논문 2009-46TC-2-14

TEIN2를 활용한 고대역 전송계층 프로토콜 테스트베드 구축 및 성능 분석

(Testbed Implementation and Performance Evaluation of High Speed Transport Protocols using TEIN2 Networks)

이 기 라*, 소 상 호*, 최 윤 철*, 박 만 규*, 이 재 용**, 김 병 철**, 김 대 영**

(Gi Ra Lee, Sang Ho So, Yun Chul Choi, Man Kyu Park, Jae Yong Lee, Byung Chul Kim, Dae Young Kim)

요 약

현재 초고속 인터넷 사용자가 급증하고 있고, 광대역 네트워크 인프라의 구축이 늘어나고 있다. 하지만, 현재 인터넷에서 널리 사용되고 있는 TCP는 기존의 인터넷 환경에 적합하며, 대역폭과 지연의 곱이 큰 네트워크 환경의 트래픽 전송에 있어서는 그 효율성이 낮은 상태이다. 이런 문제점을 개선하기 위해 최근 광대역 네트워크에 적합하도록 개선된 혼잡 제어 알고리즘에 관한 많은 연구가 진행되고 있다. 그러나 그 성능 검증이 대부분 시뮬레이션에 의해 이루어지고 있는 실정이다. 따라서 본 논문에서는 기존 TCP의 문제점과 이를 해결하기 위해 제안된 고대역 전송 프로토콜(High Bandwidth Transport Layer Protocol)을 간단히 살펴보고, 이의 성능 검증을 위한 실제 테스트베드를 구축하였으며 TEIN2 국제연구망을 통하여 한국, 중국, 독일 3개국 간 국제협력 실험을 통해 고대역 전송 프로토콜의 성능을 측정하고 문제점을 분석하였다.

Abstract

Recently, high-speed Internet users and broadband network infrastructure are increasing. However, the TCP protocol widely used in the Internet is an appropriate transport only for the legacy Internet, it is inefficient for traffic transport for network environments with large bandwidth-delay product. In order to remedy this problem, there have been many researches about improved congestion control algorithms for broadband networks. In these studies, most of performance evaluation has been done by simulations. In this paper, after we reviewed the proposed high bandwidth transport layer protocols, we implemented a real testbed, measured the performance and analyzed the problems of high-speed transport protocols through the international research network TEIN2 between three nations, Korea, China, and Germany.

Keywords: TCP, high-speed Internet, Bandwidth-delay product, TEIN2

I. 서 론

현재 인터넷을 통해 전송되고 있는 응용데이터의 80%~90%는 대표적인 전송계층 프로토콜인 TCP를 사용하여 전송되고 있다. TCP는 종단간의 적정 대역폭을 찾아 동작하는 방식으로 자신의 혼잡윈도우(congestion window)를 망 환경에 따라 AIMD(additive increase multiplicative decrease) 방식으로 변화시키면서 적정전송률을 유지하도록 동작하고 있다. 이러한 동작방식은 기존의 인터넷에서는 매우 적절한 성능을 나타내어 인터넷을 통한 데이터 전송에 큰 기여를 하였다.

그런데 최근 들어 인터넷은 백본망뿐만 아니라 종단 지역 망도 수 Gbps에서 수십 Gbps에 이르는 고대역폭을 지원하는 고대역 네트워크로 발전하고 있다. 또한 의료용 영상이나 GRID 네트워크와 같은 다양한 형태의

* 학생회원, ** 평생회원, 충남대학교 정보통신공학과
(Chungnam National University)
*** 정회원, 한국전자통신연구소
(ETRI)
※ 본 연구는 한국과학재단 (No. R01-2006-000-10154-0)의 지원으로 수행된 연구 결과물입니다.
접수일자: 2008년9월1일, 수정완료일: 2009년2월17일

대규모 데이터 전송을 지원할 필요가 있는 인터넷 응용 서비스들이 개발되고 있다. 그러나 기존의 TCP의 동작 알고리즘 특성으로 인해 이러한 고대역폭을 가지는 망에서는 가용대역폭을 충분히 활용하지 못하는 단점을 가지고 있다. 수 Gbps 또는 수 십 Gbps의 대역폭을 충분히 활용하기 위해서는 혼잡윈도우의 크기가 매우 커져야 한다. 그런데 기존의 AIMD 방식으로 혼잡윈도우를 증가시키는 방식 (additive increase)에서는 가용대역폭을 충분히 활용하기 위해 혼잡윈도우를 증가시키기까지 매우 오랜 시간을 필요로 하며, 대부분 혼잡윈도우 증가 도중 발생하는 에러에 의해 혼잡윈도우가 절반으로 줄어 (multiplicative decrease) 혼잡윈도우가 큰 값을 가지도록 증가하기 어렵다. 때문에 기존의 TCP가 고대역을 제대로 활용하지 못하는 특성을 가지게 된다. 예를 들어 패킷길이가 1500바이트이고, RTT가 100msec인 TCP 연결이 정상상태에서 10Gbps 의 성능에 도달하기 위해서는 평균 혼잡윈도우의 크기가 83,333 세그먼트가 되어야 하며, 이를 위해서는 패킷손실이 매 5,000,000,000 패킷마다 1번 이하의 손실이 발생해야 한다. 따라서 이를 달성하기 위해서는 광케이블 등의 전송매체의 평균 비트오류율이 2×10^{-14} 이하를 만족해야 한다. 하지만 이는 현재의 네트워크 전송기술로는 달성하기 어려운 조건이다.

따라서 본 논문에서는 고대역 전송망에서의 기존 TCP의 문제점에 대해 설명하고, 이를 해결하기 위하여 고대역을 가지는 망에서 TCP의 혼잡윈도우를 기존과 다른 방식으로 제어함으로써 TCP의 가용대역폭 활용도를 증가시키도록 제안된 고대역 전송계층 프로토콜(High Bandwidth Transport Layer Protocol)을 간단히 살펴본다. 그리고 시뮬레이션을 통한 성능 검증이 아니라, 실제 리눅스 기반의 고대역 전송 프로토콜 테스트베드를 구축하여 그 성능을 실험하고 현재 제안된 고대역 전송 프로토콜의 성능과 문제점을 분석한다.

Ⅱ. 고대역 전송 프로토콜 기술 개요

현재의 네트워크에 사용하는 TCP 응용과 달리 대역폭과 지연의 곱이(Bandwidth Delay Product, BDP) 큰 네트워크에서 신속한 대역을 확보하여 대역 효율을 높이고, 고정된 대역이 제공되는 네트워크에서는 중간 노드의 버퍼 크기와 무관하게 항상 최적의 성능을 나타내는 등 새로 등장하는 요구사항을 만족하고 새로운 네트워크의 특성을 잘 활용할 수 있는 새로운 TCP에 관한 많은 연구가 이루어지고 있다.

Sally Floyd 에 의해 제안된 HSTCP(HighSpeed TCP)[1]는 대역폭과 지연의 곱이 큰 광대역 네트워크의 효율적인 사용을 위해 TCP Reno를 개선한 대표적인 TCP 메커니즘으로, 기존 TCP의 AIMD(additive increase multiplicative-decrease) 알고리즘에서 혼잡윈도우의 증가인자의 범위를 1에서 73패킷까지, 감소 인자의 범위는 0.5에서 0.09까지로 수정하였다. 이 효과로 현재의 혼잡윈도우가 클수록 증가하는 크기가 커지고, 손실에 의한 혼잡윈도우 감소량이 작아지므로, 가용 대역 확보에 필요한 시간이 단축되고, 손실에 의한 회복 속도가 빨라진다. Tom Kelly가 제안한 STCP(Scalable TCP)[2]는 전송 윈도우의 크기를 좀 더 강하게 증가시키고 완만하게 감소시킴으로써 TCP의 패킷 손실 복구 시간(packet loss recovery time)을 줄여 성능을 향상시킨다. STCP는 혼잡 회피 구간에서 수신된 각 ACK에 대해 $cwnd = cwnd + 0.01$ 만큼 증가하고, 패킷 손실이 발생하였을 경우에는 현재 cwnd의 0.875배로 축소시키는 MIMD(multiplicative increase multiplicative decrease) 방식을 사용한다. 따라서 혼잡윈도우가 클수록 손실복구가 신속히 이루어진다. 그러나 HSTCP와 STCP 모두 현재의 혼잡 윈도우 크기가 클수록 증가속도가 더 빠르므로 혼잡 윈도우가 작은 연결에 대해 심각한 공평성 문제를 일으킨다.

HSTCP[1]와 STCP[2]가 패킷 손실을 기반으로 한 알고리즘인 반면, Fast(Fast AQM Scalable) TCP[3]는 지연을 기반으로 한 알고리즘으로서 혼잡의 신호로 큐잉 지연(queueing delay) 과 패킷 손실을 사용한다. 손실을 기반으로 한 접근은 단 하나의 bit 정보를 가지고 손실 확률로 사용하나 지연을 기반으로 한 접근은 멀티 bit 정보를 이용하므로 네트워크의 변동을 더 정확하게 판단한다. HSTCP는 광대역 네트워크에서 $cwnd$를 빠르게 증가시키고, 완만하게 감소시키는 알고리즘을 통하여 전송 성능과 반응성(responsiveness)을 개선하였지만 공정성과 안정성에 있어서는 TCP Reno보다 성능이 떨어지는 결과를 보인다. 하지만, FAST TCP는 각 기준에서 모두 개선된 결과를 보이고 있다.

BIC(Binary Increase Congestion) TCP[4]는 HSTCP[1]와 STCP[2]가 다른 RTT 지연을 가진 멀티 플로우가 같은 병목 대역을 사용하기 위해 경쟁을 할 때 심각한 RTT 불공정 문제를 가지고 있음을 밝히고, 이를 개선

하기 위하여 확장성(scalability)과 TCP 친화성(friendliness)를 제공하면서 큰 윈도우 상태에서 RTT 공정성을 제공하는 혼잡 제어 프로토콜을 제안한 알고리즘이다. BIC TCP의 주요 알고리즘은 크게 이진탐색 증가(binary search increase) 구간과 선형증가(additive increase) 구간으로 구성되어 있다. 패킷 손실이 발생하였을 때 BIC TCP는 multiplicative 요소(β)에 의하여 윈도우를 감소시킨다. 감소 전의 윈도우 크기가 최대값으로 설정되고, 감소 후 윈도우 크기가 최소값으로 설정된다. 그 다음 BIC TCP는 이 두 파라미터를 사용하면서 binary search increase를 수행하여 최대값 $W_{\max}$과 최소값 $W_{\min}$ 사이의 중간 지점으로 윈도우를 증가시킨다. 그러나 만약 중간 지점과 현재 최소값의 거리가 고정된 상수인 최대 증가 $S_{\max}$ 값보다 크면, 현재 윈도우 크기를 $S_{\max}$까지 선형 증가시킨다.

비록 BIC TCP가 현재 고속 네트워크 환경에서 우수한 확장성, 공정성과 안정성을 달성하고 있지만, BIC TCP의 증가함수는 특히 짧은 RTT 또는 저속(low speed) 네트워크 환경에서 너무 공격적이다. 또한 윈도우 제어에 있어서 여러 다른 단계들로 인해 프로토콜 분석하기가 매우 복잡하다. CUBIC TCP[5]는 이런 BIC TCP를 개선함으로써 BIC TCP의 윈도우 제어를 단순화하고, TCP 친화성과 RTT 공정성을 개선한 새로운 고속 TCP의 변형된 알고리즘이다. CUBIC TCP의 윈도우 증가함수는 큐빅(cubic) 함수이다. 이것의 모습은 BIC TCP의 증가함수의 모습과 매우 유사하다. CUBIC TCP의 혼잡 윈도우 증가함수는 아래 식과 같다.

$$W_{cubic} = C(t-K)^3 + W_{\max}$$

여기에서 C 는 스케일링 요소(scaling factor), t 는 마지막 윈도우 감소로부터 경과시간, $W_{\max}$ 는 마지막 윈도우 감소 바로 전 윈도우 크기이며, $K = \sqrt[3]{W_{\max}\beta / C}$ 이다. β 는 손실 이벤트 시간에 윈도우 감소를 적용하기 위한 일정한 배수(multiplication) 감소 요소이다. 즉, 마지막 감소 시간에 윈도우는 $\beta W_{\max}$ 만큼 줄인다. 윈도우는 윈도우 절감된 상태에서 매우 빠르게 증가한다. 하지만 $W_{\max}$ 에 가까워질수록 윈도우 증가를 낮춘다. $W_{\max}$ 주변에서 윈도우는 거의 증가하지 않는다. 그 다음 CUBIC은 윈도우가 천천히 증가하는 상태에서 더 가용한 대역폭이 있는지를 찾기 시작하고, $W_{\max}$로부터 벗어나는 것처럼 윈도우 증가를 촉진시킨다. $W_{\max}$ 주변에서의 느린 증가는 프로토콜의 안정성을 개선시키고, 프로토콜의 확장성을 확인한 다음에 $W_{\max}$ 로부터 벗어나 빨리 증가함으로써 네트워크 활용성을 증가 시킨다.

CUBIC은 확장성과 안정성을 유지하면서 BIC TCP의 공정성 특성을 개선하였으며, CUBIC의 주요 특징은 CUBIC 의 증가 기능이 실시간으로 규정되므로 RTT와는 무관하다는 것이다. 따라서 다른 RTT를 가진 플로우들이 같은 비율로 윈도우를 증가시켜도 RTT 공정성이 보장된다.

III. 고대역 전송 프로토콜 실험을 위한 테스트베드 구축

본 장에서는 고대역 전송 프로토콜 실험을 위한 소규모 테스트베드 구축 과정 및 고대역 전송 프로토콜의 성능 측정 과정을 설명한다.

본 논문에서 구축한 실험 환경은 <그림 1>과 같다. <그림 1>에서 보는 바와 같이 PC라우터에는 Dummynet[6]을 사용하였으며 이를 위해 다른 호스트들과 달리 FreeBSD 운영 체제를 설치하였다. 그리고 그 양편으로 2개의 sender와 2개의 receiver를 설치하였다.

네트워크나 프로토콜을 연구할 때 다양한 네트워크 환경을 emulation 할 필요가 있다. Dummynet[6]은 다양한 네트워크 인터페이스들을 통과하는 통신들의 대역폭과 큐 사이즈를 제한해서 별도의 설정 사항들을 관리하고 운용할 수 있도록 도와준다. 주로 네트워크 프로토콜을 시험하고, 대역폭을 관리하기 위한 목적으로 만들어졌으며, FreeBSD가 설치된 PC 라우터나 브릿지에 사용된다. 그리고 Dummynet을 Ipfw와 함께 사용하면 방화벽으로 지나가는 패킷의 대역폭 제어가 가능하다.

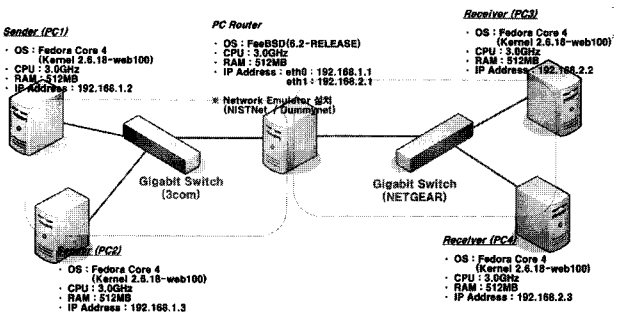


그림 1. 고대역 전송 프로토콜 테스트베드 구성
Fig. 1. Testbed construction of high bandwidth transport protocols.

```
dummynet# cd /usr/src/sys/i386/conf
dummynet# cp GENERIC mykernel
dummynet# vi mykernel
--------------------------------------------------------------------
+options         IPFIREWALL
+options         IPFIREWALL_DEFAULT_TO_ACCEPT
+options         IPDIVERT
+options         DUMMYNET
+options         DEVICE_POLLING
+options         HZ=1000
--------------------------------------------------------------------
dummynet# config mykernel
dummynet# cd ../conpile/mykernel
dummynet# make depnd
dummynet# make
dummynet# make install
```

그림 2. 커널 옵션 추가 및 커널 컴파일
Fig. 2. Addition of kernel options and kernel compile.

또한 Dummynet은 지나가는 패킷의 대역폭 제어나 확률 기반의 패킷 폐기 등의 처리가 가능하므로, 현실 세계에서 발생 가능한 다양한 네트워크 환경을 하나의 리눅스 라우터 상에 구축할 수 있게 한다.

Ipfw와 Dummynet을 모듈로 읽어 들일 수도 있지만, 기본적인 설정은 커널에서 해 주는 편이 좋다.

위의 <그림 2>는 기본적인 Dummynet 관련 커널 옵션이다. 커널 설정 옵션은 기본 커널인 GENERIC에 포함되어 있지 않으므로 Dummynet 사용이 가능한 새로운 커널을 생성하기 위해 <그림 2>와 같이 기본 커널인 GENERIC을 mykernel 이라는 이름의 새 커널로 복사한 후, mykernel 의 설정 파일에 <그림 2>의 옵션들을 추가하고 이를 컴파일 하였다. make install까지 모두 성공한 후, 이제 새로운 커널로 재부팅 하면 Dummynet의 사용이 가능하다.

Ipfw는 명령 행으로 규칙을 추가할 수 있고, 파일로 만들어진 규칙을 읽어 들이도록 할 수도 있다. Ipfw는 매우 간단하고 직관적인 문법을 갖고 있다. 기본적으로는 각 패킷을 어떻게 처리할 지의 규칙을 기술해 주면 되며 Ipfw 규칙의 기본적인 문법은 다음과 같다.

ipfw add [N] [prob X] 동작 PROTO from SRC to DST [인터페이스 정보] [옵션]

[N]은 각 규칙에 할당되는 번호이며, [prob X]는 0에서 1 사이의 값으로 패킷이 통과할 확률을 정한다. 이는 가상의 네트워크 환경을 구축하거나, 일부러 패킷을 확률적으로 버려야 할 때 사용할 수 있다. '동작' 으로는 크게 다음 [표 1]과 같은 것들이 있다.

"PROTO"는 전송하는 패킷의 프로토콜 타입으로 보통 ip, tcp, udp, icmp 등이 사용된다. 다음의 "from SRC to DST" 은 패킷을 보낸 곳과 갈 곳을 나타내는데, TCP/IP 시스템에서는 보통 IP 주소와 포트로 표현된다. Dummynet은 이러한 ipfw 문법에 맞추어 송신지

표 1. ipfw 동작
Table 1. Operation of ipfw.

command	동작 내용
allow	패킷 통과를 허가한다
deny	패킷 통과를 거부한다
unreach	패킷 통과를 거부하고 정해진 코드의 ICMP 통지를 한다.
reset	패킷 통과를 거부하고 TCP RST를 보낸다.
divert	정해진 포트의 divert(4) 소켓으로 보낸다. NAT에서 사용
fwd	패킷을 정해진 주소와 포트로 전달한다.

```
dummynet# ipfw add 100 allow all from any to any
/*pc 라우터를 통과하는 모든 패킷들을 허용*/
dummynet# ipfw add 100 pipe 1 all from 192.168.1.0 to 192.168.2.0
/*파이프 1번에 192.168.1.0/24 네트워크에서
  192.168.2.0/24 네트워크로 가는 패킷 모두에 대해 적용*/
dummynet# ipfw pipe 1 config bw 450Mbit/s
/*파이프 1번의 대역폭을 450Mbps로 제안*/
/*파이프에도 번호를 붙여 여러개를 만들 수도 있다.*/
dummynet# ipfw pipe 1 config delay 75ms
/*파이프 1번의 왕복 지연 시간을 150ms로 설정*/
dummynet#
```

그림 3. Dummynet을 이용한 대역폭 제한 예제
Fig. 3. Example of bandwidth limitation using dummynet.

와 목적지 사이에 파이프를 생성한다. 그리고 그 파이프의 대역폭을 제한하거나, 큐 사이즈를 조절하고, 혹은 파이프에 일부러 지연시간이나 패킷 손실 발생을 주어 다양한 네트워크 환경을 구축할 수 있다.

<그림 3>은 ipfw와 Dummynet을 사용하여 대역폭을 제한하고, 지연 시간과 패킷 손실을 발생시킨 간단한 예제이다.

먼저 PC 라우터의 방화벽을 통과하는 모든 패킷들을 허용하는 규칙을 정한다. 그 다음, 송신지 호스트가 있는 192.168.1.0/24 네트워크에서 목적지 호스트가 있는 192.168.2.0/24 네트워크로 가는 패킷의 경로, 즉 파이프 1번을 하나 추가하고, 이 파이프 1번의 대역폭은 450Mbps로 제한을 두었으며, 왕복 지연 시간을 150ms로 설정하는 과정을 보인다.

다음으로 PC라우터를 제외한 나머지 호스트들에 모두 Fedora Core 4를 설치하고, web100 패치를 위해 리눅스 커널 버전을 2.6.11에서 2.6.18 버전으로 업데이트 하였다. Web100[7]은 향상된 TCP 진단을 가능하게 하는 자동 튜닝기능과 TCP 도구들을 가지고 있다. 또한, 본 논문에서 그 성능을 측정하고자 하는 고대역 전송 프로토콜인 HSTCP, STCP, BIC TCP, CUBIC TCP 등의 알고리즘을 포함하고 있다.

따라서 본 논문의 테스트베드에 사용될 PC 라우터를 제외한 모든 호스트에 web100 패치를 하였다. 패치 과정은 <그림 4>와 같다.

```
[root@localhost /]# cd /usr/local/src
[root@localhost src]# tar -zxvf web100-2.5.12-200609221010.tar.gz
[root@localhost src]# cd /usr/src/kernel
[root@localhost kernel]# tar -jxvf linux-2.6.18.tar.bz2
[root@localhost kernel]# ln -s linux-2.6.18 linux
[root@localhost kernel]# linux
[root@localhost liunx]# patch -p1 <
/usr/local/src/web100/web100-2.6.8-2.5.12-200609221010.patch
/*새 리눅스 커널에 web100 패치 적용*/
```

그림 4. web100 패치 적용 과정
Fig. 4. Procedure of web100 patch application.

```
[root@localhost linux]# make menuconfig
[root@localhost linux]# vi .config
-----------------------------------------------------------------
# TCP congestion control

+CONFIG_TCP_CONF_BIC=y
+CONFIG_TCP_CONF_CUBIC=y
+CONFIG_TCP_CONF_HSTCP=y
+CONFIG_TCP_CONF_SCALABLE=y
+CONFIG_WEB100=y
+CONFIG_WEB100_STATS=y
+CONFIG_WEB100_FRERMS=384
+CONFIG_WEB100_GID=0
+CONFIG_WEB100_NET100=y
+CONFIG_WEB100_NETLINK=y
-----------------------------------------------------------------
[root@localhost linux]#
```

그림 5. 리눅스 커널 수정
Fig. 5. Modification of linux kernel.

```
❒ Enable Highspeed TCP as a default tcp congestion control:
[root@localhost /]# sysctl -w net.ipv4.tcp_congestion_control=highspeed
[root@localhost /]# cat /proc/sys/net/ipv4/tcp_congestion_control
highspeed
❒ Enable CUBIC as a default tcp congestion control:
[root@localhost /]# sysctl -w net.ipv4.tcp_congestion_control=cubic
[root@localhost /]# cat /proc/sys/net/ipv4/tcp_congestion_control
cubic
[root@localhost /]#
```

그림 6. 고대역 전송 프로토콜 선택 명령어
Fig. 6. Command for selection of high speed transport protocols.

새로운 버전의 커널 소스에 web100 패치를 적용한 후, 이에 맞게 커널 역시 새롭게 구성해야 하는데, 이것은 /usr/src/kernels/linux 디렉토리 아래 있는 .config 파일을 수정하면 된다. .config 파일은 숨김 파일이므로 ls 명령어에 -a 옵션을 이용하면 쉽게 찾을 수 있다. 그런데도 .config 파일을 찾을 수 없다면 make menuconfig 명령어를 이용하여 config 파일을 새로 만들 수 있다. 이제 .config 파일을 vi 에디터로 열어서 <그림 5> 에 보이는 다음의 항목들을 추가한다.

위의 커널 구성 과정까지 마친 후, 새로운 커널을 컴파일하고 새 커널 2.6.18-web100 버전으로 재부팅 하면 <그림 6>과 같이 간단한 명령어로 여러 고대역 전송 프로토콜을 선택적으로 사용이 가능하다.

IV. 고대역 전송 프로토콜 성능 실험 결과 및 분석

본 장에서는 위에서 구축한 테스트베드를 이용하여

표 2. 지연 시간에 따른 가용대역폭
Table 2. Available bandwidth for various delay condition.

(단위 : Mbps)

delay	reno	hstcp	stcp	bic	cubic
10ms	590.03	699.93	654.79	691.17	677.25
50ms	152.37	460.93	519.76	490.94	393.59
100ms	88.96	330.66	499.56	478.37	268.90
150ms	52.30	224.93	419.94	384.88	253.35
200ms	54.22	167.16	328.00	273.62	206.73

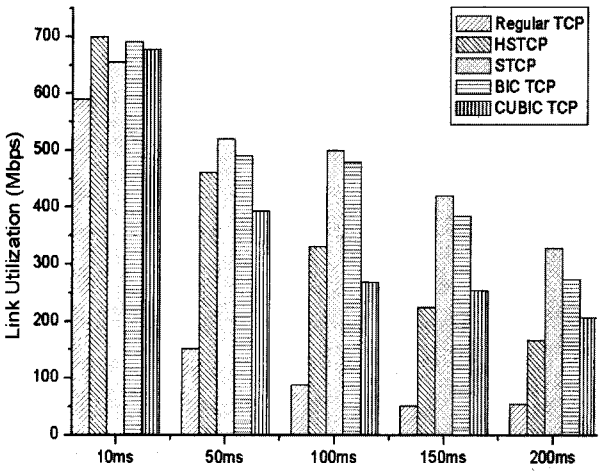


그림 7. 지연 시간에 따른 가용대역폭
Fig. 7. Available bandwidth for various delay condition.

고대역 전송 프로토콜들의 가용대역폭 활용도, 고대역 프로토콜 간의 공정성, 기존 TCP와의 공정성, RTT 공정성 등의 성능을 측정한 실험 결과를 보이고, 이를 분석한다.

첫 번째 실험은 대역폭과 지연의 곱이 큰 네트워크 환경에서 고대역 전송 프로토콜의 가용 대역폭 활용도를 보기 위해 단일 플로우를 전송하면서 지연 시간에 따른 패킷 전송 성능을 측정하였다. PC 라우터에 설치한 Dummynet 을 이용해 지연 시간을 10ms, 50ms, 100ms, 150ms, 200ms까지 변경하면서 각 프로토콜의 패킷 전송 성능을 측정하였으며, 이 때, 패킷 손실 발생률은 10^{-6}으로 설정하였다. 그 테스트 결과를 아래의 [표 2]와 <그림 7>의 막대그래프로 나타내었다.

위의 그래프에서 보는 바와 같이 대역폭과 지연의 곱이 (Bandwidth Delay Product, BDP) 작은 네트워크에서는 기존 TCP 플로우나 고대역 전송 프로토콜 플로우나 비슷한 가용 대역폭을 보였으나, 대역폭과 지연의 곱이 큰 고대역 네트워크 환경에서 기존의 TCP는 가용 대역폭을 충분히 활용하지 못하고 있음을 확인할 수 있었다.

두 번째 실험에서는 RTT 공정성을 평가하기 위하여

표 3. RTT 비율에 따른 전송 성능 비교
Table 3. Comparison of transmission performance for different RTT ratios.

Inverse RTT ratio	1	2	3	6
Regular TCP	1.04	1.71	2.74	4.86
HSTCP	0.98	1.11	2.37	6.28
STCP	1.01	1.40	3.58	13.95
BIC TCP	0.99	1.26	2.47	9.59
CUBIC TCP	0.88	1.16	1.88	2.07

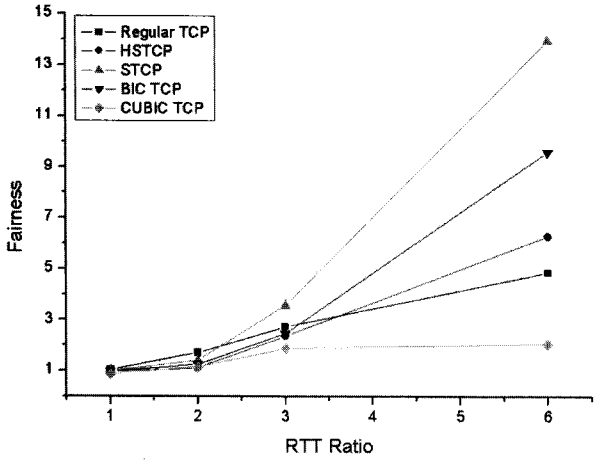


그림 8. RTT 비율에 따른 전송 성능 비교
Fig. 8. Comparison of transmission performance for different RTT ratios.

동일한 프로토콜을 사용하지만, RTT 가 다른 2개의 플로우를 동시에 전송하면서 각 전송 프로토콜들의 패킷 전송 성능에 따른 RTT 공정성을 비교하였다.

첫 번째 플로우의 지연 시간을 10ms로 고정한 상태에서 두 번째 플로우의 지연은 10ms(1배), 20ms (2배), 3ms(3배), 60ms(6배)로 변경하면서 패킷 전송 성능을 측정하였다. 아래 [표 3]과 <그림 8>은 RTT 비율에 따른 각 프로토콜의 전송 성능 차이를 비로 나타낸 것으로 측정 결과, 대부분의 고대역 전송 프로토콜이 RTT 공정성 평가에서 기존의 TCP 보다 좋지 않은 결과를 보였다.

특히 STCP 의 경우에는 RTT의 비율이 6배로 커짐에 따라 패킷 전송 성능의 차이는 10배 이상을 보였으며, HSTCP, BIC TCP 역시 기존의 TCP 보다 RTT 비율에 따른 패킷 성능의 차이가 더 컸다. 그러나 CUBIC TCP 의 경우에는 RTT fairness 개선에 중점을 둔 알고리즘인 만큼 기존의 TCP 보다 훨씬 더 개선된 성능을 보였다.

다음으로 기존 TCP 프로토콜과 고대역 전송 프로토콜 사이의 공정성을 보기 위해 한 플로우는 기존의

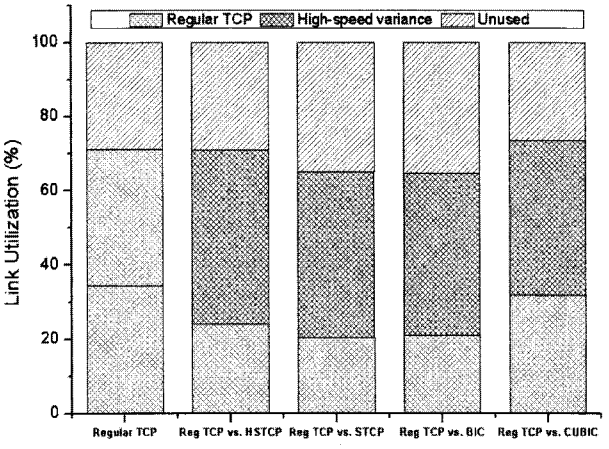


그림 9. 기존 TCP와의 공정성 (delay=10ms)
Fig. 9. Fairness with existing TCP (dealy=10ms).

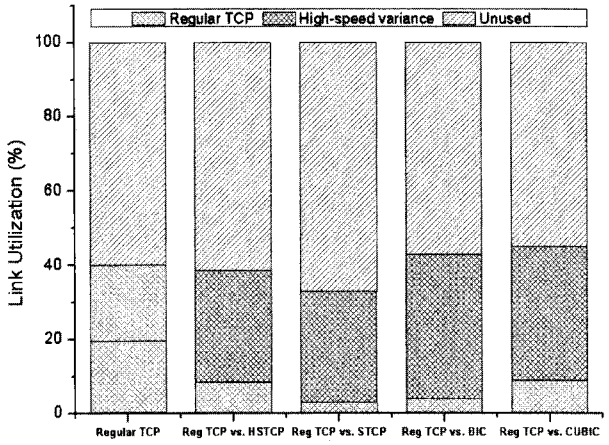


그림 10. 기존 TCP 와의 공정성 (delay=100ms)
Fig. 10. Fairness with existing TCP (dealy=100ms).

TCP 프로토콜을 사용하여 전송하고, 또 다른 하나는 고대역 전송 프로토콜을 사용하여 전송함으로써 두 플로우 사이의 공정성을 측정하였다. BDP가 큰 고대역 네트워크와 기존의 협대역 네트워크에서의 고대역 전송 프로토콜 플로우와 기존의 TCP 플로우가 경쟁했을 때 어떻게 동작하는지를 비교하기 위해 네트워크 환경을 지연시간이 10ms일 때와 100ms일 때로 나누어 테스트하였다. 이때의 패킷 손실률은 10^{-6}으로 설정하였다.

<그림 9>와 <그림 10>은 각각 지연 시간이 10ms일 때와 100ms 일 때의 결과 그래프이다. 위 막대그래프에서도 확인할 수 있듯이 기존의 작은 BDP 네트워크에서는 TCP 플로우가 고대역 전송 프로토콜 플로우와 경쟁하더라도 자신의 가용 대역폭을 비교적 그대로 잘 유지함을 보였다. 그러나 지연 시간을 100ms로 설정한 고대역 네트워크 환경에서는 두 개의 플로우 모두 기존 TCP 프로토콜을 사용하여 전송하였을 경우에는 전체

가용 대역폭을 공평하게 반반씩 나누어 씀을 볼 수가 있었으나, 기존의 TCP 프로토콜과 고대역 전송 프로토콜을 각각 사용하여 두 개의 플로우를 전송하였을 경우, 고대역 TCP 플로우가 훨씬 더 많은 대역폭을 차지함을 볼 수 있는데 이것은 기존 TCP의 대역폭을 삭감하면서 얻은 결과여서 새로 제안된 고대역 TCP 프로토콜들은 기존 TCP와의 공정성에 문제점을 나타내는 것을 알 수 있다.

V. TEIN2를 이용한 한국-중국-독일 3개국간 고대역 TCP 실험

본 장에서는 TEIN2 국제 연구망을 통하여 한국과 중국, 독일 3개국 간에 고대역 TCP를 실험한 내용을 기술한다.

기본적인 토폴로지는 <그림 11>과 같고, 한국의 KOREN 공용시험환경노드와 중국 칭화대학의 Xing Li 교수 랩, 그리고 독일 Karlsruhe 대학의 Roland Bless 교수 랩의 호스트 간에 실험을 수행하였다.

TEIN2는 2000년 우리나라 정상이 유럽 4개국 순방 시 제안하여 제 3차 ASEM 정상회의에서 공식 프로젝트로 승인을 받은 아시아와 유럽을 연결하는 지역 간(Inter-Regional) 연구망이다. 2001년 12월 한국 KOREN 과 프랑스 RENATER를 2Mbps로 연결함으로써 탄생한 TEIN2는 2005년 10월에는 155Mbps로 운영하였으나, 현재는 아시아·유럽 간의 회선을 622Mbps로 증설하였고, 중국, 베트남, 태국 등과 연결하였다. 따라

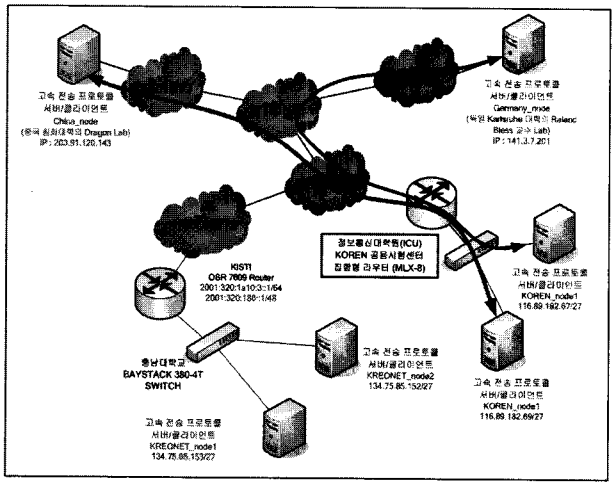


그림 11. 한국-중국-독일 3개국 간 네트워크 구성도
Fig. 11. Network configuration between three nations (KOREA-CHINA-GERMANY).

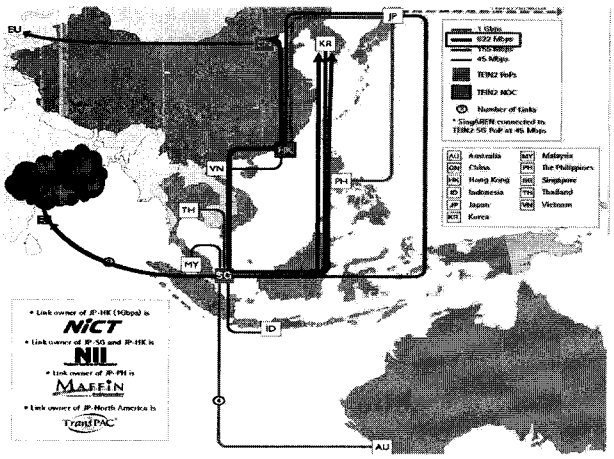


그림 12. 한국-중국-독일 3개국 간 트래픽 이동 경로
Fig. 12. Traffic transmission routes between three nations (KOREA, CHINA, GERMANY).

서 한국-중국-독일 3개국간의 고대역 TCP 실험을 동시에 진행해야 하는 본 논문에서는 아시아와 유럽을 잇는 초고속 정보통신망인 TEIN2 링크를 활용하였다.

간단한 ping 테스트 결과, 한국과 중국 간의 왕복 지연 시간은 평균 약 140ms, 한국과 독일 간의 왕복 지연 시간은 평균 약 270ms로 두 경로의 RTT 값은 약 2배 정도의 차이를 보였다. 따라서 왕복 지연 시간, 즉 RTT 값이 서로 다른 중국과 독일에서 동일한 프로토콜을 사용하는 2개의 플로우를 동시에 전송하면서 각 전송 프로토콜의 패킷 전송 성능에 따른 RTT 공정성을 비교하였다.

<그림 12>는 TEIN2를 이용하여 한국-중국-독일 3개국 간 고대역 TCP 실험 수행 시, 실제 트래픽의 이동 경로를 보이고 있다.

<그림 12>에서 보는 바와 같이 한국에서 싱가폴까지의 TEIN2 링크는 622Mbps의 대역폭을 제공하고 있으며, 이 구간에서의 병목현상으로 많은 혼잡이 발생할 것으로 예상하였다.

많은 트래픽이 송·수신 되고 있는 실제 네트워크 환경이므로 낮 시간에는 백그라운드 트래픽의 영향을 많이 받을 것 같아 주로 테스트는 밤이나 새벽 시간을 이용하였고, 각 고속 전송 프로토콜마다 총 5번씩의 테스트를 하여 그 평균값을 보였다.

<그림 13>은 테스트 수행 시간에 함께 모니터링한 TEIN2 링크의 트래픽 발생 모습이다. TEIN2 링크에 목적지가 한국인 incoming traffic이 발생하고 있음을 확인할 수 있었다.

<그림 14>의 막대그래프는 중국에서 한국으로, 독일

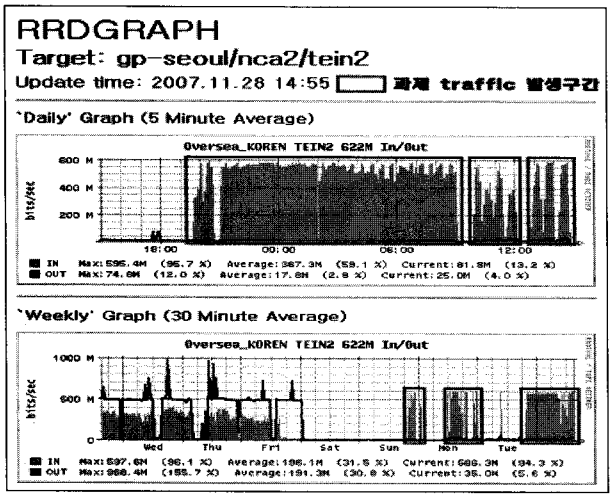


그림 13. TEIN2 발생 트래픽 모니터링
Fig. 13. Traffic monitoring on the TEIN2 networks.

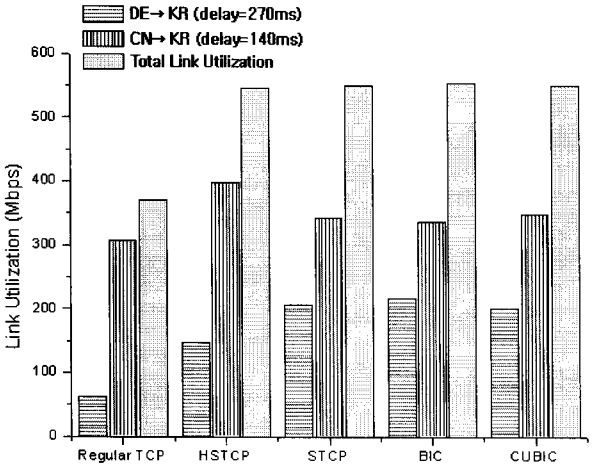


그림 14. 한국-중국-독일 3개국 간 가용대역폭 활용도 비교
Fig. 14. Comparison of Link Utilization between three nations (KOREA, CHINA, GERMANY).

에서 한국으로, 즉 동일한 방향으로 동시에 트래픽을 전송하며 RTT 비율에 따른 패킷 전송 성능을 측정한 그래프이다. TEIN2 링크는 기본적으로 622Mbps의 대역폭을 제공하고 있으며 한국과 중국 간의 왕복 지연 시간은 140ms로 대역폭과 지연의 곱(Bandwidth Delay Product, BDP)이 약 10Mbyte인 한·중 간 실제 네트워크 환경에서 기존의 TCP를 제외한 나머지 고속 전송 프로토콜들은 모두 80% 이상의 가용대역폭 활용도를 보였다. 또한, 한국과 독일 간의 왕복 지연 시간은 약 270ms로 대역폭과 지연의 곱(Bandwidth Delay Product, BDP)이 큰 한국과 독일 간 실제 네트워크 환경에서도 역시 고대역 TCP는 모두 50% 이상의 가용대역폭 활용도를 보였으나, 기존의 TCP는 TEIN2 링크가 제공하는 대역폭을 충분히 활용하지 못하고 있음을 확인하였다.

이는 로컬 테스트의 결과와 마찬가지로 대역폭과 지연의 곱이 큰 네트워크에서 TCP의 혼잡 윈도우를 기존과는 다른 방식으로 제어하는 고대역 전송 프로토콜이 역시 기존의 TCP 보다 훨씬 높은 가용 대역폭 활용도를 보임을 알 수 있다. 그러나 혼잡 회피 구간에서 혼잡 윈도우를 RTT 마다 패킷 1개의 비율로 선형 증가시키고, 혼잡 발생 시에는 이를 반으로 줄이는 기존의 TCP는 대역폭과 지연의 곱이 큰 네트워크 환경에서 실제 가용대역폭의 약 8%밖에 활용하지 못함을 보였다. 이것은 로컬 실험에서 Dummynet을 사용하여 강제로 200ms의 delay를 설정하고, 테스트 했을 때에도 역시 유사한 결과 값을 보였었다. 또한, 위 그래프에서는 두 플로우간 RTT 비율에 따라 패킷 전송 성능에도 차이가 있음을 볼 수 있었다.

그러나 이번 실험에서는 기존 연구 논문에 제시된 것과 비교하여 성능 차이의 폭이 크지 않았는데 이는 대부분의 고대역 전송프로토콜이 약 10Gbps까지의 대역폭을 고려하고 설계된 것이 많은 관계로 실제 최대 대역폭이 622Mbps로 제한된 TEIN2 국제 연구망에서는 기존의 연구 논문 결과에 준하는 고대역 TCP의 성능을 측정하는 데에는 어려움이 있었기 때문이다. 또한, 기존 논문의 시뮬레이션에서는 RTT의 비율이 3배 혹은 6배 이상 차이가 나는 네트워크 환경을 고려하였으나, 한국-중국 간 왕복 지연 시간(140ms)과 한국-독일 간 왕복 지연 시간(270ms)이 약 2배 정도밖에 차이가 나지 않는 네트워크 환경 또한 RTT 비율에 따른 패킷 전송 성능의 차이를 비교하기에 어려움이 있었다고 판단된다.

V. 결론 및 향후 연구 과제

기존에 사용하고 있는 TCP는 비교적 낮은 대역폭을 가지는 네트워크에 적합하게 설계되었기 때문에 대역폭과 지연의 곱이 큰 차세대 네트워크에는 가용대역폭을 제대로 활용하지 못하는 단점을 지니고 있다. 이를 해결하고자 다양한 고속 전송 계층 프로토콜이 제안되었는데, 본 논문에서는 TEIN2 국제연구망을 이용하여 이러한 고속 TCP 프로토콜의 성능을 측정하는 것을 목표로 하였다. 그 실험 결과, 고대역 전송 프로토콜들은 기존 TCP에 비해서 대역폭과 지연의 곱이 큰 네트워크일수록 기존 TCP에 비해 우수한 성능을 나타내는 경향을

보였다. 그리고 지금까지 제안된 고대역 전송 프로토콜은 모두 기존 TCP 프로토콜의 대역폭에 영향을 미치지 않는 범위 내에서 더 높은 성능을 갖는다고 기존 TCP와의 공정성을 주장하고 있으나, 이번 실험 결과에 따르면 제안된 고대역 프로토콜들이 실제로는 TCP의 가용대역폭을 삭감하면서 자신의 전송률을 올리는 것을 확인할 수 있었다. 아무리 고대역 전송계층 프로토콜이 높은 성능을 달성하더라도 그것이 기존 TCP 성능을 감소시킨 댓가라면 이는 적합한 프로토콜로 인정받을 수 없을 것이다. 또한, 이것은 실제 인터넷을 통한 활용 시에 큰 문제점이 될 것이므로 따라서 향후 연구 계획으로는 이에 대한 성능 개선 방안에 관한 연구를 수행하고자 한다.

참 고 문 헌

[1] S. Floyd, "HighSpeed TCP for large congestion windows", RFC3649, Dec.2003
[2] Tom Kelly, "Scalable TCP: Improving performance in highspeed wide area networks", http://www-lce.eng.cam.ac.uk/~ctk21/scalable/, December 2002.
[3] Cheng Jin, David X. Wei, Steven H. Low, "Fast TCP: Motivation, Architecture, Algorithm, Performance", Proceedings of IEEE INFOCOM, March 2004
[4] 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.
[5] Injong Rhee and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", PFLDnet 2005.
[6] Dummynet, "http://www.dummynet.com/"
[7] Web100, http://www.web100.org/

저 자 소 개

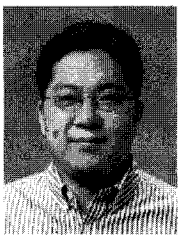
이 기 라(학생회원)
2005년 배재대학교 정보통신 공학부 학사
2008년 충남대학교 전자전파정보 통신공학과 석사
2008년~현재 LG전자 MC연구소 연구원
<주관심분야 : 이동통신 네트워크, 초고속 통신, 데이터 통신, 센서네트워크>

소 상 호(학생회원)
2007년 충남대학교 전자전파 정보통신공학과 학사
2007년~현재 충남대학교 전자전파정보통신공학과 석사
<주관심분야 : 이동통신 네트워크, 데이터 통신, 센서 네트워크>

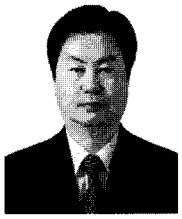
최 윤 철(학생회원)
2007년 충남대학교 전자전파 정보통신공학과 학사
2007년~현재 충남대학교 전자전파정보통신공학과 석사과정
<주관심분야 : 컴퓨터 네트워크, 데이터 통신, NetFPGA>

박 만 규(학생회원)
1999년 공주대학교 물리학과 학사
2001년 공주대학교 전기전자정보 통신공학과 석사
2001년~2004년 시스웰 기업 부설 통신기술 연구소 전임연구원
2004년~2006년 케이디넷 통신사업본부 기술팀 전임연구원/대리
2007년~현재 충남대학교 전자전파정보통신 공학과 박사과정
<주관심분야 : 이동통신 네트워크, 데이터 통신 센서네트워크>

이 재 용(평생회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1995년 한국과학기술원 전기 및 전자공학과 박사
1990년~1995년 디지콤 정보통신 연구소 선임연구원
1995년~현재 충남대학교 정보통신공학부 교수
<주관심분야 : 초고속통신, 인터넷, 네트워크 성능 분석>



김 병 철(평생회원)
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>



김 대 영(평생회원)-교신저자
1975년 서울대학교 전자공학과 학사
1977년 한국과학기술원 통신 공학 석사
1983년 한국과학기술원 통신 공학 박사
1983년~현재 충남대학교 정보통신공학부 교수
2009년 현재 미래인터넷 포럼(FIF) 부의장, APAN 부의장, ISO/IEC JTC1/SC6 의장