논문 2006-43TC-10-2

# 광대역 네트워크에서의 혼잡 제어 성능 개선을 위한 ACA-TCP 설계 및 성능 분석

## ( Design and Performance Evaluation of ACA-TCP to Improve Performance of Congestion Control in Broadband Networks )

나 상 완*, 박 태 준**, 이 재 용***, 김 병 철***

( Sang Wan Na, Tae Joon Park, Jae Yong Lee, and Byung Chul Kim )

요 약

현재 초고속 인터넷 사용자가 급증하고 있고, 광대역 네트워크 인프라의 구축이 늘어나고 있다. 하지만, 지금 사용하고 있는 TCP 혼잡 제어 알고리즘은 협대역 네트워크 환경에 적합하며, 광대역 네트워크의 트래픽 전송에 있어서 효율성이 낮은 상태이다. 이런 문제점을 개선하기 위해 광대역 네트워크에 적합하도록 개선된 혼잡 제어 알고리즘을 적용한 TCP가 요구되고 있다. 본 논문에서는 광대역 네트워크를 충분히 활용할 수 있도록 개선된 TCP 혼잡 제어 알고리즘을 제안한다. 제안된 알고리즘은 수신측에서 송신측에 보내는 ACK 정보 및 RTT 변화와 특성을 이용하여 가용한 대역폭을 예측함으로써 가용한 혼잡 윈도우 크기를 조정하고 대량의 패킷 손실을 최소화 한다. 또한, 혼잡 회피 단계에서의 기존 TCP 알고리즘을 개선함으로써 가용한 대역폭을 빠르게 활용할 수 있도록 한다. 본 논문에서는 제안된 알고리즘 성능을 평가하기 위하여 ns-2를 이용하여 시뮬레이션을 수행하였으며, 시뮬레이션 결과 광대역 네트워크 환경에서 제안된 알고리즘이 기존 HSTCP[2]보다 전송률이 향상되어 가용 대역폭 활용률을 높였을 뿐만 아니라 공정성과 RTT 공정성도 개선하였음을 보였다.

Abstract

Recently, the high-speed Internet users increase rapidly and broadband networks have been widely deployed. However, the current TCP congestion control algorithm was designed for relatively narrowband network environments, and thus its performance is inefficient for traffic transport in broadband networks. To remedy this problem, the TCP having an enhanced congestion control algorithm is required for broadband networks. In this paper, we propose an improved TCP congestion control that can sufficiently utilize the large available bandwidth in broadband networks. The proposed algorithm predicts the available bandwidth by using ACK information and RTT variation, and prevents large packet losses by adjusting congestion window size appropriately. Also, it can rapidly utilize the large available bandwidth by enhancing the legacy TCP algorithm in congestion avoidance phase. In order to evaluate the performance of the proposed algorithm, we use the ns-2 simulator. The simulation results show that the proposed algorithm improves not only the utilization of the available bandwidth but also RTT fairness and the fairness between contending TCP flows better than the HSTCP in high bandwidth delay product network environment.

**Keywords**: TCP congestion control, high bandwidth delay product, bandwidth measurement

* 정회원, (주)KT
(KT Corp.)
** 정회원, 한국전자통신연구원
(ETRI)
*** 종신회원, 충남대학교 전기정보통신공학부
(Division of Electrical and Computer Engineering, Chungnam National University)
※ 본 논문은 한국과학재단의 특정기초연구 (R01-2006-000-10154-0(2006)) 지원으로 수행되었음
접수일자: 2006년7월3일, 수정완료일: 2006년10월2일

## Ⅰ. 서 론

일반적인 전송 프로토콜로써 TCP는 높은 수준의 신뢰성과 전송률을 유지할 수 있도록 설계되어 있어 다양한 네트워크 환경에서 가장 널리 사용되고 있다. 그러나 최근 초고속 인터넷 사용자 및 트래픽의 급증과 네트워크가 협대역에서 광대역으로 변해감에 따라 협대역

네트워크에 적합하도록 설계되었던 기존 TCP의 혼잡 제어 메커니즘을 그대로 사용하였을 경우 제공된 대역을 모두 사용하기 위해서는 과도하게 긴 시간이 필요하며, 패킷 손실이 발생하였을 경우 느린 혼잡 윈도우 증가 속도로 인한 대역폭 사용 효율성 등에 있어서 문제점이 발생하고 있다.

따라서 광대역 네트워크에 적합한 TCP 혼잡 제어(congestion control) 메커니즘의 필요성이 대두되어 왔으며 현재 많은 개선된 메커니즘이 제안되고 있지만, 가용 대역폭의 효율적인 사용, 공정성(fairness)과 RTT에 따른 공정성 등에 있어서 문제점을 가지고 있다[1].

본 논문에서는 광대역 네트워크를 위해 설계된 대표적인 TCP 알고리즘인 HSTCP[2] 보다 더 효율적으로 제공된 대역폭을 사용할 뿐만 아니라 공정성 측면에서도 개선된 알고리즘을 제안한다.

본 논문은 저속 출발(SS : Slow Start) 구간에서 ACK (Acknowledgement) 수신시 마다 네트워크의 가용 대역폭을 측정하여 혼잡 윈도우(cwnd : Congestion Window)의 SS 임계값(ssthreshold)이 동적으로 설정되게 하였고, cwnd가 가용 대역폭을 최대로 활용하고 있다고 판단되는 SS 임계값에 도달한 시점에는 지수 함수 형태의 증가에서 선형 함수 형태로 변화하여 증가하도록 하였다. 세 개의 중복 ACK가 발생하여 cwnd 값이 반으로 감소한 후 다시 증가를 시작하는 혼잡 회피(CA : Congestion Avoidance) 구간에서는 개선된 SS 메커니즘을 수행되도록 하였다. 그 결과 제안된 알고리즘은 cwnd가 급격하게 증가하는 것을 방지하고 다량의 패킷이 손실되는 문제점을 개선하였으며, 기존 TCP 알고리즘이 CA 구간에서 가용 대역폭을 최대로 활용하기 위해 오랜 시간을 소비해야 하는 문제점을 개선하였다. 또한 HSTCP와의 비교를 통한 공정성(fairness) 평가에 있어서 제안된 알고리즘이 더 공정하게 대역폭을 사용하는 것을 보였으며, TCP Reno와 HSTCP와의 RTT 공정성 비교, 평가에서도 ACA-TCP가 RTT 차이에 의한 전송 성능의 차이가 가장 적게 나타나 RTT 공정성에서도 개선된 결과를 나타냈다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 기존에 제안되었던 TCP에 대해서 알아보고, 제 Ⅲ장에서는 TCP 성능을 높이기 위해서 본 논문에서 제안한 TCP 혼잡 제어 알고리즘에 대해서 알아본다. 제 Ⅳ장에서는 시뮬레이션을 통해 본 논문에서 제안한 프로토콜과 광대역 네트워크 사용을 위해 제안되었던 HSTCP와의 성능을 비교 분석하고, 제 Ⅴ장에서는 결론을 맺는다.

## Ⅱ. 관련연구

본 장에서는 협대역 네트워크에 적합하도록 제안되었던 기존 TCP에 대하여 알아보고, 기존 TCP의 혼잡 제어를 광대역 네트워크에 사용하였을 경우 가용 대역폭을 제대로 활용하지 못하는 문제점을 해결하기 위해 제안되었던 새로운 TCP 혼잡 제어 알고리즘에 대해서 알아본다.

기존 TCP 알고리즘의 대표적인 특성인 혼잡 제어는 광대역 네트워크에서 효율적인 트래픽 제어를 위하여 가장 중요하게 이루어져야 할 부분이다. 기존 TCP의 혼잡 제어는 동적 윈도우 설정을 위한 혼잡 윈도우 기반으로 혼잡 제어를 수행하고 있으며, 하나의 ACK가 수신될 때마다 한 개의 MSS(Maximum Segment Size) 만큼 증가하는 SS 구간과, 한 개의 ACK 이 수신될 때마다 1/cwnd 만큼 증가하는 CA 구간으로 나눌 수 있다. TCP는 연결 설정 후에 SS 구간으로 동작하며 cwnd가 SS 임계값과 같거나 커지게 되면 CA 구간으로 동작하게 된다. TCP 송신자는 cwnd를 선형적으로 증가시키다가 ACK가 정상적으로 수신되지 않았을 경우 망의 혼잡에 의해서 패킷이 손실되었다고 판단하고 cwnd의 값을 반으로 줄이는 AIMD(Additive Increase Multiplicative Decrease) 메커니즘을 사용한다.

### 1. 협대역 네트워크에 적합한 TCP

협대역 네트워크에서 사용되는 기존의 대표적인 TCP로는 수신되는 ACK 패킷에 의해서 cwnd를 조절하는 TCP Reno와 ACK 패킷의 RTT를 이용하는 TCP Vegas가 있다.

#### 가. TCP Reno

TCP Reno는 TCP Tahoe에 빠른 복구(fast recovery) 알고리즘을 추가로 도입한 것이다[3]. 주요 기능으로는 SS, CA, 빠른 재전송(fast retransmit) 및 빠른 복구(fast recovery)가 있다. 빠른 재전송이란 타임아웃(time out)이 발생하기전이라도 중복 ACK가 세 개 발생하면 해당 패킷을 재전송하는 알고리즘이고, 빠른 복구는 CA 구간에서 패킷 손실이 발생했을 때 전송 효율을 높이기 위하여 혼잡 윈도우를 현재 윈도우의 반으로 줄이고, 선형 함수 증가 형태로 윈도우를 증가시키며, 손실된 패킷을 재전송하는 메커니즘이다. 지금까지 TCP Reno는 협대역 네트워크에서 안정되고 전송 효율이 높은 TCP 성능을 보였다. 하지만 TCP Reno는 광대

역 네트워크에서 패킷 손실로 인하여 cwnd 값이 반으로 감소한 후에 가용한 최대 cwnd 값에 도달하기 위해서 많은 시간이 필요하게 되는 문제점을 가지게 된다. 또한, CA구간에서의 cwnd 값이 네트워크의 비트 에러에 의해 발생하는 패킷 손실로 인하여 광대역 네트워크 환경에서 가용한 최대 cwnd 값에 이르기도 전에 감소하는 문제점을 가지게 된다.

나. TCP Vegas[4]

TCP Vegas의 기본 알고리즘은 TCP Reno의 SS, 재전송, CA 등의 알고리즘을 수정한 것으로, 전송되는 패킷수를 최적 상태로 유지하기 위해 RTT에 의한 지연(delay) 정보를 이용하여 라우터에 저장되는 패킷 수를 특정 범위 이내로 제한하고, 윈도우 크기가 과도하게 커지는 현상을 제어하는 방법이다. 즉, 라우터에 저장되는 데이터양이 증가하게 되면 네트워크의 혼잡 원인이 되므로 전송로의 실제 패킷 수를 추정한 값과 전송률 기대치를 계산하고, 그 계산으로 구해진 전송률 차이 값을 이용하여 cwnd의 크기를 조절하고 변화를 주어 라우터에 저장되는 패킷 수가 일정하게 유지되도록 한다. 이 알고리즘은 네트워크의 가용 대역폭을 잘 추정하고, cwnd가 TCP Reno에 비해 부드럽게 변화하는 장점을 가지고 있지만, TCP Reno와 동시에 사용할 경우 cwnd를 공격적으로 증가시키는 TCP Reno에 비해 전송률이 떨어지는 단점을 지닌다.

### 2. 광대역 네트워크에 적합한 TCP

네트워크 기술의 발달로 광케이블 등을 이용한 광대역 네트워크의 보급에 따라 기존 협대역에 적합한 TCP 알고리즘의 문제점을 해결하고 광대역 네트워크에 적합하도록 개선한 새로운 TCP 알고리즘이 제안되고 있다. TCP 윈도우 크기를 크게 하고, 호스트의 버퍼 크기를 키움으로써 높은 성능을 유도하는 방법과 TCP 전송 윈도우 크기를 좀 더 빠르게 증가시키거나, 좀 더 천천히 감소시킴으로써 높은 성능을 유도하는 방법, 그리고 큐잉 지연(queueing delay)을 사용하여 혼잡 정도를 예측하여 성능을 개선하는 방법들이 있는데, 그 중 대표적인 메커니즘인 HSTCP[2]에 대해서 알아본다.

Floyd에 의해 제안된 HSTCP(HighSpeed TCP)는 광대역 네트워크의 효율적인 사용을 위해 제안된 대표적인 TCP 메커니즘으로, 기존 TCP 응답 기능을 수정하여 네트워크 상황에 따라 AIMD의 증가, 감소 변수를 동적으로 변경하는 새로운 TCP 메커니즘이다. RTT 내에 수신된 각각의 ACK에 대해, 혼잡 윈도우 크기를 증가시키거나 감소시킬 경우 현재 cwnd 크기에 따라 증가와 감소 변수를 변경하여 증감폭을 조정하고, 패킷 손실 확률을 반영하여 cwnd 크기를 조절한다. 즉, AIMD 알고리즘에서 혼잡 윈도우의 증가 인자의 범위를 1(기존 TCP)에서 73패킷까지, 감소 인자의 범위를 0.5(기존 TCP)에서 0.09까지로 수정하였다. 이 효과로 현재의 혼잡 윈도우가 클수록 증가하는 크기가 커지고, 손실에 의한 혼잡 윈도우 감소량이 작아지므로, 가용 대역확보에 필요한 시간이 단축되고, 손실에 의한 회복 속도가 빨라진다.

하지만, 광대역 네트워크를 효율적으로 사용하기 위해 제안된 HSTCP는 TCP의 주요 특성인 확장성(scalability), 공정성(fairness), 안정성(stability) 및 호환성(friendliness) 등의 모든 면에서는 훌륭한 성능을 나타내고 있지는 않다. 특히 패킷 손실이 낮은 특정 환경에서만 우수한 성능을 보이고 있으며, 경쟁 TCP간 공정성 등에서도 문제점을 보이고 있다[1]. 따라서 다양한 환경 및 특성을 가지고 있는 광대역 네트워크에서 향상된 성능을 가지는 TCP 혼잡 제어 알고리즘에 대한 계속적인 연구가 필요하다.

## Ⅲ. ACA-TCP 알고리즘 제안

본 절에서는 광대역 네트워크에서 기존 TCP의 혼잡 윈도우 메커니즘으로 인하여 높은 가용 대역폭을 이용하지 못하는 문제점을 해결하고, 공정성을 개선하기 위해 본 논문에서 제안한 ACA-TCP(Advanced Congestion Avoidance - TCP) 알고리즘에 대해서 알아본다.

### 1. 제안한 TCP 알고리즘의 기본 개념

본 논문에서 제안한 ACA-TCP 알고리즘은 광대역 네트워크의 CA 구간에서 제공하는 대역폭을 효율적으로 활용하기 위하여 기존 TCP 혼잡 회피 알고리즘을 개선한 것이다. ACA-TCP 메커니즘의 주요 구성은 네 부분으로 나눌 수 있다. 즉, 그림 1에서와 같이 A 구간, B 구간, 그리고 C 구간 크게 구성된다. A 구간은 기존 TCP의 SS구간과 동일하게 cwnd가 지수 함수의 증가 형태를 나타내며, B 구간은 SS 이후 CA에서 cwnd가 선형 함수 형태 증가를 하는 단계이고, C 구간에서는 세 개의 중복 ACK가 발생하여 cwnd 값이 반으로 감소한 후 대역폭 측정에 의해 새로이 ssthresh 값을 설정한 뒤 다시 SS 구간과 같이 지수 함수 증가를 한다. 다시 C 구간의 SS 이

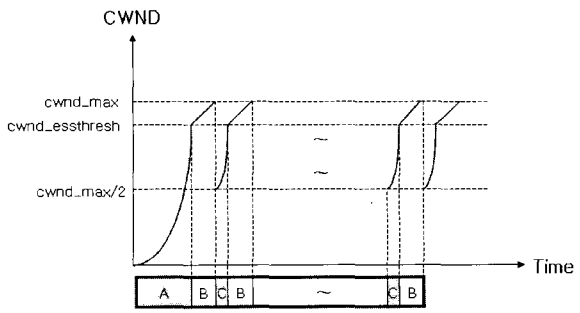


그림 1. ACA-TCP 기본 알고리즘에 의한 혼잡 윈도우 변화

Fig. 1. Congestion window behavior by Basic ACA-TCP algorithm.

후 CA에서 cwnd가 선형 함수 형태 증가를 하는 B 구간이 반복된다.

기존 TCP의 경우 SS 임계값이 적절히 예측되지 못하여 망에 트래픽이 급격히 유입될 경우 다량의 패킷 손실이 일어날 수 있고, 세 개의 중복 ACK 발생 후 혼잡 윈도우가 선형적으로 증가하게 되어 가용한 대역폭을 충분히 사용하는데 많은 시간이 경과되는 단점을 가진다. 제안한 알고리즘에서는 이를 개선하기 위해 가용 대역폭을 측정을 통해 예측하여 SS 임계값을 정하고 A, C 구간에서 지수적인 혼잡 윈도우의 증가를 수행하여 신속하게 최대 cwnd 값에 근접한 지점까지 증가시킨 후 B 구간에서는 기존 TCP의 CA 구간처럼 동작하도록 하여 가용한 최대 대역폭을 안정적으로 활용하도록 하였다.

### 가. A 구간 알고리즘

A 구간은 처음 TCP 연결시작시 또는 타임 아웃으로 cwnd가 1이 되고 나서 송신자와 수신자가 패킷을 주고받으면서 SS에 의하여 최대 가용한 대역폭까지 지수 함수 형태로 증가를 반복하면서 cwnd가 증가하는 구간이다. 가장 대표적인 TCP 알고리즘인 TCP Reno의 SS 구간에서는 전송된 세그먼트에 대한 ACK가 수신될 때마다 혼잡 윈도우 크기가 식 (1)과 같이 1씩 증가하며, 이 구간은 CA 구간으로 변경되는 지점으로 정해 놓은 cwnd_essthresh(estimated ssthreshold) 값까지 계속 cwnd가 지수함수 형태로 급격하게 증가하도록 하였다. 이때 cwnd_essthresh는 송신측에 ACK가 수신될 때 마다 새로이 설정된다.

$$cwnd += 1.0 \tag{1}$$

제안한 ACA-TCP 알고리즘은 전 구간에서 TCP Reno와 달리 송신측에 추가적인 알고리즘을 수행한다. 즉, ACA-TCP는 패킷 손실을 효과적으로 줄이고 타임아웃 발생을 사전에 방지하기 위하여 RTT와 ACK 정보를 이용하여 사전에 적정한 가용 대역폭을 측정하고 이를 기반으로 cwnd_essthesh 크기를 결정하도록 하였다. 측정된 최대 가용한 대역폭을 cwnd_essthresh로 변환 및 적용하고, cwnd_essthresh가 송신측에 ACK가 도착할 때마다 가용한 대역폭 상태에 따라서 동적으로 변화되게 함으로써 cwnd가 최대 값에 가까이 접근하였을 경우 cwnd_essthresh의 변화에 의하여 지수 함수 형태와 1차 함수 형태의 증가를 반복하면서 천천히 안정적으로 가용한 최대 윈도우 크기까지 증가하도록 하였다.

따라서, cwnd 값이 cwnd_essthresh 값 보다 작을 때는 SS를 수행하고 cwnd 값이 cwnd_essthresh 값 보다 크거나 같을 때는 CA를 수행함으로써, cwnd_essthresh 값의 변화에 따라 SS와 CA 현상이 혼합, 반복되고 최대 cwnd 값에 가까워질수록 cwnd 증가폭이 감소하게 되어 최대 대역폭을 점차 포화 수렴되는 안정된 특성을 가지게 된다. cwnd가 계속해서 cwnd_essthresh 값과 같거나 커지면 완전히 B 구간으로 전환된다.

또한 처음으로 네트워크 연결이 설정되어 SS를 수행할 때에는 네트워크가 어떤 환경인지 알 수 없으므로 초기 가용한 대역폭이 0으로 설정된 상태에서 송신측에서 ACK 수신시 마다 가용한 대역폭을 측정해 가며 안정적으로 증가하도록 하였다.

### 나. B 구간 알고리즘

B 구간은 식 (2)와 같이 기존 TCP Reno의 ssthresh 값 이후에 진행되는 기능과 동일한 알고리즘을 적용하였고, A 구간에서 정해진 cwnd_essthresh 값을 지난 뒤 혼잡 윈도우를 다시 선형 함수 형태로 최대 혼잡 윈도우 크기인 cwnd_max까지 서서히 증가하도록 하여 가용한 최대 윈도우 크기를 최대한 활용하면서 패킷 손실을 최소화하였다.

$$cwnd += \frac{1}{cwnd} \tag{2}$$

cwnd는 최대 윈도우 크기까지 증가한 후 타임 아웃 또는 세 개의 중복 ACK에 의하여 A 구간 또는 C 구간으로 전환된다.

### 다. C 구간 알고리즘

C 구간은 세 개의 중복 ACK 발생에 따른 TCP Reno의 CA 구간과는 다르게 선형 함수 형태의 증가를 하지

않고 다시 SS가 진행되도록 한 구간이다. TCP Reno의 경우 CA 구간에서 전송된 세그먼트에 대한 ACK가 수신될 때마다 혼잡 윈도우 크기가 1/cwnd 씩 증가하지만, 제안한 알고리즘은 B 구간에서 발생한 세 개의 중복 ACK에 의해 cwnd 값이 반으로 감소한 상태에서 측정된 cwnd_essthresh까지 SS를 다시 수행하도록 하였다. C 구간에서도 A 구간에서와 같이 안정된 지수 함수 형태 증가를 하고, 타임 아웃 발생을 사전에 방지하면서 패킷 손실을 효과적으로 줄이기 위하여 사전에 적정한 최대 가용 대역폭을 측정하고, 그 값을 cwnd_essthresh로 다시 변환 및 적용되도록 하였다. A 구간 알고리즘에서 설명한 바와 같이 C에서 다시 B 구간으로 변경되는 시점에는 SS와 CA 현상이 혼합, 반복되어 수행하면서 서서히 증가하고 cwnd가 계속해서 cwnd_essthresh 값과 같거나 커지면 완전히 B 구간으로 전환된다.

### 2. 대역폭 측정방법

A와 C 구간에서 cwnd_essthresh 값을 정하기 위한 대역폭 측정 알고리즘은 기존에 제안된 방법으로 그림 2와 같이 TCP 송신자 측에서 수신한 ACK를 통해 계산되어진다[5]. 임의의 $k-1$ 번째 ACK에 의해서 계산되어진 대역폭을 $BW_{E,k-1}$라 하면, $k$ 번째 ACK가 수신되었을 때 측정한 대역폭 $BW_{E,new}$ 값은 식 (3)과 같다. $T_k$는 $k-1$ 번째 ACK와 $k$ 번째 ACK가 수신된 시간 차이를 나타내며, $Packetsize_k$는 $k-1$ 번째 ACK와 $k$ 번째 ACK 사이에 확인 응답된 패킷의 총 크기를 나타낸다. 본 논문에서는 대역폭 변동성을 완화하기 위해 식 (4)를 사용하여 $k$ 번째 ACK가 도착한 시점의 대역폭 $BW_{E,k}$를 EWMA(Exponentially Weighted Moving Average) 방법으로 계산한다. 파라미터 α를 사용하여 새롭게 측정한 대역폭의 적용 비율을 조절하여 적정 대역폭을 측정할 수 있다. 이 EWMA 기법에 의하여 cwnd_essthresh 값이 급격히 최대 값에 도달되지 않도록 함으로써 SS 수행시 가용한 최대 cwnd 값에 근접하였을때 cwnd가 크게 변화하지 않고 안정적으로 최대 가용 대역폭을 활용하도록 하였다.

$$BW_{E,new} = \frac{BW_{E,k-1} * RTT + PacketSize_k}{RTT + T_k} \quad (3)$$

$$BW_{E,k} = BW_{E,k-1} * \alpha + BW_{E,new}(1-\alpha) \quad (4)$$

측정한 가용 대역폭은 식 (5)과 같이 A 구간과 C 구간에서 SS로 인한 지수 형태 증가를 CA를 위한 1차 함수 증가로 전환하기 위한 기준이 되는 SS 임계치 cwnd_essthresh 값을 정하는데 사용된다.

$$cwnd\_essthresh = \frac{BW_{E,k} * RTT}{MSS} \quad (5)$$

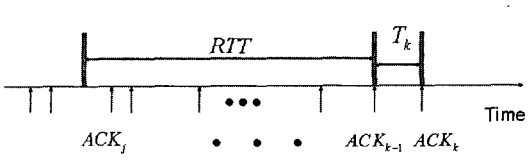


그림 2. ACA-TCP에서 대역폭 측정 방법
Fig. 2. Bandwidth measurement method in ACA-TCP.

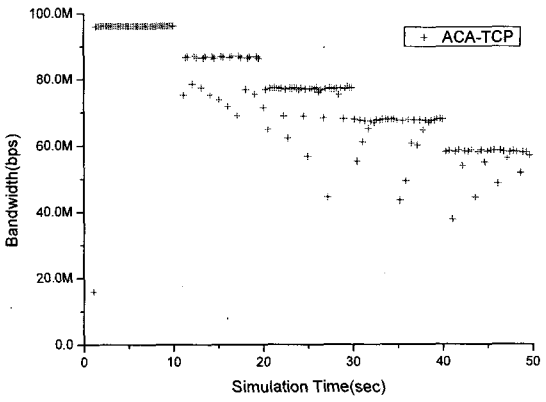


그림 3. ACA-TCP에서 대역폭 측정 결과
Fig. 3. Bandwidth measurement result in ACA-TCP.

그림 3은 시뮬레이션을 통해 가용 대역폭이 제대로 측정되고 있는지 확인한 결과이다. UDP 백그라운드 트래픽 양을 통해 가용 대역폭을 변경하면서 TCP 송신자에서 ACK가 수신되었을 때마다 식 (3)과 (4)에 의해서 측정한 가용 대역폭을 나타낸다. 최대 대역폭이 100Mbps인 상태에서 매 10초 마다 UDP 백그라운드 트래픽을 10Mbps씩 증가 시키면서 50초간 측정하였다. 그림에서 보는 바와 같이 측정한 가용 대역폭이 최대 대역폭 100Mbps에 근접한 상태에서 매 10초마다 10Mbps씩 감소함에 따라 이 알고리즘이 가용한 대역폭을 성공적으로 측정하고 있음을 알 수 있다.

## Ⅳ. 시뮬레이션 및 성능 평가

본 절에서는 광대역 네트워크에 적합하도록 제안된 HSTCP와 본 논문에서 제안한 ACA-TCP에 대해서 ns-2[6] 시뮬레이션을 통하여 비교 분석하였다. 시뮬레

이션은 단일 플로우(single flow)와 멀티 플로우(multiple flow)에 대해서 수행하였으며, 각 알고리즘에 대한 성능과 공정성에 대하여 성능 평가를 수행하였다.

### 1. 단일 플로우에서의 시뮬레이션 및 성능 평가

단일 플로우에서의 시뮬레이션 환경은 그림 4와 같이 광대역 네트워크 환경에 맞게 구성하였다. 각 네트워크는 3개의 노드와 두 개의 링크(link)로 구성하였고, 한 링크는 데이터 속도와 지연(delay)을 1Gbps와 1ms로 하였으며, 나머지 링크는 400Mbps와 49ms로 하여 병목구간이 발생하도록 구성하였다. 또한 각각의 패킷 크기는 1,000byte, 병목 구간의 queue 크기는 3,000 패킷인 상태에서 랜덤 로스 발생률(random loss rate)을 $10^{-4}$, $10^{-5}$, $10^{-6}$ 및 $10^{-7}$으로 변경하며 다양한 상황에서 각각의 TCP 성능을 분석하였다.

그림 5에서는 랜덤 로스 발생률을 $10^{-5}$로 설정한 상태에서 본 논문에서 제안한 ACA-TCP와 HSTCP의 cwnd가 변화되는 모습을 약 200초 동안 나타낸 것이다. 전반적으로 제안한 알고리즘이 HSTCP 보다 가용한 대역폭을 안정하게 확보하고 많은 패킷을 전송함을 알 수 있다. 특히, 패킷 손실이 발생하였을 경우 cwnd가 반으로 떨어

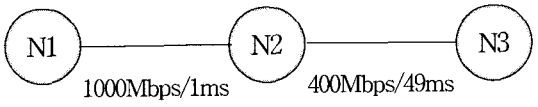


그림 4. 단일 플로우에서의 시뮬레이션 네트워크 토폴로지

Fig. 4. Simulation network topology for single flow.

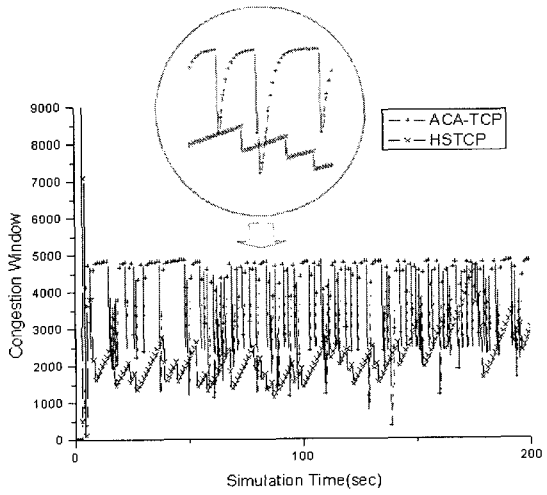


그림 5. ACA-TCP와 HSTCP의 혼잡 윈도우 변화

Fig. 5. Congestion window behavior of ACA-TCP and HSTCP.

진 상태에서 ACA-TCP는 빠르게 최대 cwnd에 근접한 값까지 cwnd를 신속하게 증가시켰다가 패킷 손실 발생 전에 cwnd-essthresh의 변화에 따라 증가속도를 낮춤으로써 가용한 대역폭을 충분히 활용하고 있음을 보여준다.

그림 6은 랜덤 로스 발생률(random loss rate)이 $10^{-5}$인 환경에서 시뮬레이션한 결과로써, 초기 약 50초 동안 cwnd가 변화하는 모습이다. 본 논문에서 제안한 ACA-TCP는 HSTCP와 달리 SS 출발시 EWMA 기법을 이용한 대역 예측 기법을 적용하여 안정적으로 cwnd가 증가하도록 한 모습을 볼 수 있다. 이로 인하여 타임 아웃 등이 발생하지 않고 안정적으로 가용한 윈도우 크기로 포화 특성을 나타내며 증가하였다. 또한 송신측에 수신된 ACK간 시간 간격이 늘어나게 되면 네트워크 혼잡이 발생하는 것으로 판단하고 cwnd_essthresh 값을 낮추

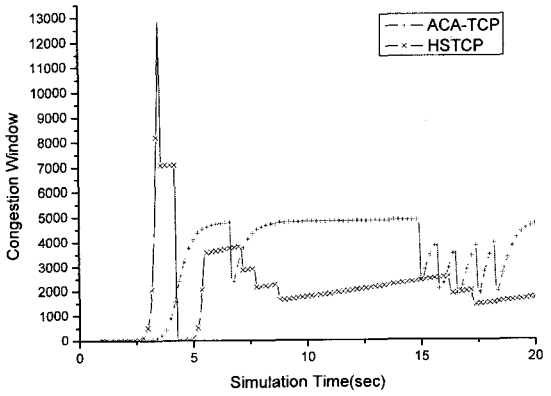


그림 6. ACA-TCP와 HSTCP의 시동구간 및 혼잡회피 구간 혼잡 윈도우 변화

Fig. 6. Congestion window behavior of ACA-TCP and HSTCP during startup and congestion avoidance phase.

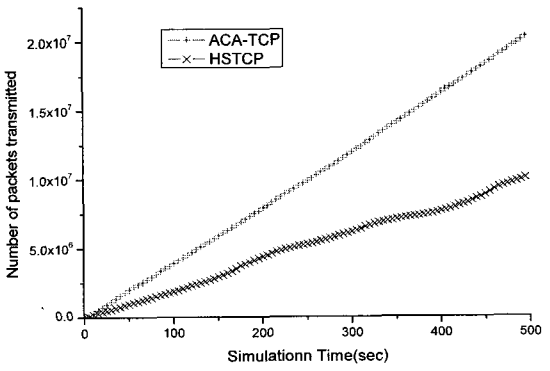


그림 7. 단일 플로우에서 성공적으로 전달된 누적패킷 수

Fig. 7. Cumulative number of packets successfully transmitted in single flow.

었다. 이때 cwnd_essthresh의 변화에 의하여 SS와 CA를 혼합, 반복 수행함으로써 cwnd가 완만하게 포화 상태로 증가하였으며, cwnd가 최대 cwnd-essthresh 값과 같거나 클 경우 완전히 선형 함수 형태의 증가함에 따라 가용한 대역폭을 빠르게 확보하면서 대량의 패킷 손실이 발생하는 문제점을 해결할 수 있었다.

또한, 그림 6은 세 개의 중복 ACK가 발생할 때 CA 구간에서 혼잡 윈도우가 ACA-TCP와 HSTCP의 각 알고리즘에 따라 어떻게 진행되고 변화하는지도 보여준다. HSTCP가 CA 구간에서 cwnd가 높은 값에서는 빠르게 증가하고 cwnd가 낮은 값에서는 느리게 증가하는 것과 달리 ACA-TCP는 혼잡 회피를 수행하는 C 구간에서 가용한 혼잡 윈도우를 빠르게 확보하기 위해 CA 구간에서 SS를 수행하고, SS 구간에서 예측된 대역폭을 바탕으로 혼잡 윈도우의 cwnd-essthresh 값을 계산하며, 그 값이 결정되면 다량의 패킷 손실 발생 방지를 위한 선형 증가 구간 시점을 알 수 있게 된다.

그림 7은 본 논문에서 제안한 ACA-TCP와 HSTCP의 성능 비교 시뮬레이션에 의한 시뮬레이션 시간이 진행됨에 따른 성공적으로 전달된 누적 패킷 수의 비교 결과를 나타낸다. 그림에서 보는 바와 같이 랜덤 로스 발생률이 $10^{-5}$인 상태에서 500초가 경과되었을 때 약 두 배 정도 더 많은 패킷이 성공적으로 전달되었다. 시간이 경과할수록 랜덤 로스 등에 의한 CA 구간이 다량으로 발생함에 따라 그 차이가 더 많이 나고 있음을 알 수 있다.

그림 8은 $10^{-4}$에서 $10^{-7}$까지 랜덤 로스 발생률을 변화시켰을 때 ACA-TCP와 HSTCP의 패킷 전송 성능을 비교한 것이다. 그림 7에서 로스 발생률이 증가함에 따라 낮은 cwnd 값에서는 느린 속도로 cwnd가 증가하는 HSTCP의 특성에 의해 ACA-TCP에 비해 HSTCP의 성능이 더 큰 폭으로 떨어지고 있음을 알 수 있다. 이 결과에 의하여 ACA-TCP가 HSTCP 보다 전송 지연 및 패킷 손실이 자주 발생할 수 있는 광대역 네트워크에서 더 우수한 성능을 나타내고 있음을 알 수 있다.

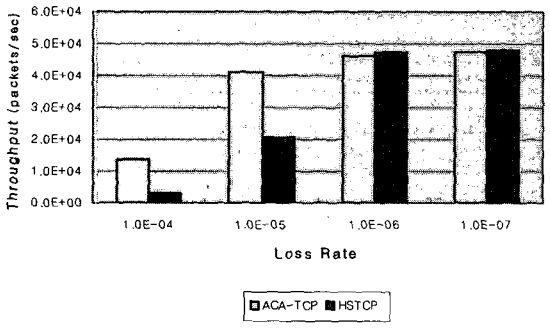


그림 8. 단일 플로우에서 손실 발생률에 따른 ACA-TCP와 HSTCP의 평균 전달률

Fig. 8. Time averaged throughput versus loss rate for ACA-TCP and HSTCP in single flow.

## 2. 멀티 플로우에서의 시뮬레이션 및 성능 평가

멀티 플로우에서의 시뮬레이션 환경은 공정성과 전송 성능을 분석하고 RTT에 따른 공정성을 평가하기 위한 것이다. 단일 플로우에서와 같이 광대역 네트워크 환경에 적합하게 그림 9와 같이 구성하였다.

이 시뮬레이션 환경은 4개의 노드와 3 개의 링크로 구성하였고, 송신 노드와 연결되어 있는 두 링크는 데이터 속도와 지연을 1,000Mbps와 1ms로 하였으며, 나머지 링크는 400Mbps와 49ms로 하여 병목구간이 발생하도록 구성하였다. 각각의 패킷 크기는 1,000byte, 병목 구간의 queue 크기는 3,000 패킷인 상태에서 랜덤 로스 발생률을 $10^{-4}$, $10^{-5}$, $10^{-6}$ 및 $10^{-7}$으로 변경하며 다양한 환경에서의 시뮬레이션을 수행하였다. 노드 N1-1과 노드 N1-2를 송신 노드, N3를 수신 노드로 설정하였고, 먼저 노드 N1-1이 패킷 전송을 시작하고, 20초 후에 동일 TCP 알고리즘

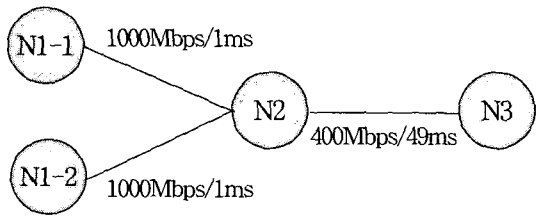


그림 9. 멀티 플로우에서의 시뮬레이션 네트워크 토폴로지

Fig. 9. Simulation network topology for multiple flows.

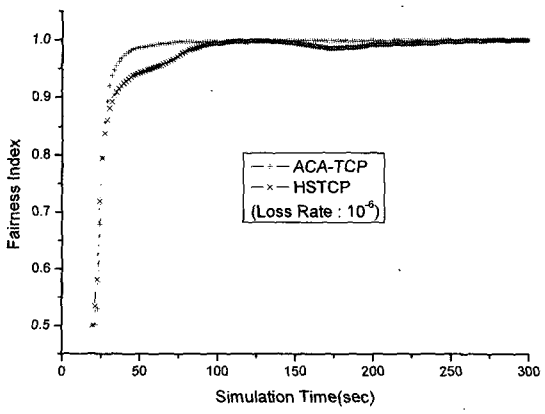


그림 10. ACA-TCP와 HSTCP의 공정성 인덱스 (loss rate:$10^{-6}$)

Fig. 10. Fairness index of ACA-TCP and HSTCP. (loss rate:$10^{-6}$)

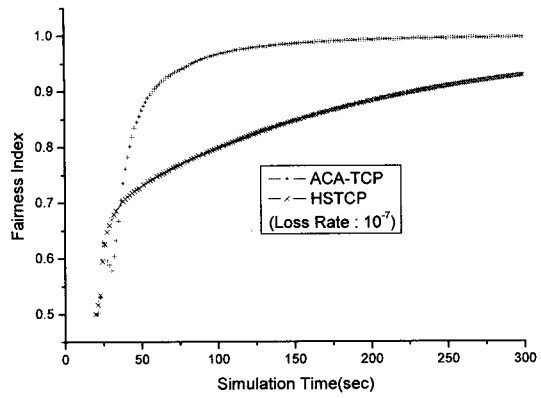


그림 11. ACA-TCP와 HSTCP의 공정성 인덱스 (loss rate:$10^{-7}$)

Fig. 11. Fairness index of ACA-TCP and HSTCP. (loss rate:$10^{-7}$)

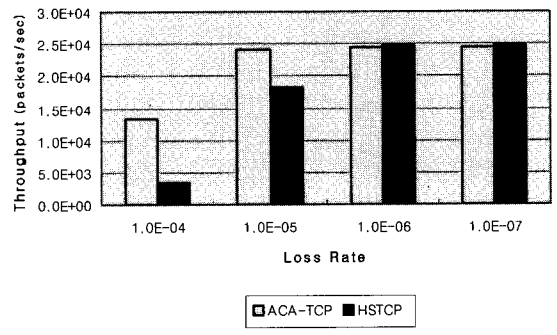


그림 12. 멀티 플로우에서 손실 발생률에 따른 ACA-TCP와 HSTCP의 평균 전달률

Fig. 12. Time averaged throughput versus loss rate of ACA-TCP and HSTCP in multiple flows.

에 의해 노드 N1-2에서 패킷 전송을 시작하도록 하였다. 이때 약 300 초 동안 시뮬레이션한 결과를 통하여 공정성과 패킷 전송 결과에 대한 각 알고리즘의 성능을 분석하였다.

그림 10과 11은 랜덤 로스 발생률을 각각 $10^{-6}$과 $10^{-7}$으로 변경하면서 ACA-TCP와 HSTCP의 공정성을 비교한 것으로 식 (7)으로 표현되는 Jain's 공정성 인덱스(fairness index)를 통해 나타내었다[7]. 공정성 인덱스 값은 1.0일 경우 100% 공정함을 의미한다. 그림 9와 10에서 보는 바와 같이 여러가지 랜덤 로스 발생률에 있어서 ACA-TCP가 HSTCP보다 1.0에 가까운 그래프 결과를 보이며 더 공정하게 전송기회를 부여하고 있음을 알 수 있다.

$$fairness\ index = \frac{(\sum x_i)^2}{n \times \sum (x_i)^2} \qquad (7)$$

그림 12는 동일한 멀티 플로우 환경에서 랜덤 로스 발생률의 변화에 따른 ACA-TCP와 HSTCP의 성능을 나타낸 그림으로, 두 송신 노드에서 성공적으로 전달된 누적 패킷 수를 합하여 표현하였다. 이 결과에 의하여 성공적으로 전달된 누적 패킷 수의 경우에 있어서도 ACA-TCP를 HSTCP와 비교할 때 랜덤 로스 발생률이 낮을 때는 비슷한 성능을 보이지만 랜덤 로스 발생률이 높아질수록 훨씬 우수한 성능을 보였다.

따라서, HSTCP의 경우 낮은 로스 발생률에서는 패킷 전송량은 높지만 공정성이 나빠지는 문제점이 발생하고, 로스 발생률이 높아질수록 공정성은 좋아지지만 패킷 전송량이 급격히 낮아지는 문제를 나타낸다. 하지만 본 논문에서 제안한 ACA-TCP는 패킷 전송 성능과 공정성 측면에 있어서 우수한 성능을 보임을 알 수 있다

그림 13은 RTT 공정성을 평가를 위하여 멀티 플로우 시뮬레이션 환경을 구성한 것으로, 본 논문에서 제안한 ACA-TCP, TCP Reno와 HSTCP 알고리즘을 통하여 패킷 전송 성능에 따른 RTT 공정성을 비교하였다. RTT에 따른 공정성 평가는 다른 RTT를 가진 플로우를 동시에 경쟁시켰을 때 RTT 비율에 따라 얼마나 공정하게 대역폭을 분배하여 사용하는지를 평가하는 것으로 광대역 네트워크에 적합하게 설계된 기존 TCP 알고리즘은 심각한 RTT 불공정성(unfairness)을 보이고 있다[1].

이 시뮬레이션 환경은 노드 N1-1과 노드 N1-2를 송신 노드, 노드 N3을 수신 노드로 정하였고, 노드 N1-1에서 시작하는 플로우의 지연을 20ms로 고정한 상태에서 노드 N1-2에서 시작하는 플로우의 지연을 각 시뮬레이션마다 20ms(1배), 60ms(3배), 120ms(6배)로 변경하면서 동일한 알고리즘을 동시에 두 플로우에서 패킷 전송을 수행하도록 한 상태에서 패킷 전송 성능을 측정하였다. 이때, 두 플로우가 경쟁을 하고 병목현상이 발생하는 구간인 노드 N2와 노드 N3 사이의 랜덤 로스 발생률은 $5\times10^{-5}$ 로 설정하였다.

표 1은 RTT 비율에 따른 각 알고리즘의 전송 성능 차이를 비로 나타낸 것이다. 측정 결과 두 플로우간 RTT 비율에 따라 패킷 전송 성능 비율도 같이 변화해야 하지

표 1. RTT 비율에 따른 전송 성능 비교

Table 1. Throughput comparison for RTT ratio.

| RTT 비율 | 1배 | 3배 | 6배 |
|---|---|---|---|
| TCP Reno | 0.85 | 11.51 | 21.34 |
| HSTCP | 2.35 | 13.66 | 39.82 |
| ACA-TCP | 0.96 | 1.66 | 4.23 |

만 HSTCP의 패킷 전송 성능 비율의 차이가 가장 많이 나고, 그 다음으로 TCP Reno의 차이가 많이 나며, ACA-TCP가 RTT 변화율에 따라 가장 근접한 전송 성능 비율을 보임으로써, 본 논문에서 제안한 ACA-TCP가 RTT 공정성에 있어서도 HSTCP보다 개선된 성능을 보임을 알 수 있다.

## V. 결 론

본 논문에서는 기존 TCP의 중요한 메커니즘인 혼잡 제어 알고리즘을 개선하여 가용한 광대역 네트워크 용량을 효율적으로 활용할 수 있는 알고리즘을 제안하였으며, ns-2를 이용하여 광대역 네트워크의 효율적인 사용을 위해 제안되었던 대표적인 알고리즘인 HSTCP와 성능을 비교 분석하였다.

시뮬레이션을 수행한 결과 본 논문에서 제안한 ACA-TCP 알고리즘이 임의의 다양한 랜덤 손실을 발생시켰을 경우 단일 플로우 뿐만 아니라 멀티 플로우에서도 랜덤 로스 발생률이 낮을 때는 HSTCP와 동등한 성능을 나타내지만 로스 발생률이 높아졌을 경우 훨씬 더 많이 패킷을 성공적으로 전달함에 따라 효율적으로 가용한 대역폭을 사용하였음을 알 수 있었다. 또한, 새로운 연결이 설정되었을 때 시작하는 SS 구간에서도 대량의 패킷 손실이 발생하기 전에 CA 구간으로 전환됨에 따라 최대 가용한 대역폭을 안정적으로 유지하였다. CA 구간에서도 개선된 SS 알고리즘을 적용하여 신속하게 가용한 대역폭을 확보함에 따라 기존 TCP에서의 대역효율 문제를 개선하였다.

다양한 랜덤 로스 발생률을 적용한 멀티 플로우 환경에서의 공정성 평가에 있어서 공정성 인덱스 분석 결과 ACA-TCP가 HSTCP에 비해 모든 환경에서 1.0에 더 근접한 결과를 보였으며, 특히 HSTCP를 사용할 경우 패킷 전송률이 좋아지나 공정성이 급격히 떨어지는 낮은 랜덤 로스 발생률 환경에서 제안한 알고리즘이 더 효과적으로 공정성을 개선함을 보였다. 또한 TCP Reno와 HSTCP와의 RTT 공정성 비교, 평가에서도 ACA-TCP가 RTT 차이에 의한 전송 성능 차이가 심하게 발생하지 않아 RTT 공정성에서도 개선된 결과를 보였다.

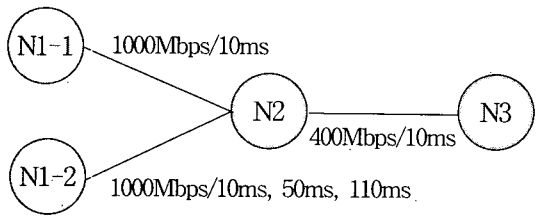


그림 13. RTT 공정성 평가를 위한 시뮬레이션 네트워크 토폴로지

Fig. 13. Simulation network topology for RTT fairness estimation.

본 시뮬레이션에서는 협대역 네트워크의 대표적인 알고리즘인 TCP Reno와의 공정성이 고려되지 않았지만 TCP Reno가 수행되는 협대역 네트워크 환경에서는 HSTCP와 마찬가지로 Low_window가 38 MSS 이하일 경우 TCP Reno를 그대로 적용하고 그 이상의 광대역 네트워크에서는 본 논문에서 제안된 알고리즘이 수행되도록 하면 TCP Reno와의 공정성 문제가 해결된다.

## 참 고 문 헌

[1] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control(BIC) for Fast Long-Distance Networks," Proceedings of IEEE INFOCOM 2004, Vol. 4, pp. 2514-2524, Mar. 2004.

[2] S. Floyd, "HighSpeed TCP for large Congestion windows," RFC 3649, Dec. 2003.

[3] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," end2end-interest Mailing List, Apr. 30, 1990.

[4] L. Brakmo and L. Peterson, "TCP Vegas : End to End Congestion Avoidance on a Global Internet," IEEE Journal on Selected Areas in Communication, Vol. 13, No. 8, Oct. 1995.

[5] D. D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," IEEE/ACM Trans. Networking, vol. 6, pp. 362-373, Aug. 1998.

[6] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns.

[7] R. Jain, 'The Art of Computer Systems Performance Analysis : Techniques for Experimental Design, Measurement, Simulation, and Modeling,' John Wiley & Sons, Inc., 1991.

## 저 자 소 개


**나 상 완**(정회원)
1995년 한남대학교
정보통신공학과 학사
2006년 충남대학교
정보통신공학과 석사
1995년~현재 (주)KT 근무
<주관심분야 : 인터넷, 데이터 통신, 초고속 통신, BcN>

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

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

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