종단간 네트워크 시스템에서 전송율 기반 TCP 혼잡제어

배용근* · 윤찬호* · 김광준**

TCP Congestion Control of Transfer Rate-based in End-to-End Network Systems

Young-geun Bae* · Chan-ho Yoon* · Gwang-jun Kim**

요 약

본 논문에서는 전송 율 기반 흐름과 혼잡 제어를 이용한 종단 간 네트워크 시스템을 통한 양방향 TCP 연결의 성능을 개선한다. TCP 패킷과 승인에 의해 버퍼를 공유하는 것은 ACK 압축이라는 결과를 초래하며, ACK 프레임 압축은 소스가 집단적으로 도착할 때 승인하며 불공정성과 처리 율 감소를 초래한다. 양방향 트래픽 때문에 발생한 처리 율 감소는 중요하다. 적절한 윈도의 크기를 가진 대칭적인 연결인 경우에서 연결의 효율은 2.0Mbps, 5.0Mbps, 7.5Mbps 세 가지 레벨의 백그라운드 트래픽에 대해서 약 20%정도의 성능이 개선된다. 반면에 지터의 처리율은 소스 노드와 목적 노드 사이에 왕복 지연 시간이 적기 때문에 약 50%정도 감소되는 것을 알 수 있다. 또한 처리 율 곡선이 2.5Mbps, 5Mbps, 7,5Mbps 백그라운드 트래픽에 대한 TCP 혼잡 회피를 위해 제안된 전송 율 기반 알고리즘에 의해 개선되어짐을 알 수 있다.

ABSTRACT

In this paper, we improve 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. For example, 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 2.5Mbps, 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 congestion avoidance as a function of aggressiveness threshold for three levels of background traffic 2.5Mbps, 5Mbps and 7.5Mbps.

키워드

Transfer Rate-Based Flow and Congestion Control , ACK compression, Throughput, Background Traffic

Ⅰ. 서 론

지난 몇 년 동안 컴퓨터 네트워크는 폭발적인 성장을 거듭해왔다. 이러한 컴퓨터 네트워크의 집합체인 인터넷과 같은 최선 노력(Best Effort) 네트워크 관리는 제어나 자원 예약의 개념이 존재하지 않으므로 네트워크가 부여되는 로드, 즉 네트워크 내부에 존재하

* 조선대학교 컴퓨터공학부
접수일자 : 2006. 10. 04
** 전남대학교 컴퓨터공학과
심사완료일자 : 2006. 11. 07

는 전체 패킷 수를 제어하지 않기 때문에 네트워크 부하가 많은 최선 노력 네트워크는 혼잡이 발생한다. 네트워크가 혼잡 상태라면 호스트와 라우터 인터페이스의 버퍼가 오버플로우 되면서 패킷 손실이 발생한다. 네트워크종단점은 혼잡 제어를 통해 혼잡에 신속하게 대응함으로써 네트워크의 신뢰성을 보장하고 혼잡으로 인한 시스템 장애를 미연에 예방함으로서 원할한 정보 전송을 추구한다[1][2][3]. 네트워크 혼잡으로 인한 장애의 발생 원인은 다양하다. 속도가 빠른 데이터가 큰 파이프에 도착하여 속도가 느린 작은 파이프로 빠져나가면 네트워크 혼잡이 발생할 수 있으며, 또한 복수의 입력 스트림이 라우터에 도착했으나 그 라우터의 출력 용량이 입력의 합보다 작은 경우에도 발생한다. TCP는 신뢰성 있는 전이중 바이트 스트림으로서 정보 전송 시 혼잡으로 인한 에러가 발생할 때 오류 복구 전략이 정체 제어 방법과 맞물려 이행된다[4][5].

본 논문에서 유선 네트워크상에서 감소된 처리율을 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 처리율이 향상됨을 입증하고자 하며, 혼잡제어 성능 평가 방식이 전송 율 기반하의 네트워크 트래픽의 물리적 모델링으로부터 얻은 시뮬레이션 결과를 통해 개선됨을 나타낸다.

II. 전송율 기반 TCP 연결 혼잡 제어 분석

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. 슬로우 스타트 과정에서 새 데이터에 대해 응답하는 각각의 수신 응답에 대해 송신 노드측에서 최대 세그먼트 크기만큼 정체 윈도우가 증가한다. 따라서 라운드 지연 시간마다 정체 윈도우가 두 배로 증가하게 되는데 정체 윈도우가 초기에 설정된 임계값에 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 임계값은 초기에 설정된 값의 절반으로 설정됨으로서 TCP는 네트워크 정체를 해결한다.

패킷 손실을 나타내는 두 가지 지표로서 발신지에서 재전송 타임아웃과 발신지에 중복된 응답신호의 도착이다. 타임아웃을 통해 패킷 손실이 감지되면 TCP는 네트워크가 정체되었다고 확신하고 슬로우 스타트 알고리즘을 사용하여 손실을 복구하며, 이어 혼잡 회피가 적용된다. 또한 발신 노드에서 중복된 응답신호를 수신하면 단순한 패킷 손실을 의미한다. 수신기는 또 다른 세그먼트가 수신된 경우에만 중복 응답을 생성할 수 있으므로 해당 세그먼트는 네트워크를 떠나 수신기 버퍼에 있으며, 따라서 두 종단점 사이에 데이터의 흐름이 아직 존재하기 때문에 흐름이 갑자기 감소하는 것을 방지하기 위해 TCP는 빠른 재전송 및 복구 알고리즘을 사용한다[6].

양방향 상태에서 TCP 슬로우 스타트 단계 동안에 윈도우 증가가 승인 압축을 이끌어 낼 수 있으며, 슬로우 스타트 단계가 완료되는 곳에서 각각의 연결은 전체적인 윈도우 세그먼트를 버스트로서 전송하면 반대편 연결의 승인 뒤에 온다.

두 연결의 윈도우 크기의 합은 승인 압축이 발생하는 라운드 전송 파이프의 대역폭과 지연의 곱을 초과하여야만 한다. 이러한 제약은 두개의 종단 시스템이 종단 노드의 윈도우가 증가되도록 허용하고 가능한 경로의 대역폭-지연의 곱을 초과하여야 한다는 것을 의미한다. 이것은 비율 제어된 네트워크에서의 경우와 비슷한 경우로서 네트워크내의 큐잉 지연은 적게 된다. 만일 이러한 조건이 만족되지 않는다면 어떠한 지속적인 큐잉도 안정된 상태에서 양쪽 노드의 IP 큐에서 발생하지 않는다는 것을 알 수 있다. 양방향 상태에서 TCP 슬로우 스타트 단계 동안에 윈도우 증가가 승인 압축을 이끌어 낼 수 있으며, 슬로우 스타트 단

계가 완료되는 곳에서 각각의 연결은 전체적인 윈도우 세그먼트를 버스트로서 전송하면 반대편 연결의 승인 뒤에 온다. 두 연결의 윈도우 크기의 합은 승인 압축이 발생하는 라운드 전송 파이프의 대역폭과 지연의 곱을 초과하여야만 한다. 이러한 제약은 두개의 종단 시스템이 종단 노드의 윈도우가 증가되도록 허용하고 가능한 경로의 대역폭-지연의 곱을 초과하여야 한다는 것을 의미한다. 이것은 비율 제어된 네트워크에서의 경우와 비슷한 경우로서 네트워크내의 큐잉 지연은 적게 된다. 만일 이러한 조건이 만족되지 않는다면 어떠한 지속적인 큐잉도 안정된 상태에서 양쪽 노드의 IP 큐에서 발생하지 않는다는 것을 알 수 있다. 그러므로 각 연결의 데이터 세그먼트들은 다른 연결의 승인과는 구별된다. Wi 와 Wj 가 세그먼트에서 두 연결의 윈도우 크기이므로 발생되는 승인 압축에 대한 식은 식(1)과 같다.

$$W_i + W_j > \rho(D_{ij} + D_{ji}) \quad (1)$$

승인 압축의 발생은 초기에 점차적으로 윈도우가 증가하는 TCP 연결의 슬로우 스타트 형태로 추적한다. 슬로우 스타트 알고리즘은 초기 윈도우 크기를 1로 설정하고 승인이 도착할 때 마다 하나씩 증가한다. 이것은 효과적으로 매 라운드 전송시간에 윈도우의 크기가 두 배가 되며, 슬로우 스타트 동안에 매 승인에 대한 수신은 종단 시스템의 출력 큐잉에 두개의 세그먼트 추가시킨다. 출력 큐는 FIFO 순서를 유지하므로 이러한 두개가 세그먼트는 반대편 연결에 대해 승인이 보내지기 전에 전송되어져야 한다. 게다가 두 개의 전송된 세그먼트에 대한 승인이 라운드 전송 지연 후에 도착할 때 그사이에는 데이터 세그먼트가 없으며 응답으로서 전송된 네 개의 세그먼트가 양방향으로 나타난다. 그러는 동안에 반대 연결의 승인이 데이터 세그먼트 뒤에 큐 되고 승인들을 집단적으로 되어 진다. 안정된 상태에서 각 연결의 전체적인 윈도우는 양방향으로 전송되고 반대편도 연결된다.

III. TCP 혼잡 제어의 개선된 연결 효율 알고리즘

슬로우 스타트 처리 과정 동안에 TCP 윈도우 증가가 양방향 트래픽 구조하에 ACK 프레임압축에 대한 결과이다. 윈도우가 최대 크기에 도발한 후에 반대 방향에서 한 쌍의 TCP 흐름을 가진 단순 네트워크 구조에서 TCP 연결의 안정된 상태 처리율이다.

ACK 프레임압축은 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 ACK 프레임뒤에 온다. 노드가 노드의 데이터 세그먼트를 노드에서 발생하는 연결의 번잡 주기로서 전송하는 동안의 시간 간격이다. 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 한쪽 방향 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 $Q_i(t)$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 $\tau_{i,k}$는 노드 i에 도착한 연결 j의 k번째 번잡 기간 동안에 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 집단화된다.

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_1은 식(2)와 같다.

$$t_1 = \tau_{i,k} + \frac{Q_j(\tau_{i,k})}{\rho} + D_{ji} \quad (2)$$

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 프레임열은 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 1과 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)과 같다.

$$\tau_{i,m} = \tau_{i,k} - \frac{(w_j - Q_j(\tau_{i,k}))}{\rho} + D_{ji} \quad (3)$$

t_1을 노드 i가 k번째 번잡 주기의 전송을 완료했을 때 시간은 식(4)와 같다.

$$t_1 = \tau_{i,k} - D_{ij} + \frac{W_i}{\rho} \quad (4)$$

$(\tau_{i,m}, t_1)$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 집단화 된다. $(\tau_{i,m}, t_1)$간격 동안에 집단화된 승인의 수는 $mim(\rho(t_1 - \tau_{j,m}), W_j)$에 의해서 주어진다. 식(3)와 식(4)을 이용하여 $\rho(t_1 - \tau_{j,m})$은 식(5)와 같다.

$$\rho(t_1 - \tau_{j,m}) = (W_i + W_j) - (D_{ij} + D_{ji}) - Q_j(\tau_{i,k}) \quad (5)$$

그러므로 연결 i의 $k+1$번째 번잡 주기의 첫 번째 세그먼트에 의해 알 수 있는 큐 크기는 식(6)과 같다.

$$\begin{aligned} Q_j(\tau_{i,k+1}) &= mim((W_i + W_j) - (D_{ij} + D_{ji})\rho - Q_i(\tau_{i,k}), W_j) \\ &= mim((W_i + W_j) - (L_{ij} + L_{ji}) - Q_i(\tau_{i,k}), W_j) \end{aligned} \quad (6)$$

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.

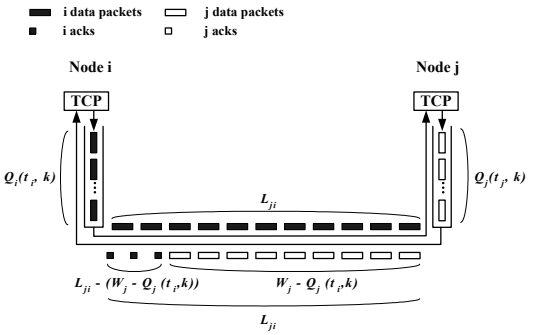

그림 1. 노드 j에 연결 i의 k번째 번잡주기의 첫 번째 세그먼트가 도착 하는 네트워크 동작

Fig. 1 Network behavior on the first segment of the kth busy period of connection i arrives at node j

그러면 ACK 프레임노드 i에 도착할 때 t_1은 식(2)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(7)과 같다.

$$\frac{W_i}{\rho} > t_1 - (\tau_{i,k} - D_{ij}) \quad (7)$$

식(2)으로부터 t_1을 대입하면 식(7)은 식(8)과 같다.

$$W_i > (L_{ij} + L_{ji}) + Q_j(\tau_{i,k}) \quad (8)$$

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(6)과 식(8)을 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(9)와 같은 식을 얻을 수 있다.

$$\begin{aligned}\tau_{i,k+2} &= t_2 + D_{ij} \\ &= \tau_{i,k+1} + \frac{W_j}{\rho} + D_{ji} + D_{ij} \\ &= \tau_{i,k} + \frac{(W_i + W_j)}{\rho} + (D_{ij} + D_{ji})\end{aligned} \tag{9}$$

$Q_i(t_1)=0$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(10)과 같다.

$$\tau_{i,k+1}=t_1+D_{ij}=\tau_{i,k}+\frac{Q_{j(\tau_{i,k})}}{\rho}+(D_{ij}+D_{ji}) \tag{10}$$

이러한 번잡 주기에 대한 첫 번째 응답은 시간 $\tau_{i,k+1}+Q_j(\tau_{i,k+1})/\rho$시간에 노드를 출발하고 식(11)과 같은 시간에 노드 i에 도착한다.

$$t_3=\tau_{i,k+1}+\frac{Q_i(\tau_{i,k+1})}{\rho}+D_{ji} \tag{11}$$

식(9)에 $\tau_{i,k+1}$을 대입하고 식(6)에 $Q_j(\tau_{i,k+1})$을 대입하면 t_3는 식(12)과 같다.

$$t_3=\tau_{i,k}+\frac{W_i+W_j}{\rho}+D_{ji} \tag{12}$$

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(13)과 같다.

$$\begin{aligned}\tau_{i,k+2} &= t_3 + D_{ij} \\ &= \tau_{i,k} + \frac{W_i + W_j}{\rho} + (D_{ij} + D_{ji})\end{aligned} \tag{13}$$

$W_j-(L_{ij}+Lji) \le W_i \le W_j+(L_{ij}+L_{ji})$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 처리율을 달성할 수 있다. 양방향 트래픽에서 처리율은 연결 사이의 상호동작 때문에 지속적으로 감소한다.

Ⅳ. 백그라운드 트래픽에 대한 처리 율 곡선 결과 및 성능 분석

연결 j의 m번째 번잡 주기의 첫 번째 세그먼트가 $t_1=\tau_{j,m}-D_{ji}$시간에 노드 j에 의해서 전송된다. t_1 시간에 노드 j출력 큐는 연결 i의 어떠한 승인도 포

함하지는 않는다. $t_1,\tau_{j,m}$간격 동안에 채널 i로부터 전체적인 세그먼트$(t_1,\tau_{j,m})=D_{ji}\rho=L_{ji}$는 노드 j에 도착된다. 더욱이 연결 i의 L_{ji}는 노드 j에 도착된다. 더욱이 연결 i의 L_{ji}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(14)와 같다.

$$Q_i(\tau_{j,m}) = W_i - (L_{ij} - L_{ji}) \tag{14}$$

양방향 배치 구조에서 전송 율이 다른 보다 일반적인 비대칭적인 경우로 확장하여 적용하고 ρ_i와 ρ_j을 각각 연결 i와 j의 전송 율을 나타내면 ACK 프레임 압축의 생성에 대한 조건은 식(15)와 같다.

$$\frac{W_i}{\rho_i} + \frac{W_j}{\rho_j} > D_{ij} + D_{ji} \tag{15}$$

즉 두 연결의 윈도우의 전송 시간의 합은 발생시키는 ACK 프레임압축에 대한 라운드 전송지연보다 반드시 크다. 이러한 필수적인 조건이 만족하면 비대칭적인 경우에서 연결 처리율에 대한 식은 식(15)와 같다. 비대칭적인 링크 율을 가지고 있는 양방향 트래픽 구조에서 연결 i의 효율성 F_i는 식(16)과 같다.

$$F_i = \begin{cases} 1 & ,w_i/\rho_i) > (w_i/\rho_i) + (D_{ij} + D_{ji}) \\ \dfrac{w(W_i/\rho_i)}{((W_i/\rho_i) + (W_j/\rho_j) + (D_{ij} + D_{ji})} & ,(W_j/\rho_j) - (D_{ij} + D_{ji}) \leq (W_i/\rho_i) \geq (W_j/\rho_i) \\ \dfrac{(W_i/\rho_i)}{(W_j/\rho_j)} & ,otherwise \end{cases} \tag{16}$$

종단 노드에서 최대 큐 크기를 결정할 수 있기 때문에 안정된 상태에서 비대칭적인 링크 율을 가지고 있는 양방향 트래픽 하에 노드 i의 IP 큐의 최대 점유기간 Q_i,max는 식(17)과 같다.

$$Q_i max \leqq \begin{cases} W_i - (D_{ij} + Dji)\rho & ,(W_i/\rho_i) > (W_j/\rho_j) + (D_{ij} + D_{ji}) \\ (W_i/\rho_i) + (W_j/\rho_j) - D_{ij} - D_{ji})\rho & ,(W_j/\rho_j) - (D_{ij} + D_{ji}) \leq (W_j/\rho_j) + (D_{ij} + D_{ji}) \\ W_i & ,otherwise \end{cases} \tag{17}$$

식 (11)은 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.

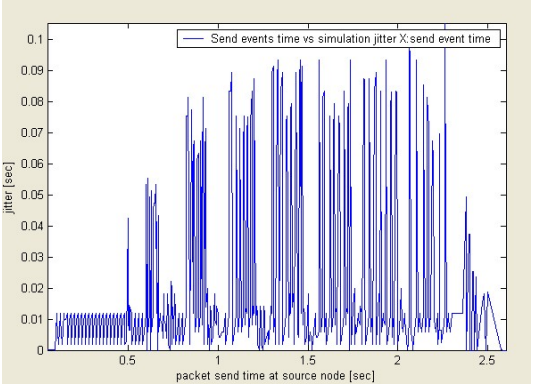

그림 2. TCP 비율 제어 2.5Mbps 백그라운드 트래픽에 대한 지터의 처리율
Fig. 2 Jitter throughput for TCP rate-controll 2.5Mbps background traffic

그림 2, 3, 4는 연결 i에 할당된 율의 함수로서 연결 i와 j의 효율성의 변동으로서 TCP 비율 제어 백그라운드 트래픽의 레벨에 대한 왕복 지연의 지터에 대한 처리율을 나타내고 있다. 그림 2의 결과에서 알 수 있듯이 중간노드의 혼잡으로 인한 송신 노드의 패킷에 대한 지터가 거의 모든 시간에서 지터 처리율이 높아짐을 알 수 있다. 이는 송, 수신 노드 사이의 혼잡으로 인한 왕복 지연 시간의 증가가 지터의 처리율을 감소시킴으로서 연결의 효율성이 낮아진다.

그림 2는 5.0Mbps의 백그라운드 트래픽에 대한 지터 처리율을 나타내는데 5군데의 지터 처리율이 높아지는데, 이는 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 처리율이 적어짐으로서 연결의 효율성이 증가함을 알 수 있다.

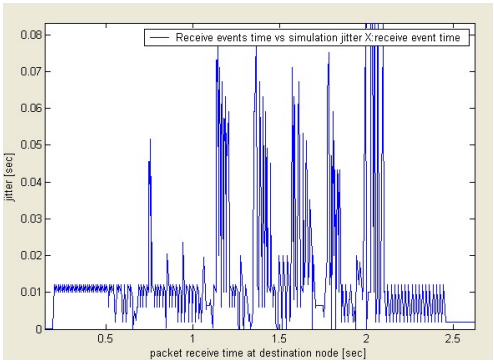

그림 3. TCP 비율 제어 5.0Mbps 백그라운드 트래픽에 대한 지터의 처리율
Fig. 3 Jitter throughput for TCP rate-controll 5.0Mbps background traffic

그림 4는 백그라운드 트래픽을 7.5Mbps로 설정한 지터 처리율에 대한 시뮬레이션 결과를 나타내고 있다. 이는 중간노드에 제안된 TCP 비율 제어를 이용하여 혼잡을 제어함으로서 왕복 지연 시간에 대한 지터 처리율이 단순히 두세 군데에 높게 나타남으로서 2.5Mbps나 5.0Mbps에 발생한 지터 처리율보다 개선됨을 알 수 있다.

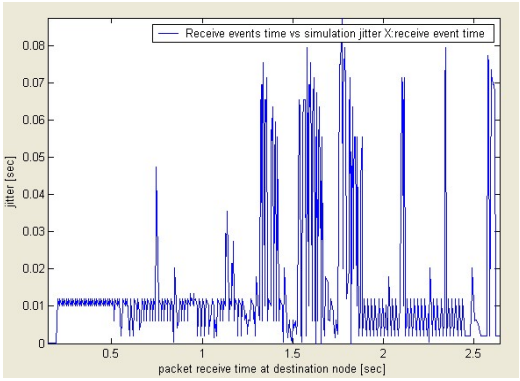

그림 4. TCP 비율 제어 7.5Mbps 백그라운드 트래픽에 대한 지터의 처리율
Fig. 4 Jitter throughput for TCP Rate-controll 7.5Mbps background traffic

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 처리율이 혼잡이 발생했을 경우의 왕복 지연 시간이 증가함을 볼 수 있다. 연결 i의 대역폭이 증가함으로서 연결 j의 처리율은 약 70Mbps가 될 때까지 선형적으로 증가한 반면에 연결 i의 처리율은 무의미한 값으로 유지된다. 이 지점을 넘어서 처리율은 방정식의 두 번째 경우에 의해서 결정되고 연결 j에서 증가하는 처리율은 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. 실제적으로 이 영역에서 노드 i의 전송 율이 증가함으로서 연결 i의 처리율 또한 증가한다. 이 영역에서 연결 i의 효율성은 연결 j의 효율성이 55%에서 75%로 증가한 것과 비교해서 왕복 지연 시간에 대한 지터 처리율은 100%에서 50%로 감소한다.

V. 결 론

TCP에서는 모든 패킷 손실이 정체로 인식되기 때문에 윈도우 크기가 줄어들고 전송 율이 감소하며, 또한 왕복 지연 시간이 증가로 인한 재전송 타임아웃이 증가함으로서 TCP 성능이 떨어진다. 유선 네트워크의 전송 제어 프로토콜은 윈도우 기반 대신 속도 기반의 전송 제어 방식으로 이 문제를 효과적으로 해결한다. 주로 송신노드 속도와 수신노드 속도의 비율을 사용하여 패킷 손실 및 재전송 타임아웃 대비 전송 율을 제어한다.

본 논문에서는 유선 네트워크상에서 양방향 트래픽이 존재하는 곳에서 전송 율 기반하의 혼잡 제어를 통해 처리 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. TCP 백그라운드 트래픽 레벨에 대한 처리 율 곡선에 대한 시뮬레이션 결과는 유선 네트워크상의 중간노드 혼잡으로 인한 왕복 지연 시간의 증가가 지터의 처리율을 감소시킴으로서 연결의 효율성이 낮아진다. TCP 비율 제어를 이용한 백그라운드 트래픽이 2.5Mbps에서 5.0Mbps로 대역폭이 높아짐으로서 중간 노드의 혼잡

이 어느 정도 해소되어 2.5Mbps에서 나타낸 것보다 왕복 지연 시간에 대한 처리율이 적어짐으로서 연결의 효율성이 증가함을 알 수 있다. 또한 백그라운드 트래픽을 7.5Mbps로 설정한 지터 처리율에 대한 시뮬레이션 결과는 혼잡을 제어함으로서 2.5Mbps나 5.0Mbps에 발생한 지터 처리율보다 개선됨을 알 수 있다.

참고 문헌

[1] 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.

[2] P. Barford and M. Crovella, "Generating representative workloads for network and server performance evaluation", In Proc. ACM SIGMETRICS '98, pp.151-160, 1998.

[3] L. Brakmo and L. Peterson, " TCP Vegas: end to end congestion avoidance on a global internet.", IEEE J. Select Areas Commun., Vol. 13, No. 8, pp.1465-1480, 1995.

[4] N. Golmie and D. Su, "Analysis of the Rate-Based Flow Control Mechanism for Available Bit Rate Traffic in ATM Networks", Proceedings of the 3rd International Conference on Optical Communications and Networks, pp233-266, Jan. 1996.

[5] V. Paxson and S. Floyd, "Wide-area traffic the failure of Poisson modeling", In Proc. ACM SIGCOMM '94, pp.257-268, 1994.

[6] G. Pecelli and B. G. Kim, "Dynamic behavior of feedback congestion control schemes", In Proc. IEEE INFOCOM '95, 1995.

저자 소개

배용근(Yong-geun Bae)

1984년 조선대학교 컴퓨터공학과 졸업(공학사)
1987년 조선대학교 전자과졸업(공학석사)
2000년 원광대학교 대학원 전자 공학과 졸업(공학박사)
현재 : 조선대학교 전자정보공과대학 컴퓨터공학부 교수
※ 주관심분야 : 마이크로프로세서, 프로그래밍 언어, 논리회로

윤찬호(Chan-ho Yoon)

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

김광준(Gwang-jun Kim)

1993년 조선대학교 컴퓨터공학과 졸업(공학사)
1995년 조선대학교 대학원 컴퓨터공학과 졸업(공학석사)
2000년 조선대학교 대학원 컴퓨터공학과 졸업(공학박사)
2000년~2001년Dept.of Electrical & Computer Eng. Univ. of California Irvine Postdoc.
2003년~2006년 2월 여수대학교 컴퓨터공학과 조교수
2006년 3월~현재 전남대학교 컴퓨터공학과 조교수
※ 주관심분야 : ATM망, 인터넷 통신, 컴퓨터 네트워크, 실시간 통신 프로그래밍, 영상 처리 및 통신, 프로그래밍 언어(Visual C++, Java), 이동 통신, 의료영상 통신 등