정체 윈도우는 네트워크 상태에 따라 동적으로 조정된다. 세그먼트 손실이 감지되면 임계 값은 초기에 설정된 값의 절반으로 설정됨으로서 TCP는 네트워크 정체를 해결한다.

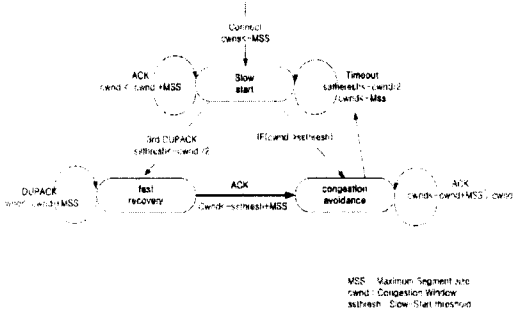


그림 3. TCP의 정체 제어
Fig. 3 TCP Congestion control

III. 전송 율 기반 TCP 연결 효율 알고리즘

양방향 트래픽 구조하에서 슬로우 스타트 처리 과정 동안에 발생되는 TCP 윈도우 증가에 대한 승인 압축은 반대쪽 연결의 승인에 의해서 서로 분리되는 TCP 연결의 세그먼트의 버스트를 일으킨다. 각각의 노드는 전체적인 윈도우 세그먼트를 단일 버스트로서 전송하고 이것은 반대쪽 연결의 데이터의 윈도우 증가를 위해 승인 뒤에 온다. 전송된 데이터 세그먼트의 크기는 동일하고 양방향에서 전송 율은 동일한 것으로 가정하고, 초당 세그먼트의 단위는 ρ이다. 승인의 전송 시간은 데이터 세그먼트의 전송 시간보다 상당히 적으므로 분석을 하기 위해서 승인의 전송시간을 0으로 설정한다.

두 연결의 윈도우 크기는 안정적이고 각각 노드 i와 j에서 발생하는 연결에 대한 윈도우 크기를 W_i와 W_j의 세그먼트로 나타낸다. 노드 i부터 j까지의 한쪽방향 링크를 채우기 위해 필요한 세그먼트의 수를 L_{ij}로 나타내고 반대편 링크에서는 L_{ji}로 하고 L_{ij}는 ρ와 D_{ij}의 곱으로 L_{ji}는 ρ와 D_{ji}의 곱으로 된다. $Q_i(t)$는 데이터 세그먼트만을 고려하고 승인에 의해 점유되는 공간을 무시한 채로 시각 t에서 노드 i출력 IP 큐의 점유기간이 비슷한 방법으로 $Q_j(t)$는 노드 j에 대한 점유 기간이다. $\tau_{i,k}$는 노드 j에 도착한 연결 i의 k번째 번잡 기간 동안에 노드 i에 의해 전송된 첫 번째 세그먼트 시간이다. 마찬가지로 $\tau_{j,k}$는 노드 i에 도착한 연결 j의 k번째 번잡 기간동안에 노드 j에 의해 전송된 첫 번째 세그먼트의 시간이다. 몇 개의 승인은 k번째 번잡 주기의 첫 번째 세그먼트가 노드 j에 도착할 때 노드 j의 출력 큐에서 세그먼트의 수 $Q_j(\tau_{i,k})$에 의해서 주어진 연결 j의 $\tau_{i,k}$시간의 처리과정이 끝나고 나서 번잡 기간 후에 함께 집단화 된다. 처리 율은 라운드 전송 파이프의 대역폭-지연 곱을 초과하는 윈도우 크기의 합이며, 노드 i의 윈도우 크기가 노드 j의 윈도우 크기와 노드 i에서 노드 j로 링크를 채우기 위해 필요한 세그먼트의 수 L_{ij}와 노드 j에서 노드 i로 링크를 채우기 위해 필요한 세그먼트의 수 L_{ji}의 합 보다 클 때 식(3.1)과 같다.

$$W_i > W_j + (L_{ij} + L_{ji}) \tag{3.1}$$

노드 j에서 연결 i의 승인의 최악의 경우 큐잉 지연은 W_i이므로 노드 i는 노드 j의 윈도우를 다 비우지 못하게 되며, W_i의 세그먼트로 구성된 연결 i의 각각의 번잡 주기는 집단화된 W_i승인의 순서와 함께 동반하여 다음 번잡 주기에 의해 즉시 뒤따라온다. ACK 프레임 전송 시간이 0이기 때문에 연결 i는 이 경우에 있어서 완벽한 처리 율을 달성할 수 있다. 그러나 라운드 전송 지연은 W_i의 세그먼트 전송시간과 같기 때문에 연결 j의 처리 율은 영향을 받는다.

연결의 번잡 주기는 주기적인 동작이 존재한 상태에서 윈도우 크기 W_i와는 별도로 분리되어서 공

간화 된다. 링크 용량에 비례하는 연결의 처리 율을 연결 효율성이라고 하며, 각각의 연결은 W_i세그먼트 전송시간 동안에 윈도우 세그먼트를 전송하므로 i노드와 j노드에서 연결의 효율성은 식(3.2)와 같다.

$$F_i = 1, F_i = W_j/W_i \tag{3.2}$$

연결의 k번째 번잡 주기의 첫 번째 세그먼트가 노드 j에 도착할 때 노드 j의 IP 큐의 점유 기간은 $Q_j(\tau_{i,k})$이므로 번잡 주기의 첫 번째 세그먼트에 대한 승인은 $\tau_{i,k}+Q_j(\tau_{i,k})/\rho$시간에 노드 j에서 출발하고 시간 t_1에서 노드 i로 도착하고, t_1 은 식(3.3)과 같다.

$$t_1 = \tau_{i,k} + \frac{Q_j(\tau_{i,k})}{\rho} + D_{ji} \tag{3.3}$$

시간 t_1에서 노드 i의 출력 IP 큐의 점유기간은 먼저 $\tau_{i,k} - D_{ij}$시간에 노드 i의 k번째 번잡 주기 전송을 시작함으로서 이루어지고 번잡 주기에 대한 첫 번째 ACK 프레임시간이 지난 후에 노드 i로 되돌아오고, 노드 i는 식(3.4)과 같은 세그먼트로 전송된다.

$$\rho(t_1-(\tau_{i,k}-D_{ij}))=Q_j(\tau_{i,k})+(L_{ij}+L_{ji}) \tag{3.4}$$

노드 i의 각각의 번잡 주기는 W_i세그먼트로 구성되고 노드 i의 출력 큐는 k번째 번잡 주기의 첫 번째 승인이 t_1시간에 되돌아 올 때 식(3.5)과 같은 세그먼트를 갖는다.

$$Q_i(t_1) = W_i - (Q_j(\tau_{i,k}) + L_{ij} + L_{ji}) \tag{3.5}$$

$Q_j(\tau_{j,k}) \le W_j$ 와 $W_i > W_j + (L_{ij}+L_{ji})$이므로 $Q_i(t_1)$은 0보다 크다. 그러므로 연결 j의 모든 승인은 버스트로서 노드 j로 되돌아가고 연결 i는 전송을 위해서 출력 큐에서 이용 가능한 데이터를 항상 가지고 있다. 다음 번잡 주기는 $t_1+Q_i(t_1)/\rho$시간에 노드 i에서 시작하고 $t_1+Q_i(t_1)/\rho+D_{ij}$시간에 노드 j로 도착한다. 그러므로 $\tau_{i,k}+1 = t_1+Q_i(t_1)/\rho+D_{ij}$과 같다. 식(3.3)과 식(3.5)으로부터 t_1과 $Q_i(t_1)$을 각각 대입하면 위식은 식(3.6)과 같이 된다.

$$\tau_{i,k+1} = \tau_{i,k} + \frac{W_i}{\rho} \tag{3.6}$$

유사한 방법으로 $\tau_{j,k+1}$은 식(3.7)과 같이 구할 수 있다.

$$\tau_{j,k+1} = \tau_{j,k} + \frac{W_i}{\rho} \tag{3.7}$$

식(3.6)는 $W_i > W_j + (L_{ij}+L_{ji})$인 경우에 적용할 수 있으며 $W_i < W_j - (L_{ij}+L_{ji})$인 경우에 대해서는 보완적이고 동일한 분석으로 취급한다. 보다 복잡한 경우를 고려하면 각 연결의 윈도우는 라운드 전송 지연에 다른 연결의 전체 윈도우를 더한 것으로 이루어진 파이프를 오버플로우 할 정도로 충분하게 크지 않은 것으로서 식 (3.8)과 같다.

$$W_j - (L_{ij}+L_{ji}) \le W_j \le W_j + (L_{ij}+L_{ji}) \tag{3.8}$$

식(3.8)은 식(3.6)에 의해 적용되지 않는 모든 경우에 대해서 적용된다. 이 경우에서는 연결의 번잡 주기가 주기적인 동작이 존재하나 동작은 보다 복잡하지 않는다는 것을 다시 한번 알 수 있다.

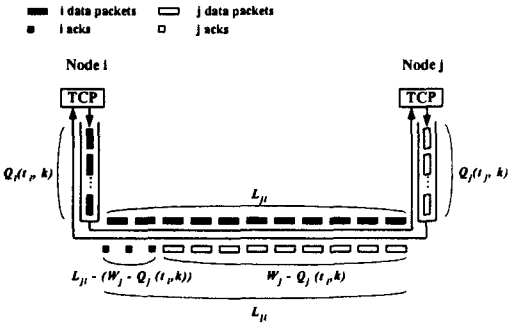


그림 4. 노드 j에 연결 i의 k번째 번잡주기의 첫 번째 세그먼트가 도착 하는 네트워크 동작
Fig. 4 Network behavior on the first segment of the kth busy period of connection i arrives at node j

연결 j의 $k+1$번째 번잡 주기 동안에 전송된 첫 번째 세그먼트가 노드 j에 도착할 때 노드 j에서 큐 크기인 $Q_i(\tau_{i,k+1})$을 결정할 필요가 있다. 이러한 큐는 연결 i의 $k+1$번째 번잡 주기 이전에 노드 j에 의해 수신된 집단화가 된다. 이러한 ACK 프레임 열은 연결 j의 m번째 번잡 주기 동안에 전송된 세그먼트를 인지한다. $\tau_{j,m}$은 연결 j의 m번째 번잡

주기에서 첫 번째 세그먼트가 노드 i에 도착한 시간이며, $\tau_{i,k}$로부터 $\tau_{j,m}$을 다음과 같이 결정한다. 그림 4와 같이 $\tau_{i,k}$시간에 노드 j는 m번째 번잡 주기의 $W_i - Q_j(\tau_{j,k})$을 전송한다.

그러므로 $\tau_{i,k} - (W_i - Q_j(\tau_{i,k})/\rho$시간에 m번째 번잡 주기의 전송을 시작하고 이러한 번잡 주기의 첫 번째 세그먼트는 D_{ji}의 지연 후에 노드 i에 도착한다. 그러므로 $\tau_{i,m}$은 식 (3.9)과 같다.

$$\tau_{i,m} = \tau_{i,k} - \frac{(w_j - Q_j(\tau_{i,k}))}{\rho} + D_{ji} \tag{3.9}$$

t_1을 노드 i가 k번째 번잡 주기의 전송을 완료했을 때 시간은 식(3.10)과 같다.

$$t_1 = \tau_{i,k} - D_{ij} + \frac{W_i}{\rho} \tag{3.10}$$

$(\tau_{i,m}, t_1)$간격 동안에 노드 i는 연결 j의 m번째 번잡 주기로부터 세그먼트를 수신하나 노드 i의 출력 큐는 비어있지 않은 상태로 남아 있으며, 노드 i에서 노드 j쪽에 의해 발생된 승인은 집단화된다. $(\tau_{i,m}, t_1)$간격 동안에 집단화된 승인의 수는 $mim(\rho(t_1 - \tau_{j,m}), W_j)$에 의해서 주어진다. 식(3.9)와 식(3.103)을 이용하여 $\rho(t_1 - \tau_{j,m})$은 식(3.11)과 같다.

$$\rho(t_1 - \tau_{j,m}) = (W_i + W_j) - (D_{ij} + D_{ji}) - Q_j(\tau_{i,k}) \tag{3.11}$$

그러므로 연결 i의 $k+1$번째 번잡 주기의 첫 번째 세그먼트에 의해 알 수 있는 큐 크기는 식(3.12)과 같다.

$$\begin{aligned} Q_j(\tau_{i,k+1}) &= mim((W_i + W_j) - (D_{ij} + D_{ji})\rho - Q_i(\tau_{i,k}), W_j) \\ &= mim((W_i + W_j) - (L_{ij} + L_{ji}) - Q_i(\tau_{i,k}), W_j) \end{aligned} \tag{3.12}$$

이러한 결과를 이용하여 노드의 연속적인 번잡 주기 사이에서 간격을 계산 할 수 있다. t_1을 연결 j의 k번째 번잡 주기에 대한 첫 번째 승인이 노드 i로 되돌아오는 시간을 나타낸다.

그러면 ACK 프레임노드 i에 도착할 때 t_1은 식(3.6)에서와 같다. 이러한 승인이 노드 i에 도착할 때 노드 i의 출력 큐는 비어 있거나 또는 비어 있지 않을 수도 있다. 첫 번째 $Q_i(t_1) < 0$경우로서 노드 i가 t_1시간에 노드의 k번째 번잡 주기에서 세그먼트의 전송을 완료하지 않았을 때 발생하는 경우로서 식(3.14)과 같다.

$$\frac{W_i}{\rho} > t_1 - (\tau_{i,k} - D_{ij}) \tag{3.14}$$

식(3.3)으로부터 t_1을 대입하면 식(3.14)은 식(3.15)과 같다.

$$W_i > (L_{ij} + L_{ji}) + Q_j(\tau_{i,k}) \tag{3.15}$$

위의 경우에서 노드 i는 연결 j의 전체적인 윈도우 W_j에 대해 압축된 승인을 가지고 k번째 번잡 주기의 전송 다음에 오고, 연결 i의 $k+1$번째 번잡 주기 다음에 온다. 그러므로 승인의 전송 시간을 무시함으로서 노드 i는 $\tau_{i,k} + W_i/\rho - D_{ij}$ 시간에 $k+1$번째 번잡 주기를 시작할 것이고, $t_1 = \tau_{i,k+e}(W_i/\rho) - D_{ij}$ 시간에 이 번잡 주기의 마지막 세그먼트 전송을 완료한다. 이 번잡 주기의 첫 번째 세그먼트는 $\tau_{i,k+1} = \tau_{i,k} + W_i/\rho$ 시간에 노드 j에 도착한다. $k+1$번째 번잡 주기가 연결 j의 세그먼트 W_j에 대해 압축된 ACK 프레임다음에 옴으로서 이러한 번잡 주기의 첫 번째 패킷은 노드 j에서 $Q_j(\tau_{i,k+1}) = W_j$세그먼트 큐 크기임을 알 수 있고 동일한 결과는 식(3.13)과 식(3.15)를 조합함으로서 얻을 수 있다. 그러므로 $k+1$번째 번잡 주기에 대한 첫 번째 응답은 $\tau_{i,k+1} + W_{j/\rho}$시간에 출발하고 $t_2 = \tau_{i,k+1} + W_j/\rho + D_{ji}$시간에 노드 j에 도착한다. $W_i \le W_j + (L_{ij} + L_{ji})$임으로 $t_2 \ge t_1$임을 쉽게 계산할 수 있고, 노드 i가 연결 i의 $k+1$번째 번잡 주기의 전송을 완료한다. 그러므로 $k+2$번째 번잡주기는 t_2시간에 시작되고, D_{ij}지연 후에 노드 j에 도착한다. 그러므로 $\tau_{i,k+2}$는 식(3.16)과 같은 식을 얻을 수 있다.

$$\begin{aligned} \tau_{i,k+2} &= t_2 + D_{ij} \\ &= \tau_{i,k+1} + \frac{W_j}{\rho} + D_{ji} + D_{ij} \\ &= \tau_{i,k} + \frac{(W_i + W_j)}{\rho} + (D_{ij} + D_{ji}) \end{aligned} \tag{3.16}$$

$Q_i(t_1) = 0$인 경우로서 이 경우는 $W_i \le W_j + (L_{ij} + L_{ji})$일 때만 발생한다. 이 경우에서 연결 i의 $k+1$번째 번잡 주기는 t_1시간에 노드 i로 출발하고, 그것의 첫 번째 세그먼트는 $t_1 + D_{ij}$시간에 노드 j에 도착한다.

그러므로 $\tau_{i,k+1}$은 식(3.17)과 같다.

$$\tau_{i,k+1} = t_1 + D_{ij} = \tau_{i,k} + \frac{Q_{j(\tau_{i,k})}}{\rho} + (D_{ij} + D_{ji}) \quad (3.17)$$

이러한 번잡 주기에 대한 첫 번째 응답은 시간 $\tau_{i,k+1} + Q_j(\tau_{i,k+1})/\rho$시간에 노드를 출발하고 식(3.18)와 같은 시간에 노드 i에 도착한다.

$$t_3 = \tau_{i,k+1} + \frac{Q_i(\tau_{i,k+1})}{\rho} + D_{ji} \quad (3.18)$$

식(3.17)에 $\tau_{i,k+1}$을 대입하고 식(3.13)에 $Q_j(\tau_{i,k+1})$을 대입하면 t_3는 식(3.19)과 같다.

$$t_3 = \tau_{i,k} + \frac{W_i + W_j}{\rho} + D_{ji} \quad (3.19)$$

노드 i는 t_3시간에 $k+1$번째 번잡 주기의 전체를 전송함으로서 t_3에서 $k+2$번째 번잡 주기가 시작된다. 이러한 번잡 주기의 첫 번째 세그먼트는 $t_3 + D_{ij}$시간에 노드 j에 도착한다. 따라서 $\tau_{i,k+2}$는 식(3.20)과 같다.

$$\begin{aligned} \tau_{i,k+2} &= t_3 + D_{ij} \\ &= \tau_{i,k} + \frac{W_i + W_j}{\rho} + (D_{ij} + D_{ji}) \end{aligned} \quad (3.20)$$

$W_j - (L_{ij} + Lji) \le W_i \le W_j + (L_{ij} + L_{ji})$인 경우에 대해 TCP 연결의 주기적인 동작이다. 윈도우 크기는 $W_i = W_j = 4$개의 세그먼트로 가정하고 링크 상에서 각각의 세그먼트 시간은 1초로 한다. 각기 방향에서 네트워크의 지연은 2초로서 L_{ij}와 L_{ji}는 2초이다. 그러므로 각기 연결에 허용하는 네 개의 세그먼트는 윈도우 크기는 한쪽 방향 배치 구조에서 노드의 최대 처리 율을 달성할 수 있다. 양방향 트래픽에서 처리 율은 연결 사이의 상호동작 때문에 지속적으로 감소한다.

IV. 시뮬레이션 결과 및 성능 개선

4.1 전송 율 기반하의 TCP 버전에 따른 대역폭

유선 네트워크에서 다양한 TCP 버전, 즉 Tahoe, Reno, Vegas, TCP Rate라고 부르는 전송 율 기반 알고리즘과 혼잡 제어를 적용한 다양한 형태를 살펴보고자 한다. 몇 개의 세그먼트 손실이 발생한 경우 빠른 재전송 및 복구가 매우 효과적인 해결책이 된다. 발신노드는 수신노드에서 수신된 모든 세그먼트에 대해 응답신호를 수신한다. 한두 개의 세그먼트가 손실되면 발신노드는 마지막으로 보낸 응답과 동일한 응답 번호를 중복 응답을 받는다. 중복된 응답을 비롯하여 세 개의 응답을 수신한 발신기는 재전송 타이머가 종료할 때까지 기다리지 않고 손실된 세그먼트들을 전송하고 아직 응답되지 않은 데이터 전송을 다시 시작하면서 파이프를 채우고 응답신호를 기다린다. 이렇게 함으로서 발신노드가 중복된 응답을 수신하면 데이터가 순서에서 벗어나 도착하거나 손실되었음을 의미한다.

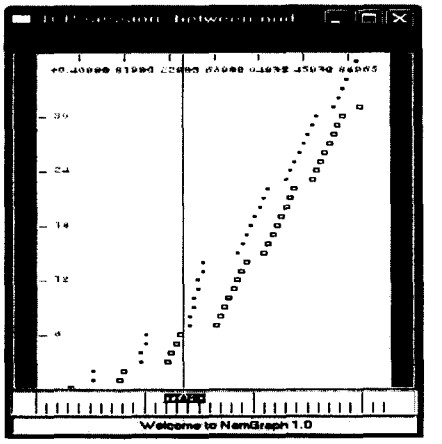


그림 5. TCP Tahoe 곡선
Fig. 5 The curver of TCP Tahoe

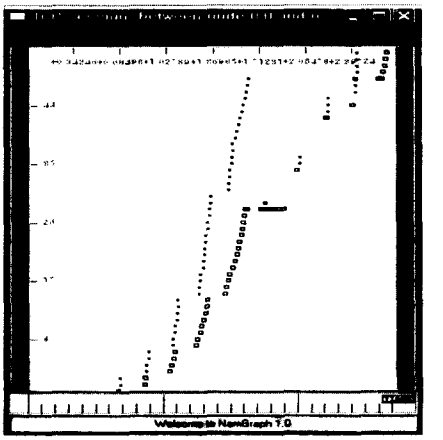


그림 6. TCP Reno 곡선
Fig. 6 The curve of TCP Reno

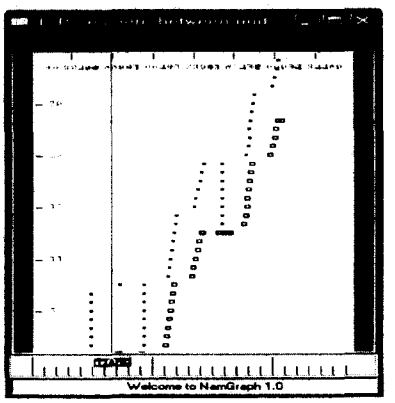


그림 7. TCP New-Reno 곡선
Fig. 7 The curve of TCP New-Reno

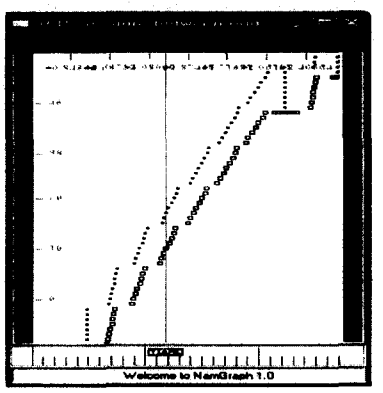


그림 8. TCP Sack 곡선
Fig. 8 The curve of TCP Sack

그림 5, 6, 7, 8는 TCP 각 버전에 대한 패킷의 흐름 및 TCP에 따른 대역폭의 변화를 보여준다. 그림 5의 시뮬레이션 결과는 TCP Tahoe 버전의 슬로우 스타트 방법을 이용한 것으로서 윈도우 크기는 4로 설정하였고 TCP 및 CBR 트래픽 서비스를 적용한 경우의 결과를 나타내고 있다. 위 그림에서 나타낸바와 같이 노드 사이의 혼잡이 발생하지 않음으로서 ACK 승인 압축이 정상적으로 처리됨을 알 수 있다. 이는 혼잡제어의 윈도우 크기 및 슬라이딩 윈도우가 노드사이의 연결을 설정함에 있어 노드의 부하가 발생하지 않음을 의미한다. 그림 6은 TCP Reno를 적용한 경우의 시뮬레이션 결과로서 윈도우 크기를 8로 설정하여 TCP 및 CBR 트래픽 서비스를 전송함으로서 패킷 전송시 에러가 발생할 경우 빠른 재전송 및 복구를 담당한다. 결과에서 알 수 있듯이 인해 TCP 트래픽의 패킷 순서 28번째가 1.56×10^{-3}시간부터 1.78×10^{-3}까지 재전송하는 형태를 취하고 있는데, 송신 노드에서 패킷이 드롭되어 정상적으로 전송되지 않았거나 또는 혼잡으로 인한 ACK 응답 신호가 송신 노드의 타임스탬프의 타임머의 설정시간 이내에 전송 패킷에 대한 응답신호가 도착하지 않아서 발생한 경우이다. 또한 윈도우 크기를 4로 설정한경우에 비해 윈도우 크기를 8로 증가 시킴으로서 노드간의 패킷 전송에 대한 부하가 증가함으로서 혼잡이 발생할 수 있다. 그림 7은 윈도우 크기를 8로 설정한 상태에서 다중 패킷 손실 문제를 처리하는 TCP NweReno의 버전을 사용한 경우의 시뮬레이션 결과를 나타내고 있다. 결과에서 알 수 있듯이 TCP 트래픽 패킷의 재전송과 CBR 패킷이 재전송하는 형태를 취하고 있으며, TCP 트래픽의 14번째 패킷이 1.34×10^{-3}시간부터 1.86×10^{-3}시간까지 재전송이 이루어짐을 알 수 있으며 동시에 CBR 트래픽의 첫 번째 패킷부터 여덟 번째 패킷까지 다중 패킷이 재전송하는 형태를 취하고 있음을 알 수 있다. 이는 다중 패킷 손실에 대한 빠른 재전송 및 복구를 의미한다. 그림 8도 역시 윈도우 크기를 8로 설정하여 혼잡 회피 기법을 적용하는 TCP SACK의 버전을 적용한 경우의 시뮬레이션 결과를 나타내고 있다. 48번째 TCP 패킷 순서가 3.15×10^{-3} 시간부터 3.89×10^{-3}까지 재전송의 형태를 취하고 있다. 그림 8의 결과는 그림 6과 7보다 늦은 시간에 재전송이 이루어지고 있는데 이는 전송 율 기반하에서 혼잡 회피와 슬라이딩 윈도우를 적용함으로서 TCP 패킷의 재전송 및 빠른 복구가 효율적으로 수행되고 있음을 알 수 있다. 실제적으로 이러한 경우에 노드사이의 혼잡이 발생하지 않으면서 동시에 혼잡을 회피하는 기법은 슬라이딩 윈도우의 slow start와 패킷 전송시 에러가 발생할 경우 빠른 재전송 알고리즘을 적용한 TCP Tahoe와, 빠른 재전송 및 복구를 담당하는 TCP Reno, TCP Reno의 다중패킷 손실문제를 처리하는 TCP NewReno, 혼잡회피 기법을 사용하는 TCP Vegas, 선택적 응답 기법의 SACK 등 각각의 TCP 버전에 따른 여러 가지 기법이 있다.

4.2 백그라운드 트래픽에 대한 처리 율 곡선

양방향 트래픽 하에서 각기 노드에서 출력 IP 큐의 최대 크기에 대한 식을 유도한다. 노드 j에서 최대 큐 크기는 연결 i노드의 번잡 주기의 첫 번째 세그먼트가 노드 j에 도착할 때 알 수 있다. 게다가 큐 크기는 노드에서 발생하는 연결의 윈도우 크기를 절대로 초과하지는 않는다. 안정적인 상태에

서 양방향 트래픽하의 노드 i의 IP 큐의 최대 점유 기간 $Q_i$max는 식(4.1)과 같다.

$$Q_i,\max \leq \begin{cases} W_i - (L_{ij} + L_{ji}), & W_i > W_j + (L_{ij} + L_{ji}) \\ (W_i + W_j) - (L_{ij} + L_{ji}), & W_j - (L_{ij} + L_{ji}) \leq W_i \leq W_j + (L_{ij} + L_{ji}) \\ W_i, & otherwise \end{cases} \quad (4.1)$$

두 번째 항은 직접적으로 식(3.8)의 결과로부터 발생하고 세 번째 경우는 큐 크기가 노드에서 발생하는 연결의 윈도우 크기를 초과할 수 없다는 것이다. 첫 번째 항을 고려하면 노드 i에서 최대 큐 크기는 연결 j의 번잡 주기의 첫 번째 세그먼트가 노드에 도착할 때 알 수 있다. $\tau_{j,m}$은 연결 j의 m번째 번잡 주기의 첫 번째 세그먼트가 노드 i에 도착할 때 시간이다. 이러한 시간에서 큐 크기는 노드 i에서 큐 된 세그먼트의 수에 의해 할 수 있고 네트워크의 나머지에서 연결 i에 속해 있는 세그먼트 수와 ACK 프레임수를 더한 것은 위도우 크기 W_i와 같다.

연결 j의 m번째 번잡 주기의 첫 번째 세그먼트가 $t_1 = \tau_{j,m} - D_{ji}$시간에 노드 j에 의해서 전송된다. t_1시간에 노드 j출력 큐는 연결 i의 어떠한 승인도 포함하지는 않는다. $t_1, \tau_{j,m}$간격 동안에 채널 i로부터 전체적인 세그먼트$(t_1, \tau_{j,m})$=$D_{ij}\rho$=L_{ji}는 노드 j에 도착된다. 더욱이 연결 i의 L_{ji}는 노드 j에 도착된다. 더욱이 연결 i의 L_{ij}세그먼트는 $\tau_{j,m}$시간에 노드 i로부터 노드 j까지 링크 상에서 전송된다. 그러므로 $\tau_{j,m}$에서 전체 적인 시스템에서 연결 i의 전체적인 세그먼트와 승인은 식(4.2)과 같다.

$$Q_i(\tau_{j,m}) = W_i - (L_{ij} - L_{ji}) \quad (4.2)$$

양방향 배치 구조에서 전송 율이 다른 보다 일반적인 비대칭적인 경우로 확장하여 적용하고 ρ_i와 ρ_j을 각각 연결 i와 j의 전송 율을 나타내면 ACK 프레임압축의 생성에 대한 조건은 식(4.3)과 같다.

$$\frac{W_i}{\rho_i} + \frac{W_j}{\rho_j} > D_{ij} + D_{ji} \quad (4.3)$$

즉 두 연결의 윈도우의 전송 시간의 합은 발생시키는 ACK 프레임압축에 대한 라운드 전송지연보다 반드시 크다. 이러한 필수적인 조건이 만족하면 비대칭적인 경우에서 연결 처리 율에 대한 식(3.20)은 식(4.3)과 같다. 비대칭적인 링크 율을 가지고 있는 양방향 트래픽 구조에서 연결 i의 효율성 F_i는 식(4.4)과 같다.

$$F_i = \begin{cases} (w_i/\rho_i) > (w_i/\rho_i) + (D_{ij} + D_{ji})\text{일때}, & 1 \\ (W_j/\rho_j) - (D_{ij} + D_{ji}) \leq (W_i/\rho_i) \geq (W_j/\rho_i)\text{일때}, & \frac{w(W_i/\rho_i)}{((W_i/\rho_i) + (W_j/\rho_j) + (D_{ij} + D_{ji})} \\ otherwise, & \frac{(W_i/\rho_i)}{(W_j/\rho_j)} \end{cases} \quad (4.4)$$

비슷하게 식(4.4)을 확장함으로서 종단 노드에서 최대 큐 크기를 결정할 수 있기 때문에 안정된 상태에서 비대칭적인 링크 율을 가지고 있는 양방향 트래픽하에 노드 i의 IP 큐의 최대 점유기간 Q $_i$,max는 식(4.5)와 같다.

$$Q_i, max \leq \begin{cases} (W_i/\rho_i) > (W_j/\rho_j) + (D_{ij} + D_{ji})\text{일때}, W_i - (D_{ij} + Dji)\rho \\ (W_j/\rho_j) - (D_{ij} + D_{ji}) \leq (W_j/\rho_j) + (D_{ij} + D_{ji})\text{일때}, \\ (W_i/\rho_i) + (W_j/\rho_j) - D_{ij} - D_{ji})\rho \\ otherwise, \quad W_i \end{cases} \quad 4.5)$$

식 (4.4)는 두 방향에서 전송 율이 다르다면 고대역폭 연결의 효율성은 저대역폭 윈도우 전송 시간에 의해 제한된다는 것으로 이것은 심각한 성능의 감소를 초래한다. 이것을 설명하기 위해 윈도우 크기가 $W_i = W_j = 128$ Kbytes 인 것을 선택하고 링크 지연들은 $D_{ij} = D_{ji} = 5ms$인 예를 고려한다. 연결 j가 155Mbits/s의 안정 율로 할당된다. 그림 6, 7, 8은 연결 i에 할당된 율의 함수로서 연결 i와 j의 효율성의 변동으로서 TCP 비율 제어 백그라운드 트래픽의 레벨에 대한 왕복 지연의 지터에 대한 처리 율을 나타내고 있다.

그림 9는 TCP 비율 제어 2.5Mbps 백그라운 트래픽에 대한 왕복 지연의 지터 처리 율을 나타낸다. 그림 9의 결과에서 알 수 있듯이 중간노드의 혼잡으로 인한 송신 노드의 패킷에 대한 지터가 거의 모든 시간에서 지터 처리 율이 높아짐을 알 수 있다. 이는 송, 수신 노드 사이의 혼잡으로 인한 왕복 지연 시간의 증가가 지터의 처리 율을 감소시킴으로서 연결의 효율성이 낮아진다. 그림 10은 5.0Mbps의 백그라운드 트래픽에 대한 지터 처리 율을 나타내는데 5군데의 지터 처리 율이 높아지는데, 이는 TCP 비율제어를 이용한 백그라운드 트

래픽이 2.5Mbps에서 5.0Mbps로 대역폭이 높아짐으로서 중간 노드의 혼잡이 어느 정도 해소되어 2.5Mbps에서 나타낸 것보다 왕복 지연 시간에 대한 처리 율이 적어짐으로서 연결의 효율성이 증가함을 알 수 있다. 그림 11은 백그라운드 트래픽을 7.5Mbps로 설정한 지터 처리 율에 대한 시뮬레이션 결과를 나타내고 있다. 이는 중간노드에 제안된 TCP 비율 제어를 이용하여 혼잡을 제어함으로서 왕복 지연 시간에 대한 지터 처리 율이 단순히 두세 군데에 높게 나타남으로서 2.5Mbps나 5.0Mbps에 발생한 지터 처리 율보다 개선됨을 알 수 있다. 실제적으로 중간노드의 왕복 지연 패킷에 대한 연결 i의 이용 가능한 대역폭이 낮을 때 연결 j는 그것의 이용 가능한 링크 대역폭의 몇 퍼센트만이 이용할 수 있다. 예를 ρ_i가 20 Mbps 일 때 연결 j의 처리 율이 혼잡이 발생했을 경우의 왕복 지연 시간이 증가함을 볼 수 있다.연결 i의 대역폭이 증가함으로서 연결 j의 처리 율은 약 70Mbps가 될 때까지 선형적으로 증가한 반면에 연결 i의 처리 율은 무의미한 값으로 유지된다. 이 지점을 넘어서 처리 율은 방정식의 두 번째 경우에 의해서 결정되고 연결 j에서 증가하는 처리 율은 낮은 비율에서 발생한다. 연결 i의 전송 율이 70 Mbps에서 155 Mbps로 증가함으로서 55%에서 75%로 연결 j의 효율성이 증가하는데 이는 왕복 지연 시간을 줄이는 결과를 초래한다.

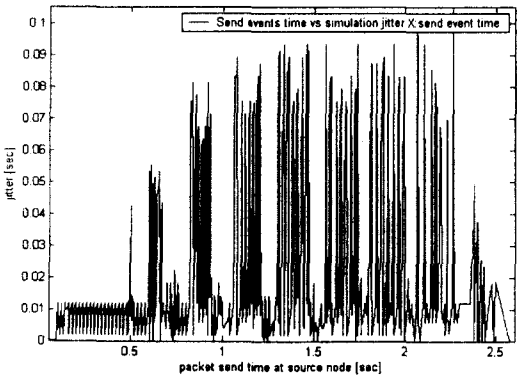


그림 9. TCP 비율 제어 2.5Mbps 백그라운드 트래픽에 대한 지터의 처리율

Fig. 9 Jitter throughput for TCP Rate-controll 2.5Mbps background traffic

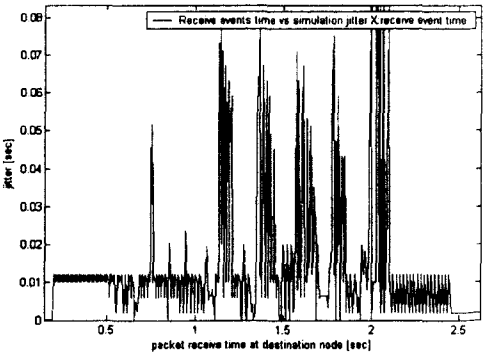


그림 10. TCP 비율 제어 5.0Mbps 백그라운드 트래픽에 대한 지터의 처리율

Fig. 10 Jitter throughput for TCP Rate-controll 5.0Mbps background traffic

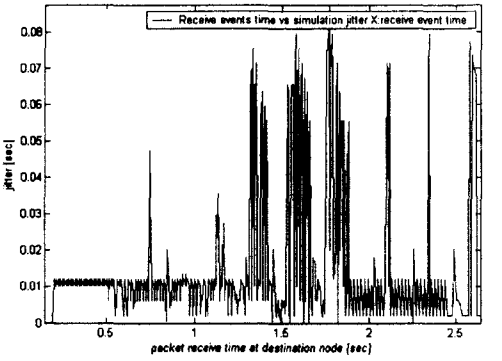


그림 11. TCP 비율 제어 7.5Mbps 백그라운드 트래픽에 대한 지터의 처리율

Fig. 11 Jitter throughput for TCP Rate-controll 7.5Mbps background traffic

실제적으로 이 영역에서 노드 i의 전송 율이 증가함으로서 연결 i의 처리 율 또한 증가한다. 이 영역에서 연결 i의 효율성은 연결 j의 효율성이 55%에서 75%로 증가한 것과 비교해서 왕복 지연 시간에 대한 지터 처리 율은 100%에서 50%로 감소한다. 반대 연결의 데이터 세그먼트를 전송하기 위해 사용된 동일한 가상 채널을 공유하는 연결의 승인을 가지고 있는 명확한 유선 네트워크의 가상 채널을 통해 전송된 트래픽의 효율성을 이용하여 출력 IP 큐의 승인에 대한 서비스 시간을 아주 적게 설정하였으며, 출력 트래픽의 모든 큐잉은 유선 네트워크 링크 계층에서 발생하게 한다.

그림 12, 13, 14는 유선 네트워크의 링크 계층을

각각의 가상채널에 대해 분리된 큐를 제공하였으며, 각각의 가상 채널의 전송 율은 2.5Mbps, 5.0Mbps, 7.5Mbps와 최대 윈도우 크기는 64 Kbytes로서 7개의 세그먼트로 설정한 TCP 백그라운드 트랙픽에 대한 중간노드의 포워드 패킷 처리 율에 대한 시뮬레이션 결과를 나타내고 있다.그림 12에서 나타낸바와 전송 율이 2.5Mbps로 설정한 상태에서 송신 노드에서 전송된 모든 패킷의 순서에 대해서 중간 노드의 지터의 처리 율이 전체적으로 높게 설정되어 나타남을 알 수 있는데 이는 중간 노드의 연결의 효율성의 저하로 인한 왕복 시간의 증가로 인해 지터의 처리 율이 높게 나타나는데 중간 노드의 혼잡 원인으로 인한 결과이다. 그림 13의 시뮬레이션 결과는 전송 율을 5.0Mbps로 설정한 상태에서의 중간 노드의 지터 처리 율을 나타내고 있다. 결과에서 알 수 있듯이 전송 율의 증가, 즉 연결 효율성이 높아짐으로서 혼잡회피가 이루어짐으로서 왕복 지연의 시간 증가로 인한 지터 처리 율이 낮게 나타나고 있다. 그림 14는 비율 제어된 TCP의 백그라운드 트래픽을 7.5Mbps로 설정하여 시뮬레이션 한 결과를 나타내고 있다.

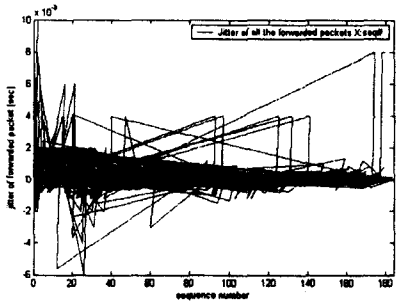


그림 12. TCP 비율 제어 2.5Mbps 백그라운드 트래픽에 대한 중간노드의 포워드패킷 처리율
Fig. 12 Intermediate node forward packet throughput for TCP Rate-controll 2.5Mbps background traffic

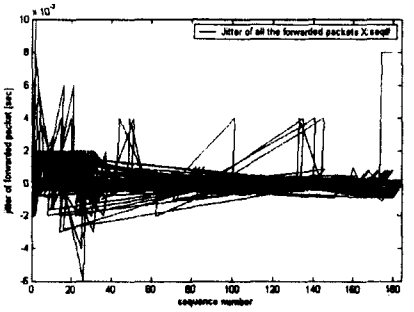


그림 13. TCP 비율 제어 5.0Mbps 백그라운드 트래픽에 대한 중간노드의 포워드패킷 처리율
Fig. 13 Intermediate node forward packet throughput for TCP Rate-controll 7.5Mbps background traffic

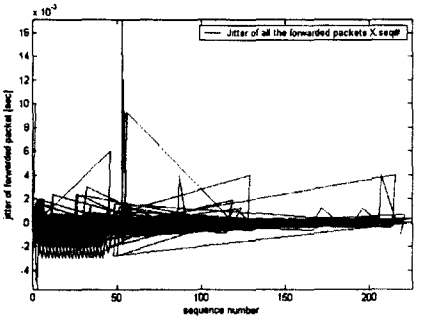


그림 14. TCP 비율 제어 7.5Mbps 백그라운드 트래픽에 대한 중간노드의 포워드패킷 처리율
Fig. 14 Intermediate node forward packet throughput for TCP Rate-controll 7.5Mbps background traffic

결과에서 알 수 있듯이 송신 노드의 모든 패킷 순서에 대해서 지터 처리 율이 2.5Mbps와 5.0Mbps보다 비교하여 전체적으로 낮게 분포되어 있다. 이러한 결과는 노드 사이의 링크 계층에서 수행된 대역폭의 증가 즉, 연결의 효율성의 증가로 인한 혼잡 회피로 인한 왕복 지연 시간이 짧아지며, 또한 전송된 패킷에 대한 지터 처리 율이 낮아지는 것은 제안된 비율 제어 방식의 연결 효율성의 성능 개선이 이루어짐을 알 수 있다. 종단간 노드의 서로 다른 쌍 사이의 트래픽이 다른 가상채널을 통해 전송되어지며, 상호간에 완전하게 분리되어 연결의 각 쌍에 대해 양방향 트래픽의 처리 율을 이용함으로서 중간노드의 왕복지연에 대한 처리 율 결과가 개선되는데, 이는 양방향 트래픽의 대역폭 지연의 값에 대한 TCP 연결 효율성이 지연이 전혀 없는 처리 율에 거의 근접함을 알 수 있다.

V. 결 론

본 논문에서는 유선 매체에서 TCP 성능이 저하되는 원인을 살펴보고 이를 개선하기 위해 제시한 다양한 방식을 검토해 보았다. 이러한 방식은 TCP 성능을 향상시키지만 저마다 확장성이나 경제성과 같은 한계를 가지고 있다. 고정 호스트와 라우터 TCP 스택을 변경해야 하거나 TCP 의미론을 바꿔야 하는 솔루션도 있다. 또한 특정 조건에서는 성능이 향상되지만 그러한 조건이 사라지면 성능이 저하되는 경우도 있다. 따라서 제안된 솔루션 중 전적으로 만족스러운 솔루션은 없다.

유선 네트워크 상에서 양방향 트래픽이 존재하는 곳에서 전송 율 기반하의 혼잡 제어를 통해 처

리 율 성능을 개선하기 위해 비동기 전송 모드 유선 네트워크의 동적인 TCP 연결을 분석하며, 양방향 트래픽을 네트워크 경로를 통해 동일한 종단 노드 쌍 사이의 반대 방향에서 데이터를 전송하는 TCP 연결로부터 생긴 트래픽 패턴을 사용하였다. 또한 TCP의 혼잡 윈도우 업데이트에서 간단한 인터페이스를 통한 함수 호출 형태를 이용하여 TCP 확장을 개발하였으며, Tahoe, Reno, Vegas에 전송율 기반 알고리즘을 다양한 TCP 버전에 적용하여 그 결과 생성되는 프로토콜의 처리 율 성능을 개선시켰다.

TCP 백그라운드 트래픽 레벨에 대한 처리 율 곡선에 대한 시뮬레이션 결과는 유선 네트워크 상의 중간노드 혼잡으로 인한 왕복 지연 시간의 증가가 지터의 처리율을 감소시킴으로서 연결의 효율성이 낮아진다. 송신 노드에서 전송된 모든 패킷의 순서에 대해서 중간 노드의 지터 처리율의 시뮬레이션 결과는 중간 노드의 연결의 효율성의 저하로 인한 왕복 시간의 증가로 인해 지터의 처리 율이 높게 나타나는데 중간 노드의 혼잡 원인으로 인한 결과이다. 이러한 결과는 노드 사이의 링크 계층에서 수행된 대역폭의 증가 즉, 연결의 효율성의 증가로 인한 혼잡 회피로 인한 왕복 지연 시간이 짧아지며, 또한 전송된 패킷에 대한 지터 처리 율이 낮아지는 것은 제안된 비율 제어 방식의 연결 효율성의 성능 개선이 이루어짐을 알 수 있아다.

참고문헌

[1] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Two-way TCP traffic over ATM: Effects and analysis." Tech. Rep. UCCSC-CRL-96-23, Univ. of California, Santa Cruz, 1996.

[2] L. Zhanb, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic", in Proc., ACM SIGCOMM'91, pp. 133-147. Sept. 1991.

[3] R. Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey", Computer Networks and ISDN Systems, Vol. 28, No. 13, pp. 1723-1738, Oct. 1996.

[4] R. Wilder, K. K. Ramakrishnan, and A. Mankin, "Dynamics of congestion control and avoidance of two-way traffic in an OSI testbed", ACM Comput. Commun. Rev., Vol. 21, No. 2, pp. 43-58, Apr. 1991.

[5] Romanow , A. and Floyd, s. "Dynamics of TCP Traffic over ATM Networks", Proc., ACM SIGCOMM'94, pp. 79-88, 1994.

[6] S. Floyd and A. Romanow, "Dynamics of TCP traffic over ATM networks", in Proc., ACM SIGCOMM'94, pp. 79-88, Sept. 1994.

[7] V. Jacobson, "Congestion avoidance and control", in Proc., ACM SIG-COMM'98, pp. 314-329, 1998.[1] A. Adas and A. Mukherjee "On resource management and QoS guarantees for long dependent traffic", In Proc. IEEE INFOCOM '95, pp. 779-787, 1995.

[8] C. Huang, M. Devetsikiotis, I. Lambadaris and A. Kaye, "Modeling and simulation of self-similar variable bit rate compressed video: a unified approach", In Proc. ACM SIGCOMM '95, pp. 114-125, 1995.

[9] Hyogon Kim, "A Non-Feedback Congestion Control Framework for High-Speed Data Networks", Ph.D thesis, University of Pennsylvania, 1995.

[10] Hyogon Kim and David Farber, "The failure of conservative congestion control in large bandwidth-delay product networks", In Proc. INET '95, 1995.

[11] T. V. Laskshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products random loss", IEEE/ACM Trans. Networking, Vol. 5, No. 3, pp. 336-350, 1997.

[12] N. Likhanov and B. Tsybakov, "Analysis of an ATM buffer with self-similar ("fractal") input traffic", In Proc. IEEE INFOCOM '95, pp. 985-992, 1995.

[13] I. Norros, "A storage model with self -similar input. Queueing Systems", Vol. 16, pp. 387-396, 1994.

[14] K. Park, W. Wang, "QoS-sensitive transport of real-time MPEG video using adaptive forward error correction", In Proc. IEEE Multimedia Systems '99, pp. 426-432, 1999.

[15] B. Ryu and A. Elwalid. "The importance of long-range dependence of VBR video tra-

ffic in ATM traffic engineering: myths and realities", In Proc. ACM SIGCOMM '96, pp. 3-14, 1996.

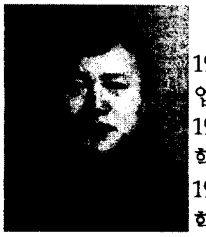
저자 소개

김광준(Gwang-Jun Kim)

1993년 조선대학교 컴퓨터공학과 졸업(공학사)
1995년 조선대학교 대학원 컴퓨터공학과 졸업(공학석사)
2000년 조선대학교 대학원 컴퓨터공학과 졸업(공학박사)
2000년~2001년 Dept. of Electrical & Computer Eng. Univ. of California Irvine Postdoc.
2003년~현재 여수대학교 컴퓨터공학과 전임강사
※관심분야 : ATM망, 인터넷 통신, 컴퓨터 네트워크, 실시간 통신 프로그래밍, 영상 처리 및 통신, 프로그래밍 언어(Visual C++, Java), 이동 통신 등

윤찬호(Chan-ho Yoon)

1997년 호남대학교 컴퓨터공학과 졸업(공학사)
2000년 조선대학교 대학원 컴퓨터공학과 졸업(공학석사)
현재 : 조선대학교 대학원 컴퓨터공학과 박사과정
※관심분야 : ATM망, 데이터 통신, 컴퓨터 네트워크, TCP/IP 혼잡제어, 이동 통신 등



김천석(Chun-Suk Kim)

1980년 광운대학교 전기공학과 졸업(공학사)
1982년 건국대학교 대학원 전기공학과 졸업(공학석사)
1998년 경남대학교 대학원 전기공학과 졸업(공학박사)
1980년~현재 여수대학교 전자통신공학과 교수
※관심분야 : 디지털 신호 처리, 무선 통신, 정보 이론, ATM망, 인터넷 통신, 컴퓨터 네트워크 등