Pacing 적용을 통한 High-Speed TCP 프로토콜의 성능 개선 방안

정회원 최 영 수*, 이 강 원*, 조 유 제*, 한 태 만**

Performance Enhancement of High-Speed TCP Protocols using Pacing

Young-Soo Choi*, Gang-Won Lee*, You-Ze Cho*, Tae-Man Han** *Regular Members*

요 약

지금의 high-speed TCP 프로토콜은 공평성과 TCP friendliness에 심각한 문제가 있으며, 이는 high-speed TCP의 도입에 있어 중요한 문제가 된다. 본 논문에서는 high-speed TCP의 버스티한 특성을 줄여 TCP friendliness와 공평성 향상시키는 방안으로 high-speed TCP에 pacing 적용을 제시하고 성능 분석을 수행하였다. Pacing은 TCP 송신원의 수정만으로 적용 가능하며 TCP 혼잡 제어 방식과는 독립적으로 동작하기 때문에 도입이 쉽다는 장점을 가진다. 시뮬레이션 결과 제안된 방안은 high-speed TCP의 버스티한 특성을 효과적으로 줄이고 성능 저하 없이 기존의 high-speed TCP에 비해 향상된 TCP friendliness와 RTT에 따른 공평성을 제공함을 보였다.

Key Words : Pacing, Congestion control, High-Speed TCP, Fairness, TCP Friendliness

ABSTRACT

Recent studies have pointed out that existing high-speed TCP protocols have a severe unfairness and TCP friendliness problem. As the congestion window achieved by a high-speed TCP connection can be quite large, there is a strong possibility that the sender will transmit a large burst of packets. As such, the current congestion control mechanisms of high-speed TCP can lead to bursty traffic flows in high speed networks, with a negative impact on both TCP friendliness and RTT unfairness. The proposed solution to these problems is to evenly space the data sent into the network over an entire round-trip time. Accordingly, the current paper evaluates this approach with a high bandwidth-delay product network and shows that pacing offers better TCP friendliness and fairness without degrading the bandwidth scalability.

Ⅰ. 서 론

인터넷 기술의 발전에 따라 전세계의 연구 기관은 Abilene이나 ESNet과 같은 망을 통해 1~10Gbps와 같은 초고속 링크로 연결되고 있으며, 광 인터넷과 같은 기술의 발전에 따라 이러한 망의 초고속화는 앞으로도 계속 될 것으로 예상된다. 최근 Atlas 프로젝트나 OptiPuter 프로젝트와 같이 다양한 연구나 응용에서는 기가 혹은 페타 바이트의 매우 큰 데이터를 WAN을 통해 전송하고 있으며, 생명 정보학, 지구 과학, 천문학, 고성능 컴퓨팅 등 다양한 분야에서 TCP를 사용한 초고속 데이터 전송의 필요성이 대두되고 있다[1,2].

TCP는 대부분의 인터넷 트래픽의 전송 프로토콜로

* 경북대학교 전자 전기 컴퓨터 학부(yschoi@eecs.knu.ac.kr, kw0314@eecs.knu.ac.kr, yzcho@ee.knu.ac.kr),
** 한국전자통신원구원(tmhan@etri.re.kr)
논문번호 : KICS2004-08-168, 접수일자 : 1998년 6월 8일
※본 연구는 정보통신부의 정보통신기초기술연구지원사업(03-기초-0012)과 대학 IT연구센터(ITRC) 육성 지원사업으로 수행되었습니다.

표 1. TCP 및 high-speed TCP의 혼잡 제어 알고리즘.

Regular TCP	ACK: $w \leftarrow w + a/w$ LOSS: $w \leftarrow w - b \times w$	$a=1, b=0.5$
HSTCP	ACK: $w \leftarrow w + a(w)/w$ LOSS: $w \leftarrow w - b(w) \times w$	*if* $w \geq Low\_Window(=38)$ $a(w) = 0.1578 \times w^{0.8024} \times b(w)/(2 - b(w))$, $b(w) = 0.052 \ln w + 0.6892$ *else* *regular TCP congestion control*
STCP	ACK: $w \leftarrow w + a$ LOSS: $w \leftarrow w - b \times w$	*if* $w \geq Low\_Window(=16)$ $a=0.01,\ b=0.125$ *else* *regular TCP congestion control*
BI-TCP	ACK: *if*$(target\_win \leq S_{max})$ $w \leftarrow w + (target\_win - w)/w$ *else* $w \leftarrow w + S_{max}/w$ LOSS: $w \leftarrow w - b \times w$	*if* $w \geq Low\_Window(=14)$ $S_{max}=32,\ b=0.125$ *else* *regular TCP congestion control*

사용되고 있으며, 지금까지 비교적 다양한 환경에서 무난한 성능을 보였다. 하지만 최근 망의 대역폭-지연곱 (BDP : Bandwidth Delay Product)이 증가함에 따라 지금의 TCP는 망의 대역폭을 충분히 사용할 수 없으며, 불안정해지기 쉽다는 것이 밝혀졌다[3-7]. [4]에 따르면 지금의 TCP는 10Gbps 수율을 위해 MTU (Maximum Transmission Unit)가 1500 바이트인 경우 83,333 RTT의 시간을 필요로 하며, 이는 RTT가 100ms인 경우 약 1.5시간에 해당한다. 또한 10Gbps의 수율을 위해 TCP는 약 의 패킷 손실률을 필요로 하며, 이는 망의 이론적 BER (Bit Error Rate) 보다 낮기 때문에 지금의 TCP는 위와 같은 큰 수율을 낼 수 없다. 이러한 문제점을 해결하기 위해 다수의 TCP 연결 이용[8], 다수의 TCP 연결을 에뮬레이션하는 혼잡 제어 이용[9,10], TCP 관련 파라미터 튜닝[11-17], 패킷 크기 증가[18]와 같은 기존 TCP를 기반으로 한 다양한 수율 향상 기법이 제시되었다. 하지만 이러한 방식들 TCP friendliness를 보장하지 못하는 문제를 가진다[4,8].

최근 위와 같은 문제를 해결하기 위하여 HSTCP (High Speed TCP)[4], STCP (Scalable TCP)[5], FAST TCP[6], XCP (eXplicit Control Protocol)[3], BI-TCP (Binary Increase TCP)[7]와 같은 초고속 전송 프로토콜에 관한 연구가 진행되고 있다. 이중 XCP는 ATM 망에서 ABR (Available Bit Rate)의 ER (Explicit Rate) 알고리즘과 유사한 방식으로, 혼잡 정도를 명시적으로 통지하기 때문에 매우 뛰어난 효율, 공평성 그리고 안정성을 제공한다. 하지만 XCP는 송신원, 수신원 그리고 라우터의 변경을 필요로 하기 때문에 도입의 문제점이 있으며, TCP friendliness에 관한 연구가 미비하다. 따라서 본 논문에서는 편의를 위해 XCP를 제외한 HSTCP, STCP, BI-TCP를 high-speed TCP 혹은 high-speed 플로우라 하고 이들 프로토콜만을 고려하였다.

지금까지의 high-speed TCP 연구는 주로 대역폭 확장성 (bandwidth scalability)과 TCP friendliness에 초점을 맞추어 진행되었다. 여기서 대역폭 확장성은 링크 대역폭의 증가함에 따라 high-speed TCP가 가용 대역폭을 충분히 활용하여 높은 링크 효율 및 수율을 제공할 수 있어야 함을 의미한다. High-speed TCP는 큰 BDP 환경에서 기존의 TCP 보다 수율이 높으며, 따라서 기존의 공평성 지수와 같은 개념을 high-speed TCP에 바로 적용할 수 없으며, 기존 연구에서도 high-speed TCP는 기존의 TCP의 수율을 크게 저하시키지 않아야 한다는 정도로만 TCP friendliness에 관한 언급이 되어있다[4,7]. 본 논문에서는 [7]을 비롯한 최근 high-speed TCP 관련 논문에서와 같이 링크 효율을 100%에 가깝게 하면서, TCP의 수율 저하가 최소화 되는 것을 TCP friendliness라고 정의하여 성능 평가를 수행하였다. 최근 연구 결과, 현재까지의 high-speed TCP는 TCP friendliness에 문제점이 있으며, RTT에 따른 공평성 즉, 서로 다른 RTT를 가지는 high-speed TCP간의 수율의 차가 매우 크다는 문제점이 밝혀졌다[7].

High-speed TCP는 일반적으로 큰 혼잡 윈도우 (cwnd : Congestion Window)를 가지기 때문에 송신원은 하나의 ack (Acknowledgement)로 수천 혹은 수백 개와 같은 매우 큰 수의 패킷을 버스티하게 전송할 가능성이 매우 높다. 본 논문에서는 high-speed TCP의 버스티한 동작이 TCP friendliness뿐만 아니라

RTT에 따른 공평성 문제에도 큰 영향을 미치는 것을 밝히고, 이를 해결하기 위해 high-speed 플로우에 pacing을 적용하는 방안을 제시하였다. 제시한 방안은 도입이 쉬우며, high-speed TCP의 혼잡 제어 방식과는 독립적으로 적용될 수 있기 때문에 유연한 high-speed TCP 혼잡 제어 설계를 가능하게 한다. 본 논문에서는 pacing 적용 여부에 따른 high-speed TCP의 대역폭 확장성, TCP friendliness, 그리고 RTT에 따른 공평성 성능을 시뮬레이션을 통하여 분석하여, 제시된 방안이 대역폭 확장성의 저하 없이 TCP friendliness와 공평성을 향상시킬 수 있음을 보였다.

본 논문의 구성은 서론에 이어 II 장에서는 기존의 TCP와 TCP의 문제점, 그리고 높은 수율을 위해 지금까지 연구된 기법과 그 문제점을 살펴본다. 또한 대표적 high-speed TCP 프로토콜의 혼잡 제어 알고리즘과 pacing 기법을 고찰한다. III장에서는 지금까지의 high-speed TCP 연구의 문제점 및 이를 해결하기 위한 pacing 적용 방안을 기술한다. IV 장에서는 시뮬레이션 환경을 설명하고 pacing의 시뮬레이터 상의 구현에 대하여 기술한다. V 장에서는 시뮬레이션 결과를 분석한다. 마지막으로 VI 장에서는 결론을 맺는다.

II. 관련 연구 고찰

본 장에서는 기존의 TCP의 혼잡 제어 기법과 고속 WAN 환경에서의 TCP의 문제점 그리고 대표적 high-speed TCP의 혼잡 제어 기법에 관하여 고찰한다. 본 논문에서는 편의를 위해 바이트 단위가 아닌 패킷 단위로 TCP가 동작한다고 가정하며, 세그먼트와 패킷을 혼용하여 사용한다.

TCP의 혼잡 제어는 크게 slow start와 congestion avoidance로 나누어진다. TCP는 cwnd 변수를 관리하여 ack되지 않은 데이타의 크기가 cwnd를 넘지 않는 한에서 패킷을 전송한다. 평형 상태에서의 TCP의 폭주 제어 방식은 AIMD (Additive Increase Multiplicative Decrease)라 불린다. 즉, congestion avoidance 단계에서 TCP는 하나의 ack당 cwnd를 1/cwnd로 증가 시키며, 결과적으로 cwnd는 RTT 당 1씩 증가한다. 패킷이 손실된 경우 TCP는 cwnd를 현재 cwnd의 1/2로 줄인다. TCP의 혼잡 제어 기법은 표 1에 나타낸 것과 같다.

TCP는 이와 같이 cwnd를 점진적으로 천천히 증가시키는 반면, 패킷 손실시 cwnd를 급격히 감소시킨다. 높은 수율을 위해 TCP는 큰 값의 cwnd를 필요로 한다. 하지만 지금의 TCP는 패킷 손실 후 감소시킨 cwnd가 패킷 손실이 일어나기 직전의 cwnd 값으로 되기까지에는 많은 시간을 필요로 한다. 따라서 지금의 TCP는 매우 큰 값의 cwnd를 유지하기 어려우며, 이는 TCP의 응답 함수(response function)으로도 알 수 있다. 응답 함수는 평형 상태에서 평균 패킷 손실 확률과 TCP 수율간의 관계를 나타내며 TCP의 응답 함수는 $T=\frac{1.2}{\sqrt{p}}$ 이다[4,19]. 여기서 T는 RTT당 패킷 전송률 즉 수율을, p는 패킷 손실률을 나타낸다. 예를 들어 1500 바이트 MTU와 100ms의 RTT를 가지는 TCP가 10Gbps 수율을 위해서는 평균 cwnd가 약 83000 세그먼트이어야 하며, 이는 2×10^{-10}의 패킷 손실률을 필요로 한다. 이 패킷 손실률은 망의 이론적 BER 보다 낮은 값으로, TCP의 응답 함수에서 알 수 있듯이 지금의 TCP는 높은 수율을 얻는데 문제점이 있다[4].

1. 기존의 높은 수율을 위한 방안

High-speed TCP 연구와는 별개로 TCP를 사용하여 높은 수율을 얻기 위한 다양한 연구가 진행되고 있다. 이들 연구는 크게 TCP의 관련 파라미터를 조절하는 기법, 패킷 크기를 증가시키는 기법, 그리고 복수의 TCP 연결을 사용하는 기법으로 나누어진다. 먼저 TCP 관련 파라미터 조절을 기반으로 한 대표적 연구로는 GridDT[11], TCP Tuning Daemon[12-14] 등이 있다. 이들 방식은 주로 수신원이나 NIC(Network Interface Card)의 버퍼 크기 조정을 통하여 높은 수율을 얻는 방식이다. 패킷 크기를 증가시키는 대표적인 방법으로는 jumbo frame 기법이 있다[18]. Jumbo frame은 이더넷 프레임 크기를 9000 바이트로 늘려 수율을 향상 시키는 기법이다. 마지막으로 MulTCP, P-TCP(Parallel TCP) 등의 방식과 같이 가상적으로 N개의 TCP 연결을 에뮬레이션 하거나 다수의 TCP 연결을 이용한 수율 향상 연구가 진행되었다[9,10,15-17]. 이 밖에도 bbcp[20], [21], bbftp[22]와 같은 다양한 TCP 수율 향상 도구들이 개발되었다. 이러한 방식들을 사용함으로써 TCP 수율을 증가시킬 수 있으며, 앞에서 살펴본 TCP의 대역폭 확장성 문제를 어느 정도 완화할 수 있다. 하지만 이러한 기법들은 대역폭 확장성이 제한되어 있고 TCP friendliness하지 않다는 문제점을 가진다.

2. High-speed TCP 프로토콜 관련 연구

큰 BDP를 가지는 망에서 TCP의 수율 향상을 위

하여 먼저 HSTCP가 제안되었다[4]. HSTCP에서 사용되는 파라미터 *Low_Window*, *High_Window*, *High_P*는 각각 다음과 같은 의미를 가진다. 먼저 *Low_Window*는 TCP와의 friendliness를 보장하기 위한 파라미터로, HSTCP는 cwnd가 *Low_Window* 보다 작으면 TCP와 동일한 혼잡 제어 알고리즘을 사용하고 크면 HSTCP 응답 함수를 기반으로 한 혼잡 제어를 수행한다. HSTCP는 설계를 간단하게 하기 위해 응답 함수가 그림 1과 같이 로그-로그 스케일에서 직선이 되도록 하였다. *High_Window*와 *High_P*는 HSTCP 응답 함수의 상한 값 명시를 위하여 사용되는 파라미터로, *High_P*는 HSTCP의 cwnd가 평균 *High_Window*일 때의 패킷 손실률을 의미한다. *Low_Window*, *High_Window*, *High_P*의 권고값은 각각 38, 83333, 이며, 이 값을 사용한 응답 함수에 대응한 HSTCP의 혼잡 제어 알고리즘은 표 1과 같다. 표에서와 같이 RTT당 cwnd 증가량 a가 1로 고정된 TCP와 달리 HSTCP의 RTT당 cwnd 증가량 a(cwnd)는 현재 HSTCP의 cwnd 값의 함수이며, cwnd의 값이 커짐에 따라 증가한다. 또한 패킷 손실시 TCP는 cwnd를 b만큼 즉, 1/2로 줄이는 반면 HSTCP는 cwnd가 커짐에 따라 b(cwnd)도 1/2에서 0.1로 감소량이 작아진다

STCP는 TCP, HSTCP와 같은 AIMD 방식이 아닌 MIMD (Multiplicative Increase Multiplicative Decrease) 기반의 혼잡 제어를 수행한다[5]. 표 1에서와 같이 STCP는 cwnd의 값이 16보다 작은 경우 기존의 TCP와 같은 혼잡 제어를 수행하며, 이보다 큰 경우 MIMD 혼잡 제어 방식을 사용한다.

BI-TCP는 RTT에 따른 high-speed TCP 간의 공평성 문제를 해결하기 위하여 제안되었다[7]. BI-TCP의 혼잡 제어의 주요 알고리즘은 표 1과 같이 binary search와 additive increase으로 나누어진다.

BI-TCP는 패킷 손실이 일어나지 않은 cwnd를 최소 cwnd, 패킷 손실이 일어난 cwnd를 최대 cwnd로 두고 그 값들 사이에 있는 평형 상태에 해당하는 cwnd 값을 이진 검색 기법을 이용하여 찾아나간다. 또한 이진 검색 기법으로 결정된 cwnd의 변화량이 RTT당 S_{max} 이상이 되어야 할 경우 BI-TCP는 최대 변화량을 S_{max}로 제한하며 이를 additive increase라 한다. 즉, BI-TCP는 패킷 손실에 관한 히스토리 정보를 바탕으로 평형 상태에 해당하는 cwnd를 찾아가는 것을 기반으로 한 혼잡 제어 기법이다.

이제까지 대부분의 high-speed TCP 연구는 대역폭

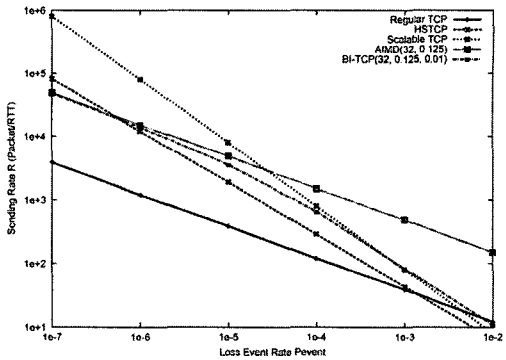


그림 1. TCP 및 High-speed TCP 프로토콜의 응답 함수[7].

확장성과 TCP friendliness를 중심으로 이루어졌다으며, 연구 결과 대부분의 high-speed TCP는 어느 정도 대역폭 확장성 문제를 해결할 수 있음이 밝혀졌다. 하지만 high-speed TCP의 도입에 중요한 TCP friendliness와 RTT에 따른 공평성 문제는 아직도 대부분의 방식에 문제가 있으며 이에 관한 연구가 미흡하다.

3. Pacing 기법 관련 연구

지금의 TCP 송신원은 ack를 수신했을 경우 윈도우가 허용하는 한 패킷을 연속적으로 전송한다. 또한 수신원은 패킷을 수신하는 즉시 해당하는 ack를 전송한다. Pacing은 이러한 TCP의 버스티한 특성을 줄이기 위해 연구된 방안으로, 윈도우 기반 흐름 제어와 전송률 기반 흐름 제어를 혼합한 기법이다. Pacing은 다른 트래픽에 의한 ack compression 문제를 해결하기 위해 처음으로 제안되었다[23]. 이후 pacing은 TCP 연결 시작 시 slow start를 피하기 위해[24,25], 패킷 손실 후[26,27], 혹은 TCP 연결이 idle 상태에 있다가 다시 데이터를 전송할 때[28]와 같이 TCP의 자가 동기(self clocking)를 이용할 수 없을 경우를 중심으로 연구가 진행되었다.

Pacing은 송신원 혹은 수신원 측에서 구현 가능하다. 먼저 송신원 측에서 pacing을 구현할 경우, 송신원은 ack를 수신하더라도 즉시 패킷을 전송하는 대신 패킷간에 적절한 지연을 두어 패킷을 전송한다. 이와 유사하게 수신원 측에서 pacing을 구현할 경우 수신원은 패킷 수신 즉시 ack를 전송하는 대신 ack에 일정한 간격을 두어 전송한다. 그림 2는 기존의 TCP와 송신원에서 pacing을 구현한 TCP의 동작의 차이점을 보여준다. 그림과 같이 pacing은 하나의 RTT동안 cwnd 만큼의 패킷을 전송하는 것은 기존의 TCP와

동일하지만, 윈도우가 허용하는 한 연속적으로 패킷을 전송하는 TCP와 달리 pacing은 패킷간에 cwnd/RTT의 간격을 두어 패킷을 전송한다.

Ⅲ. High-speed TCP의 TCP friendliness및 공평성 향상을 위한 pacing 적용 방안

High-speed TCP는 일반적으로 큰 cwnd를 가진다. 따라서 역방향 경로에서 다른 트래픽에 의해 ack compression이 발생하는 경우 송신원은 연속적 ack를 수신하게 되며, 이에 따라 송신원은 다수의 패킷을 연속적으로 전송할 확률이 매우 높다. 또한 역방향 경로가 혼잡한 경우 ack 패킷의 손실로 인해 수백 혹은 수천 개의 패킷에 해당하는 ack가 도착 할 수 있으며, 이러한 경우 기존의 high-speed TCP는 수백 혹은 수천 개의 패킷을 연속적으로 전송한다. 따라서 소수의 high-speed 플로우는 망 자원을 독점할 확률이 매우 높으며, 다른 플로우들은 거의 동일한 시간에 패킷 손실을 경험한다. 뒤에서 살펴볼 그림 4의 시뮬레이션 결과에서와 같이 라우터 버퍼가 감소할수록 high-speed TCP 간의 공평성은 저하되며, 이는 같은 high-speed TCP 플로우 간에서도 짧은 RTT의 플로우가 망 버퍼를 독점하는 현상 때문이다. 따라서 high-speed TCP의 버스티한 특성을 줄이는 기법이 없는 한 high-speed TCP는 global synchronization과 같은 문제를 야기할 가능성이 있으며, 무엇보다도 TCP friendliness하지 않을 가능성이 매우 높다.

기존의 TCP는 앞에서 살펴본 바와 같이 RTT당 cwnd 증가양 a가 1로 고정되어 있다. 따라서, RTT가 짧은 TCP 플로우는 긴 플로우에 비해 높은 수율을 가진다. High-speed TCP에서도 TCP와 유사하게 RTT에 따른 플로우간 불공평성이 존재한다. 하지만 이러한 불공평성은 다음과 같은 이유로 기존 TCP 상에서의 불공평성보다 매우 심각하며, 이는 high-speed TCP 도입에 있어 중요한 문제이다. HSTCP, STCP의 RTT 당 cwnd 증가양 a는 표 1과 같이 cwnd가 커짐에 따라 증가한다. 따라서 RTT가 짧은 high-speed 플로우는 긴 high-speed 플로우에 비해 보다 빨리 높은 cwnd를 가질 수 있다. 또한 새로운 high-speed TCP가 생겼을 때, 기존의 high-speed TCP의 cwnd는 큰 값을 가지고 이에 따라 패킷 손실시 cwnd 감소양 b(w)는 작은 값을 가진다. 그러므로 두 high-speed 플로우가 같은 값의 수율로 수렴하기까지에 많은 시간을 필요로 한다. 이러한 문제점들로 인해 서로 다른 RTT를 가지는 high-speed 플로우 간의 수율 차이 문제는 기존 TCP에서의 문제보다 매우 심각하다. 예를 들어 V 장 시뮬레이션 결과에서 보이는 바와 같이, 두 개의 HSTCP의 RTT 비가 6인 경우 플로우 간 수율의 비는 약 140이다. 이러한 HSTCP, STCP의 문제점을 해결하기 위하여 BI-TCP는 binary search와 additive increase 단계로 구성된 혼잡 제어 기법을 제안하였다. [7]에서와 같이 BI-TCP가 고속 망에서 사용된 경우 BI-TCP는 대부분의 시간을 additive increase로 동작하기 때문에, cwnd가 높은 상황에서 BI-TCP는 RTT당 cwnd 증가량을 32로 고정한 AIMD 방식과 유사하게 동작한다. 하지만 AIMD는 TCP friendliness에 문제가 있으며 high-speed TCP의 혼잡 제어 기법에 독립적이지 않다는 단점을 가진다. 또한 HSTCP, STCP뿐만 아니라 BI-TCP의 버스티한 특성은 앞에서 살펴본 high-speed TCP와 기존 TCP와의 관계와 유사하게, 짧은 RTT를 가지는 high-speed 플로우 소수가 긴 RTT를 가지는 high-speed 플로우에 비해 망 자원을 지나치게 사용하는 문제를 야기할 가능성이 크다. 따라서 high-speed TCP의 버스티한 특성은 RTT에 따른 공평성 문제를 유발하는 또 하나의 원인이 된다.

이전의 연구는 모두 이러한 high-speed TCP의 버스티한 특성이 문제점을 야기할 수 있다는 것을 인식하였지만 아직까지 버스티한 특성을 줄이기 위한 관련 연구나 적용 시 효용성에 관한 연구가 진행되고 있지 않다. 본 논문에서는 high-speed TCP의 버스티한 특성을 줄이기 위해 high-speed 플로우의 lifetime 동안 pacing을 적용하는 방안을 제안한다. Pacing을 선택한 이유는 먼저 pacing은 high-speed TCP의 혼잡 제어 알고리즘에 독립적이기 때문이다. 따라서 BI-TCP, HSTCP, STCP 모두에 적용 가능하며, 향후에 제안될 high-speed TCP 프로토콜의 혼잡 제어 알고리즘에 독립적일 수 있기 때문에 보다 유연한 혼잡 제어 설계가 가능하다. 또한 본 논문에서는 수신원측에서의 pacing은 ack compression, 역방향 경로의 혼잡과 같은 문제에 취약하기 때문에 송신원측에서의 pacing 적용을 고려하였다. 따라서 pacing은 송신원의 수정만으로 구현 가능하고 도입이 쉽다는 장점을 가진다.

Pacing은 직관적으로 TCP의 버스티한 특성을 줄이기 때문에 높은 수율 및 링크 효율을 제공하고, 망 관점에서도 바람직하다고 인식되고 있다. 하지만 기존의 TCP에 pacing을 적용했을 때의 효용성에 관해서

는 아직까지 논란이 있으며, 특히 pacing을 사용한 TCP의 수율이 pacing을 사용하지 않은 TCP의 수율보다 낮은 문제점을 가지고 있다[32]. 본 논문에서는 기존의 TCP가 아닌 high-speed TCP에만 pacing 적용을 플로우의 lifetime 동안 적용할 것을 제안한다. 뒤에서 살펴볼 시뮬레이션 결과와 같이 high-speed TCP에 pacing을 적용하더라도 high-speed TCP의 대역폭 확장성 성능은 저하되지 않으며 매우 높은 링크 효율을 제공할 수 있다. 또한 pacing 적용은 high-speed TCP간의 공평성 및 TCP friendliness를 향상 시킬 수 있다.

High-speed TCP는 다양한 성능 평가 지표를 통하여 이루어질 수 있으나, 본 논문에서는 특히 대역폭 확장성, RTT에 따른 공평성과 TCP friendliness를 주요 지표로 하여 성능 분석을 수행하였다. 본 논문에서 고려한 high-speed TCP는 HSTCP, STCP, BI-TCP이다. 앞에서 살펴본 바와 같이 XCP, FAST와 같은 주요 high-speed TCP 프로토콜이 존재하나 다음과 같은 이유에서 성능 평가에서는 제외되었다. 먼저 XCP는 앞에서 살펴본 바와 같이 송신원, 수신원뿐만 아니라 망의 변경을 요구하기 때문에 도입에 문제가 있다. FAST는 TCP Vegas와 유사하게 RTT를 기반으로 한 DCA(Delay based Congestion Avoidance) 프로토콜이다. 여기서 DCA는 RTT의 증가 혹은 감소 추세를 기반으로 망의 혼잡 상태를 예측하고 혼잡 제어를 수행하는 프로토콜을 의미한다. 하지만 최근 연구 결과 RTT의 증가 추세와 혼잡에 의한 패킷 손실간의 상관성이 매우 적으며, DCA의 점진적 도입이 어렵다는 것이 밝혀졌다[29]. 따라서 본 논문에서는 FAST 프로토콜을 성능 분석에서 제외하였다.

Ⅳ. Pacing 구현 및 시뮬레이션 환경

본 논문에서는 ns 시뮬레이터를 이용하여 성능 평가를 수행하였다[30]. TCP는 Sack1을 이용하였으며, pacing은 리키 버킷(leaky bucket) 알고리즘을 수정하여 구현하였다. 패킷의 크기는 1000 바이트이며, Pacing은 high-speed 플로우에만 적용하고 기존의 TCP에는 적용하지 않았다. pacing을 위해서 패킷 간의 전송 간격은 cwnd/RTT로 스케줄링 되며, 정확한 RTT 측정을 위해 TCP timestamp 옵션을 사용하였다.

시뮬레이션을 위한 망 구성은 그림 3과 같다. 그림과 같이 병목 링크의 대역폭과 송신원-라우터, 수신원-라우터 간의 지연을 조절하여 다양한 대역폭 및 RTT에서의 성능 평가를 수행하였다. 라우터 R1의 패킷 스케줄링 기법은 FIFO이며, 버퍼 관리 기법은 Drop Tail을 사용하였다. 본 논문에서는 플로우들의 평균 RTT와 대역의 곱을 BDP로 정의하고, 라우터의 버퍼 크기는 특별히 명시되지 않는 한 BDP의 100%로 설정하였다.

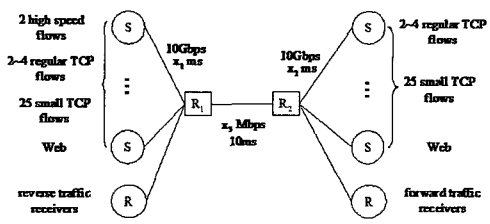


그림 3. 시뮬레이션을 위한 망 구성.

High-speed 플로우는 순방향으로만 발생되며 임의의 플로우 시작 시간을 가진다. 백그라운드 트래픽은 웹 트래픽, 최대 cwnd의 크기가 64로 제한되는 25개의 small TCP 트래픽, 그리고 시뮬레이션에 따라 2개에서 4개의 long lived TCP 트래픽을 사용하였다. small TCP, long-lived TCP는 각각 임의의 RTT와 플로우 시작 시간으로 설정하여 phase effect[31]와 동기화된 피드백 (synchronized feedback)에 의한 효과를 줄였다. 다른 트래픽이 존재하지 않는 경우 링크 대역폭의 약 20~50%의 대역폭을 차지하는 상당한 양의 웹 트래픽을 발생시켰다. 시뮬레이션은 200초간 수행되었으며, 평형 상태에서의 성능을 평가하기 위해 100~200초간의 성능을 분석하였다. 그리고 시뮬레이션 결과는 모두 20번의 실험의 평균값을 나타낸다. HSTCP의 *Low_Window*, *High_Window*, *High_P*는 31, 83000, 0.0000001이며, STCP의 *Low_Window*, *a*, *b*는 16, 0.01, 0.125, BI-TCP의 S_{max}, S_{min}, β는 각각 32, 0.01, 0.125로 설정하였다[4,5,7].

Ⅴ. 시뮬레이션 결과 및 성능 분석

본 장에서는 pacing의 적용 여부에 따른 high-speed TCP의 성능 분석 결과를 시뮬레이션을 통하여 비교 분석한다. 본 논문에서는 대역폭 확장성, RTT에 따른 공평성과 TCP friendliness를 주요 성능 지표로 고려하였다. STCP는 MIMD 기반의 혼잡 제어를 수행하기 때문에 RTT에 따른 공평성 문제가 매우 심각하다. 뒤에서 살펴볼 시뮬레이션 결과에서와

같이 RTT의 비가 6인 경우 두 STCP의 수율 비는 약 300배를 보여준다. 예를 들어 2.5Gbps 링크 사용 시 40ms, 240ms RTT를 가지는 STCP의 수율은 각각 2.2Gbps와 6.84Mbps가 되며, 심각한 공평성 문제를 가진다. 또한 그림 4와 같이 STCP는 매우 aggressive하여 가장 좋지 않은 TCP friendliness를 보여준다. 또한 대역폭 확장성 면에서도 표 2와 같이 HSTCP, BI-TCP는 고속 망에서 매우 높은 효율을 제공하기 때문에, 본 논문에서는 STCP가 high-speed TCP 프로토콜로 부적합하다고 판단, STCP에 pacing을 적용한 성능 분석은 생략하였으며 pacing을 적용하지 않은 STCP만을 성능 비교를 위하여 고려하였다.

1. 대역폭 확장성

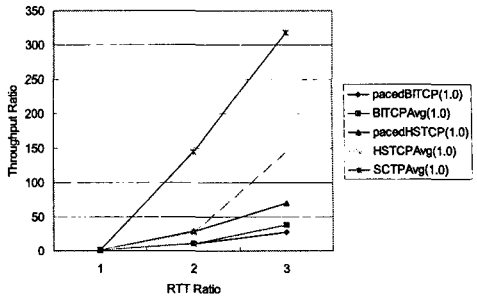
High-speed TCP는 기존의 TCP가 망의 가용 대역폭을 충분히 활용하지 못하는 문제점을 해결하기 위하여 제안되었다. 따라서 high-speed TCP의 대역폭 확장성은 pacing의 적용 여부를 떠나 우선적으로 평가해야 할 성능 지표이다. 성능 분석을 위하여 병목 링크의 대역폭을 변화시켜가며 40ms의 RTT를 가지는 2개의 high-speed 플로우, 순/역방향 각각 4개의 long-lived TCP, 25개의 small TCP, 그리고 웹 트래픽을 이용하였다. 병목 링크의 대역폭 변화에 따른 링크의 효율은 표 2와 같다. 표와 같이 HSTCP, BI-TCP, STCP는 모두 pacing의 적용 여부와 관계없이 100%에 가까운 링크 효율을 보여주며, 따라서 기존의 high-speed TCP는 대역폭 확장성 문제를 어느 정도 해결한 것으로 볼 수 있다. 또한 pacing은 high-speed TCP의 대역폭 확장성에 저하를 초래하지 않는 것을 알 수 있다.

2. RTT에 따른 공평성

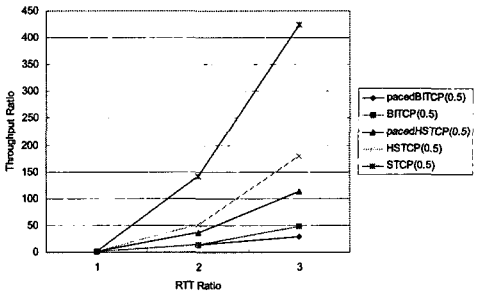
Pacing 적용에 따른 RTT 공평성 분석을 위하여 high-speed 플로우 1의 RTT는 40ms로 하고, high-speed 플로우 2의 RTT를 40, 120, 240ms로 변화시켜가며 두 high-speed 플로우간의 수율 비를 구하였다. 병목 링크의 대역폭은 2.5Gbps이며, 역방향과 순방향 모두 4개의 long-lived TCP, 25개의 small TCP, 그리고 웹 트래픽을 발생시켰다.

표 2. 병목 링크 대역폭 변화에 따른 효율.

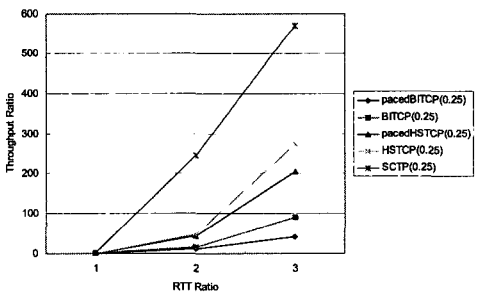
	HSTCP	paced HSTCP	BI-TCP	paced BI-TCP	STCP
20Mbps	93.59	93.98	96.28	96.48	96.01
100Mbps	92.60	93.39	98.54	98.36	98.98
500Mbps	97.93	97.77	99.89	99.76	99.81
2.5Gbps	99.98	99.94	99.97	99.98	99.99



(a) 버퍼 크기 = 100% BDP



(b) 버퍼 크기 = 50% BDP



(c) 버퍼 크기 = 25% BDP

그림 4. RTT 변화에 따른 high-speed TCP 플로우간의 수율 비.

그림 4-(a)는 버퍼 크기가 BDP의 100%일 때의 pacing 적용 여부에 따른 각 high-speed TCP 방식들의 수율 비를 나타낸다. 그림과 같이 RTT의 비가 6인 경우 HSTCP 플로우간의 수율 비는 약 140이 되며, pacing을 적용한 경우 약 70으로 수율비가 줄어듦을 알 수 있다. High-speed TCP와 같이 RTT당 cwnd의 증가량이 현재 cwnd에 비례하는 방식은 기

존의 TCP보다 매우 심각한 RTT 불공평성을 초래한다. 즉, RTT가 짧은 플로우는 RTT가 긴 플로우보다 빨리 cwnd를 높은 값으로 가지게 되며, 앞에서 살펴본 바와 같이 cwnd가 매우 큰 플로우는 연속적인 패킷 전송을 할 확률이 매우 높게 된다. 따라서 짧은 RTT를 가진 high-speed TCP 플로우는 병목 라우터의 버퍼를 독점할 확률이 높으며, RTT에 따른 공평성에 심각한 성능 저하를 초래한다. Pacing은 짧은 RTT를 가지는 플로우가 병목 라우터의 버퍼를 독점하는 것을 방지할 수 있기 때문에 그림 4와 같이 pacing의 사용은 high-speed TCP의 혼잡 제어 알고리즘과는 독립적으로 RTT에 따른 공평성을 향상 시킬 수 있다. 그림 4-(b)와 (c)는 각각 병목 라우터의 버퍼 크기를 BDP의 25, 50%로 변화시켜 가며 수율 비를 나타낸 그림이다. 그림과 같이 버퍼 크기가 감소할수록 모든 방식은 공평성이 저하됨을 알 수 있다. 이것은 버퍼 크기가 작아질 수록 RTT가 짧은 high-speed TCP가 버퍼를 독점할 확률이 높고, 이에 따라 다른 플로우들의 수율 저하가 발생하기 때문에 발생한다. 하지만 그림과 같이 pacing은 다양한 버퍼 크기에서도 수율의 비를 약 1/2로 낮출 수 있으며, RTT에 따른 공평성 문제를 high-speed TCP의 혼잡 제어 방식과는 독립적으로 다양한 버퍼 크기에서도 향상 시킬 수 있다.

3. TCP friendliness

High-speed TCP는 대역폭 확장성을 보장함과 동시에 기존 플로우의 심각한 수율 저하를 야기하지 않아야 한다. TCP friendliness 성능 분석을 위하여 2개의 long-lived TCP, 25개의 small TCP, 그리고 웹 트래픽, 2개의 high-speed 플로우를 사용하였고, 병목 링크 대역폭을 변화 시켜가며 각 플로우 유형의 수율을 구하였다. 그림 5는 각 플로우 유형에 따른 병목 링크의 대역폭 사용량을 나타내었다.

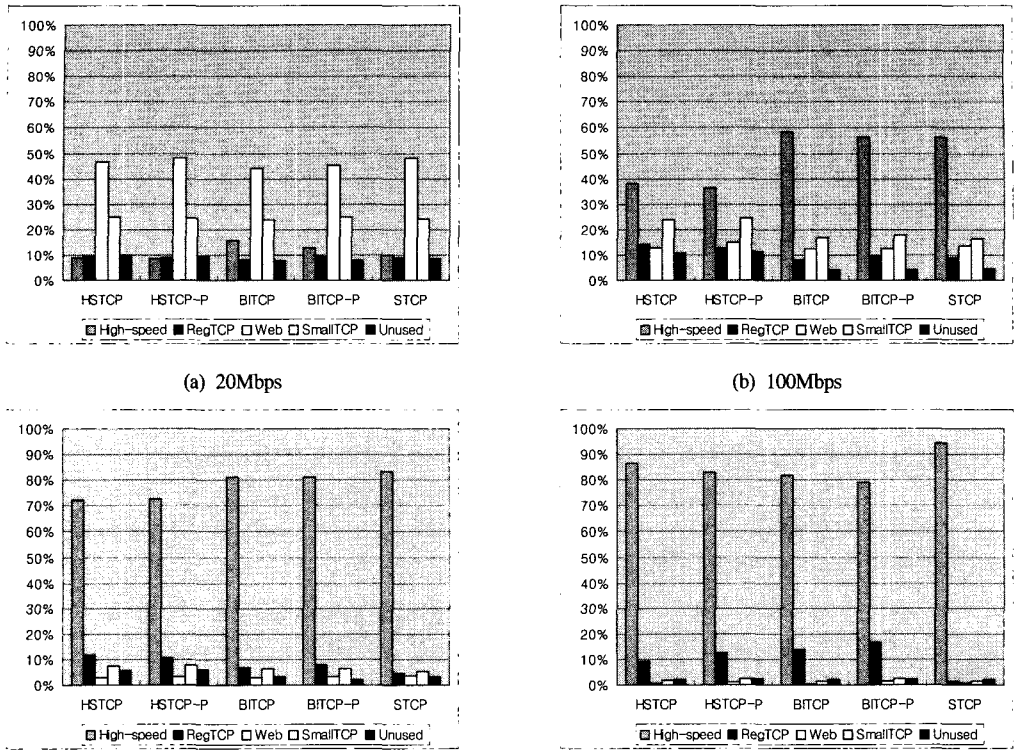
먼저 병목 링크 대역폭이 20Mbps인 경우 BI-TCP를 제외한 각 방식은 비슷한 TCP friendliness를 보여준다. 반면 BI-TCP는 가장 좋지 않은 TCP friendliness를 보여주는데, 이는 BI-TCP의 *Low_Window*의 값이 14로 타 방식에 비해 비교적 낮고, 또한 cwnd가 작은 경우 대부분의 혼잡 제어가 binary search에 의해 이루어지기 때문이다. 100Mbps의 경우 pacing을 적용한 HSTCP, HSTCP, pacing을 적용한 BI-TCP, BI-TCP, STCP의 순서로 향상된 TCP friendliness를 보여준다. BI-TCP의 TCP friendliness 저하 이유는 20Mbps 환경에서 기술한 바와 같다. STCP는 BI-TCP와 유사하게 Low_Window의 값이 16으로 HSTCP에 비해 낮은 편이며, MIND 기반의 혼잡 제어 방식을 사용하기 때문에 이러한 결과를 가진다. HSTCP와 pacing을 적용한 HSTCP를 비교하였을 때, high-speed 플로우의 수율은 pacing을 적용한 방식이 약간 낮다. 하지만 망에서 사용되지 않은 대역폭의 크기는 서로 비슷하기 때문에 HSTCP는 기존의 TCP의 수율을 낮추며 자신의 수율을 높이는 것을 알 수 있다. 따라서 pacing을 적용한 HSTCP가 기존의 HSTCP보다 향상된 TCP friendliness를 가지는 것을 알 수 있다.

500Mbps, 2.5Gbps 환경에서는 웹, small TCP, long-lived TCP가 사용하는 대역폭은 망의 대역폭에 비해 현격히 적다. 이것은 앞에서도 지적한 바와 같이 TCP의 근본적 문제점이며, 기존의 TCP는 망 자원을 효율적으로 사용하지 못한다는 것을 보여준다. 500Mbps, 2.5Gbps 환경에서 pacing을 적용한 HSTCP, HSTCP, pacing을 적용한 BI-TCP, BI-TCP, STCP의 순서대로 TCP friendliness하다는 것을 알 수 있다. 2.5Gbps 환경에서 링크의 사용 효율은 앞에서 살펴본 바와 같이 모든 방식이 100%에 가까운 매우 높은 효율을 가진다. 또한 STCP는 RTT에 따른 공평성과 더불어 TCP friendliness에도 심각한 문제를 가지는 것을 알 수 있다. HSTCP는 앞의 환경과는 달리 BI-TCP보다 낮은 TCP friendliness를 제공하며, BI-TCP가 가장 뛰어난 TCP friendliness를 보여주는 것을 알 수 있다. 또한 Pacing을 적용한 HSTCP는 BI-TCP와 비슷한 성능의 TCP friendliness를 보여주며, pacing을 적용한 BI-TCP가 가장 뛰어난 성능을 나타낸다.

실험 결과에서와 같이 pacing은 기존 방식의 대역폭 확장성의 저하 없이 다양한 링크 대역폭 환경에서 high-speed TCP 혼잡 제어 알고리즘과는 독립적으로 향상된 TCP friendliness를 제공할 수 있다.

Ⅳ. 결 론

큰 BDP를 가지는 망에서 대역폭의 효율적 사용을 위해 최근 high-speed TCP에 관한 연구가 진행되고 있다. 하지만 기존의 연구는 주로 대역폭 확장성에 초점을 맞추어 연구가 수행되었으며, RTT에 따른 공평성, TCP friendliness에 관한 연구가 미흡하다. 본 논문에서는 high-speed TCP의 매우 버스티한 특성이 RTT에 따른 공평성, TCP friendliness 저하에 심각한 영향을 줄 수 있음을 밝히고 버스티한 특성을 줄이기



(a) 20Mbps (b) 100Mbps

(c) 500Mbps (d) 2.5Gbps

그림 5. 병목 링크 대역폭 변화에 따른 TCP friendliness.

위한 방안으로 pacing의 적용을 제안하였다. Pacing은 high-speed TCP의 혼잡 제어 기법과는 독립적으로 적용 가능하고 송신원 측에서만 수정할 수 있으므로 도입에 있어서 매우 뛰어난 장점을 가진다. 또한 시뮬레이션을 통하여 pacing은 다양한 망 환경에서 high-speed TCP의 대역폭 확장성 성능을 저하시키지 않으면서 RTT 공평성 및 TCP friendliness 성능을 향상 시킬 수 있음을 보였다.

참 고 문 헌

[1] The ATLAS Experiment : http://atlasexperiment.org/

[2] OptIPuter : http://www.calit2.net/news/2002/9-25-optiputer.html

[3] D. Katabi, Mark Handley, and Charles Rohrs, "Internet Congestion Control for High Bandwidth-Delay Product Networks," In *Proceedings of the ACM Sigcomm*, Aug. 2002.

[4] S. Floyd, "HighSpeed TCP for Large Congestion Windows," *RFC3649*, Dec. 2003.

[5] T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", *ACM Computer Communication Review*, vol.33, no. 2, pp. 83-91, Apr. 2003.

[6] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," In *Proceedings of the IEEE Infocom*, March, 2004.

[7] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast, Long Distance Networks," In *Proceedings of IEEE Infocom*, March, 2004.

[8] H. Bullot, R. Cottrell, and R. Hughes-Jones, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks," In *Proceedings of Second International Workshop on Protocols for Fast Long-Distance Networks*, 2004.

[9] P. Gevros, F. Risso, and P. Kirstein, "Analysis

of a Method for Differential TCP Service," In *Proceedings of the IEEE Globecom*, vol. 3, pp. 1699-1708, Dec. 1999.

[10] J. Crowcroft and P. Oechslin, "Differentiated End-to-end Services using a Weighted Proportional Fair Share TCP," *ACM Computer Communication Review*, vol. 28, no. 3, pp.53-69, 1998.

[11] S. Ravot, "TCP transfers over high latency/bandwidth networks & Grid DT," In *Proceedings of First International Workshop on Protocols for Fast Long-Distance Networks*, Feb. 2003.

[12] T. Dunigan, M. Mathis, and B. Tierney, "A TCP Tuning Daemon," In *Proceedings of SuperComputing: High Performance Networking and Computing*, 2002.

[15] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S.Tuecke., "GridFTP Protocol Specification," *Global Grid Forum Recommendation GFD.20*, 2003.

[17] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, 2000.

[18] Jumbo Frame Information : http://sd.wareo nearth .com/~phil/net/jumbo/

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," In *Proceedings of ACM Sigcomm*, pp. 303-314, 1998.

[20] bbcp : http://www.slac.stanford.edu/~abh/

[21] B. Tierney, J. Lee, T. Chen, H. Herzog, G. Hoo, G. Jin, and B. Johnston, "Distributed parallel data storage systems: A scalable approach to high speed image servers," in *Proceedings of the Second ACM International Conference on Multimedia*, 1994.

[22] bbftp : http://doc.in2p3.fr/bbftp/.

[23] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two Way Traffic," In *Proceedings of the ACM Sigcomm*, pp. 133-147, Sept. 1991.

[24] M. Aron and P. Druschel, "TCP: Improving startup dynamics by adaptive timers and congestion control," *Technical Report TR98-318*, Rice University, 1998.

[25] V. N. Padmanabhan and R. H. Katz, "TCP fast start: A technique for speeding up web transfers," In *Proceedings of the IEEE Globecom*, 1998.

[26] J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," *ACM Computer Communication Review*, vol. 26, no. 4, pp. 270-280, Oct. 1996.

[27] M. Mathis, J. Semke, J. Madhavi, and K. Lahey, "The Rate-Halving Algorithm for TCP Congestion Control," Work in Progress, Internet Draft, June, 1999.

[28] V. Visweswaraiah and J. Heidemann, "Improving Restart of Idle TCP Connections," *Technical Report TR97-661*, University of Southern California, 1997.

[29] J. Martin, A. Nilsson, and I. Rhee, ""Delay Based Congestion Avoidance for TCP," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 356-369, 2003.

[30] The Network Simulator ns2

[31] S. Floyd and E. Kohler, "Internet Research Needs Better Models," *ACM Computer Communication Review*, vol. 33, no.1, pp. 29-34, Jan. 2003.

[32] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," in *Proceedings of IEEE Infocom*, 2000.

최 영 수 (Young-Soo Choi) 정회원

1998년 2월 : 경북대학교 전자공학과 졸업
2000년 2월 : 경북대학교 전자공학과 석사
2000년 3월~현재 : 경북대학교 전자공학과 박사과정

<관심분야> 트래픽 제어, 차세대 인터넷 프로토콜 및 이동 통신망, 광 인터넷

이 강 원 (Gang-Woo Lee) 정회원

2002년 2월 : 경북대학교 전자공학과 졸업
2004년 2월 : 경북대학교 전자공학과 석사
2004년 3월~현재 : 경북대학교 전자공학과 박사과정

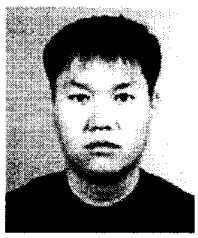
<관심분야> 네트워크 이동성, 트래픽 제어, BcN

조 유 제 (You-Ze Cho) 정회원

1982년 2월 : 서울대학교 전자공학과 졸업 (공학사)
1983년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사)
1988년 8월 : 한국과학기술원 전기 및 전자공학과 졸업 (공학박사)
1989년 3월~현재 : 경북대학교 공과대학 전자전기컴퓨터학부 교수
2002년 2월~2003년 1월 미국 국립표준연구소(NIST), 객원연구원
1992년 8월~1994년 1월 Univ. of Toronto, 객원교수

<관심분야> 트래픽 제어, 차세대 이동통신망, 차세대 인터넷 프로토콜, 센서 네트워크

한 태 만 (Tae-Man Han) 정회원



1985년 2월 : 경북대학교 전자공학과
2003년 2월 충남대학교 컴퓨터과학과(석사)
2003년 현재 충남대학교 컴퓨터과학과 박사과정 재학 중
1986 ~ 1987년 삼성전자 통신연구소
1987 ~ 1995년 LG정보통신 중앙연구소
1995 ~ 현재 한국전자통신연구원 네트워크연구소 스트리밍기술팀 선임연구원

<관심분야> 인터넷 QoS, 멀티캐스트 기술, 트랜스코딩 기술, 스트리밍 프로토콜 기술, HDTV 콘텐츠 제작 기술