링크다운 환경에서 TCP에 대한 SCTP의 멀티호밍 효과

Multihoming Effect of SCTP Over TCP in the Link-down Environment

최용운, 이용진
한국교원대학교 기술교육과

Yong-Woon Choi(habangmo@yonsei.ac.kr), Yong-Jin Lee(lyj@knue.ac.kr)

요약

현재 인터넷에서 사용 중인 연결지향형 트랜스포트 계층 프로토콜은 TCP이다. TCP는 단일 경로 상에서 단일 스트림을 이용하여 통신하기 때문에, 경로상의 링크가 다운되는 경우 통신 불능이라는 결함을 지니고 있다. 새로운 트랜스포트 계층 프로토콜인 SCTP는 두 개 이상의 경로를 제공하는 멀티호밍(multi-homing) 특성을 갖고 있기 때문에 1차 경로 상의 링크가 다운되는 경우 TCP와 달리 대체 경로를 이용하여 통신이 가능하다. 따라서 본 연구에서는 링크다운 환경에서 NS-2 시뮬레이션을 이용하여 SCTP의 멀티호밍 효과를 측정하고 분석하였다. 특히 SCTP는 TCP처럼 싱글호밍으로도 사용될 수 있기 때문에 $SCTP_{single\text{-}homing}$과 $SCTP_{multi\text{-}homing}$으로 구분하였다. 시뮬레이션에서는, 링크다운 시간, 대역폭, 그리고 RTT(Round Trip Time)를 변화시키면서 TCP와 SCTP의 처리율과 대역폭 이용률을 측정하였다. 링크다운 시간에 따른 비교 결과, $SCTP_{multi\text{-}homing}$의 처리율이 TCP의 처리율 보다 평균적으로 18 % 우수한 것으로 확인되었다. RTT와 대역폭에 따른 결과는 $SCTP_{multi\text{-}homing}$의 처리율이 TCP의 처리율 보다 각각 27 %와 9 % 우수한 것으로 판명되었다. $SCTP_{single\text{-}homing}$과 TCP 사이에는 별다른 성능 차이가 없었다. 종합적으로는 링크다운 환경에서 $SCTP_{multi\text{-}homing}$의 성능이 TCP보다 평균 18% 우수하였다. 이 결과는 실제 인터넷 경로 상에서 링크 다운이 발생되는 경우 TCP에 대한 SCTP의 멀티호밍 효과를 추정하기 위한 벤치마크로 사용될 수 있다.

■ 중심어 : | 링크다운 | SCTP | TCP | 멀티호밍 |

Abstract

TCP(Transmission Control Protocol) is currently used connection-oriented protocol as a typical transport layer protocol in the Internet. However, it has deficiency not be able to communicate with other TCP entities when any link included in the path is down because of single-homing on single path. SCTP(Stream Control Transmission Protocol) suggested as the new transport layer protocol supports multi-homing feature, which provides several paths between source and destination. It can communicate with other SCTP entities using alternate path even when any link on the primary path is down. This paper aims to measure and analyze the multi-homing effect of SCTP over TCP in case of link-down using NS-2 simulator. We classify SCTP into $SCTP_{single\text{-}homing}$ and $SCTP_{multi\text{-}homing}$ because SCTP with single-homing can also be used like TCP. We measured throughput and bandwidth utilization varying link-down duration, bandwidth, and RTT(round trip time). Simulation results show that throughput of $SCTP_{multi\text{-}homing}$ is more than that of TCP by 18 % on average. It is also shown that $SCTP_{multi\text{-}homing}$ on varying RTT and bandwidth increases the throughput of TCP by 27% and 9% on average, respectively in the link-down environment. In above cases, more or less difference between $SCTP_{single\text{-}homing}$ and TCP on throughput and bandwidth utilization was found. To summarize, multi-homing effect of SCTP over TCP on throughput is about 18 % on average in the link-down environment This experimental result can be used as the benchmark in order to estimate the multi-homing effect of SCTP over TCP when the link-down happens in the real Internet.

■ keyword : | Link-down | SCTP | TCP | Multi-homing |

접수번호 : #090527-002
접수일자 : 2009년 05월 27일
심사완료일 : 2009년 06월 03일
교신저자 : 이용진, e-mail : lyj@knue.ac.kr

I. 서 론

현재 대부분의 인터넷 응용 프로그램은 통신 프로토콜로 응용계층에서 HTTP를 사용하고 트랜스포트 계층에서 TCP를 사용한다. TCP는 싱글호밍(single-homing)을 이용하는 연결지향형(connection-oriented) 프로토콜로 송신지와 수신지 사이를 단일경로로 연결하고 있기 때문에 경로 상에 링크다운이 발생하는 경우 통신 불능 상태가 된다. IETF는 2000년에 SCTP(Stream Control Transmission Protocol)를 TCP(Transmission Contron Protocol), UDP(User Datagram Protocol)에 이은 제3의 트랜스포트 계층 프로토콜 표준으로 제정하고, 이를 RFC 2960으로 공표하였다. 이어 2006년에 RFC 2960은 RFC 4460으로 발전하였다. SCTP는 윈도우 기반의 혼잡제어, 에러 탐지, 그리고 재전송 등의 TCP 혼잡 제어 메커니즘을 상당부분 흡수하였다[9]. 그러나 SCTP는 TCP에 없는 DOS 공격에 대한 견고성, TCP의 헤드-오브-라인 블러킹(head-of-line blocking)을 감소시키기 위한 멀티 스트리밍(multi-streaming), 그리고 양 종단 호스트 사이에 두 개 이상의 IP 주소를 할당하여 얻어지는 멀티호밍(multi-homing)등의 기능을 제공한다. 또한 초기 슬로우 스타트 구간에서 SCTP의 초기 윈도우는 2xMTU(Maximum Transfer Unit)인데 비해[9], TCP의 초기 윈도우는 1xMSS(Maximum Segment Size)로 정의되어 있었으나, 개정된 TCP 명세는 초기 윈도우의 크기를 임의의 세그먼트까지 증가할 수 있도록 되어 있다.

본 연구에서는 SCTP의 멀티호밍 특성에 주목하여 단일 경로를 갖는 TCP에 비해 대체 경로를 갖는 SCTP의 성능이 링크다운 시간에 따라 얼마나 향상되는지를 연구하고자 한다.

본 연구와 관련하여 기존의 TCP와 SCTP의 성능을 평가하는 연구를 살펴보면 다음과 같다. 송정화, 이미정, 고석주(2003)는 NS-2를 사용하여 SCTP의 멀티호밍의 특성 및 재전송 정책에 대한 성능 평가를 연구한 바가 있다. 이 연구에서는 TCP와 비교하기보다는 SCTP의 대체 경로에 대한 패킷 손실률을 가지고 대역폭과 RTT를 조정하면서 실험을 하였으나, 링크다운 환경은 고려되지 않았다. 김주회(2004)는 실제 리눅스 환경에서 두 대의 단말기와 두 대의 라우터를 가지고 TCP를 이용하여 SCTP를 시뮬레이션한 바 있으나, 실제 SCTP를 사용하지 않은 단점을 가지고 있으며 링크다운 환경 역시 고려되지 않았다. 하종식, 고석주(2005)도 리눅스 환경에서 두 대의 단말기로 테스트베드를 구성하여 전송시간과 데이터 양, 총 수신 패킷수를 연구한 바가 있으나, 링크다운 환경에서의 멀티호밍 효과를 측정하지는 않았다. 박재성, 고석주(2008)의 연구 역시 리눅스 환경에서 전송지연, 패킷 손실률 그리고 SCTP의 스트림 개수를 가지고 성능을 비교 하였지만 링크다운 환경을 고려하지 않았다. 따라서 본 연구에서는 NS-2 시뮬레이터(Network Simulator 2)를 사용하여 링크다운 환경에서의 SCTP의 멀티호밍 효과를 TCP와 비교하고 분석한다. 측정 대상은 처리율(throughput)과 대역폭 이용률(bandwidth utilization)이며, SCTP는 멀티호밍을 갖는 $SCTP_{_multi\text{-}homing}$과 싱글호밍을 갖는 $SCTP_{_single\text{-}homing}$의 두 가지로 나누어 TCP와 비교한다.

전체 5장으로 구성되는 본 연구는 2장에서 링크다운 환경에서의 TCP와 SCTP의 특성과 동작을, 3장에서 시뮬레이션 모델을 통한 연구방법을, 4장에서 시뮬레이션 결과 및 해석을, 마지막으로 5장에서 본 연구의 요약과 추후 연구 과제에 대해 논의한다.

II. 링크다운 환경에서의 TCP와 SCTP

TCP는 인터넷에서 가장 널리 사용되는 연결 지향적이고 신뢰성 있는 전송을 수행하는 트랜스포트 계층 프로토콜로 전이중 방식으로 통신하고 두 종단 간에 데이터 순서가 지켜지는 순차적 스트림 서비스를 제공한다. 현재의 인터넷에는 여러 가지의 원인으로 인해 오류가 발생될 수 있고, 이때 전송되는 데이터는 손상되거나 손실될 수 있기 때문에 TCP는 데이터 전송의 신뢰성을 보장하기 위해서 두 종단 간에 데이터가 정확히 수신되었는지를 확인하여 이상 유무를 판단하고 에러나 혼잡

발생시 적절한 제어를 수행한다.

손실 또는 손상된 패킷을 처리하기 위해 TCP는 재전송 방식을 사용한다. 이때 재전송 타이머를 이용하게 되는데 타이머의 값은 보통 RTT(Round Trip Time)의 2 배로 설정하게 된다. 송신측은 전송된 데이터에 대해서 RTO(Retransmission Time Out)동안 ACK를 받지 못하거나 3개의 중복된 ACK를 받게 되면 해당되는 데이터를 재전송하게 된다. TCP는 혼잡 윈도우를 이용한 슬로우-스타트(slow-start), 혼잡 회피(congestion avoidance), 신속한 복구(fast recovery)와 신속한 재전송(fast retransmission)을 사용하여 혼잡 제어를 수행한다.

그러나 TCP는 단일 경로를 사용하기 때문에 링크다운이 발생하면 대체 경로(alternate path)를 찾을 수 없다. 따라서 계속해서 재전송 타이머 값을 증가시키고 혼잡 윈도우를 1로 하여 재전송을 시도하지만 응답을 받을 수 없기 때문에 나중에는 정상으로 경로가 돌아올 때까지 기다리게 된다.

[그림 1]은 TCP가 링크다운 후 대체 경로가 없어서 송수신이 되지 못하고 있는 모습을 나타낸다.

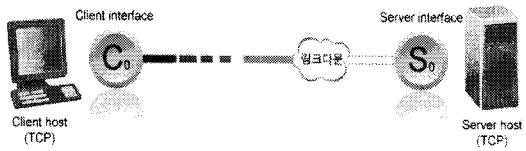


그림 1. 링크다운시 TCP의 동작

SCTP는 UDP의 메시지 지향(message-oriented) 특성과 TCP의 연결 지향 및 신뢰적 전송 특성을 모두 포함하도록 만들어 졌다. SCTP는 TCP와 달리 4-way 연결 설정 및 3-way 연결 종료 등의 기능을 제공하고, 멀티 스트리밍과 멀티호밍 특성을 제공한다. SCTP의 멀티스트리밍 특성을 이용하면 하나의 세션에서 다양한 종류의 콘텐츠를 식별하고 전달할 수 있으며, 바이트기반 전송방식인 TCP의 헤드-오브-라인 블러킹 문제점을 해결할 수 있다. SCTP의 멀티호밍 특성을 이용하면 단말기는 두 개 이상의 IP 주소를 한 세션에서 사용할 수 있으며, 이를 통해 경로 장애에 대한 복구기능을 제공할 수 있다. 두 개 이상의 이더넷 카드에 각각 할당한 IP 주소들을 SCTP 어소시에션(association)을 설정할 때 서로 교환하면, 각 IP에 해당되는 데이터 경로를 확보할 수 있게 된다. 연결이 성립되면, SCTP는 데이터 전송을 위해 여러 개의 경로 중 하나를 1차 경로(primary path)로 선택하고 다른 경로들은 유휴(idle) 상태로 되게 한다. 연결 설정 후 SCTP는 1차 경로를 이용하여 데이터를 전송한다. 유휴 상태에 있는 경로에 대해서는 heart beat chunk를 이용하여 경로가 이용가능한지를 주기적으로 확인한다. 전송 도중에 1차 경로에 있는 링크가 다운되는 경우 SCTP는 유휴 상태에 있는 경로 중에서 데이터 전송이 가능한 경로를 하나 선택해서 1차 경로로 설정한 후 데이터를 재전송하게 된다. [그림 2]는 SCTP의 멀티 호밍에 대한 특징을 보여준다.

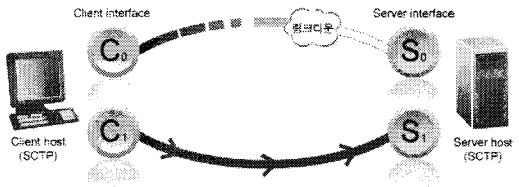


그림 2. 링크다운시 SCTP의 동작

한편 SCTP는 TCP처럼 단일 경로를 갖는 싱글호밍으로도 사용가능하기 때문에 본 연구에서는 SCTP를 단일 경로를 가지는 $SCTP_{_single\text{-}homing}$과 두 개 이상의 경로를 가지는 $SCTP_{_multi\text{-}homing}$으로 나누어서 TCP와 비교 분석하고자 한다.

Ⅲ. 시뮬레이션 모델 및 실험환경

본 연구에서 사용한 시스템의 사양은 [표 1]과 같다. 운영체제는 윈도우 상에서 구동되는 cygwin 버전 1.3.22가 사용되었으며, NS-2 시뮬레이터는 Ns-allinone-2.29가 사용되었다. 실험 계획은 먼저 링크가 다운되지 않았을 때의 SCTP와 TCP의 처리율과 대역폭 이용률을 비교 분석한 후에 SCTP와 TCP의 처리율이 동일하게 나오는 시뮬레이션 시간을 선정하였다.

표 1. 시스템 사양

구분	시 스 템 사 양
하드웨어	DualCore Intel Core 2 Duo E4500, 2200Mhz RAM 2G HDD 250G
운영체제	cygwin 1.3.22
NS 버전	Ns-allinone-2.29
I P	210.93.110.211

링크다운 간격은 5초, 10초, 20초로 변경해 가면서 SCTP와 TCP의 처리율과 대역폭 이용률을 비교분석한다. 대역폭은 0.1 Mbps, 0.5 Mbps, 1 Mbps, 5 Mbps, 10 Mbps로 변화시키고 RTT(Round Trip Time)는 10 ms, 50 ms, 100 ms, 200 ms, 500 ms, 700 ms, 1000 ms로 변화시키면서 처리율과 대역폭 이용률을 분석한다. 또한 링크다운 상황에서 TCP와 SCTP의 CWND(congestion window)가 어떻게 변화하는지 관찰하고 이 변화가 처리율에 어떻게 영향을 주는지 살펴본다.

한편 동일한 환경과 동일한 조건변수를 가지고 실험을 해야 하기 때문에 SCTP와 TCP의 초기 변수를 맞추어 주었다. SCTP는 RFC 2960 문서에서 데이터 청크(chunk) 1468 바이트와 SCTP 일반 헤더 12 바이트와 IP 헤더 20 바이트를 포함하여 1500 바이트로 정의되었다. 데이터 청크 1468 바이트는 다시 헤더 16 바이트와 데이터 1452 바이트로 나뉘므로 실제 데이터의 크기는 1452 바이트가 된다[9]. 따라서 TCP도 데이터 1460 바이트와 TCP 헤더 20 바이트와 IP 헤더 20 바이트를 포함하여 1500 바이트로 동일하게 설정하고 실험을 진행하였다.

표 2. 사용 변수

구분	크 기
Packet size	1500 byte
Data size	TCP(1460 byte), SCTP(1452 byte)
MTU	1500 byte
ssthresh	TCP(65700 byte), SCTP(65340 byte)

전송 선로간의 최대 전송 단위인 MTU(Maximum Transfer Unit)는 1500 바이트로 설정하고 ssthresh(slow-start threshold)는 TCP는 65,700 바이트(45 * 1460), SCTP는 65,340 바이트(45 * 1452)로 설정하였다. 본 연구에 사용된 응용 계층 프로토콜은 FTP이고, [표 2]는 실험에 사용된 변수를 나타낸다.

실험의 측정 요소인 처리율과 대역폭 이용률은 다음과 같이 정의된다. 일반적으로 데이터 크기를 f, 데이터 전송시간을 Δ라 정의하면 처리율 r은 식 (1)과 같다[8].

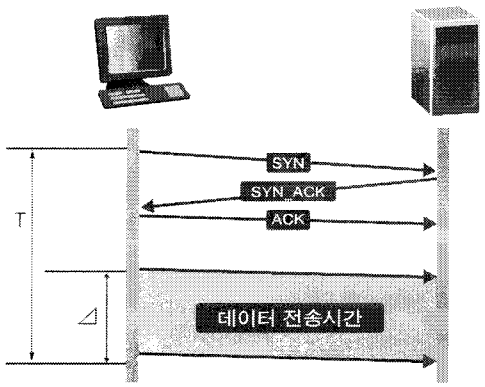
$$r = \frac{f}{\Delta} \tag{1}$$

식 (1)에서 데이터 전송시간(Δ)은 [그림 3]에서 Δ로 표시된 부분이고, 시뮬레이션 시간은 T로 표시되었다. Δ는 T에서 TCP와 SCTP의 연결 설정과 연결 해지를 수행하는 3-way 연결 설정 시간([그림 3]의 (a)) 및 4-way 연결 설정 시간([그림 3]의 (b))을 제외한 순수 데이터 전송 시간이다. 여기서 f는 Δ동안에 전송된 데이터의 크기이다.

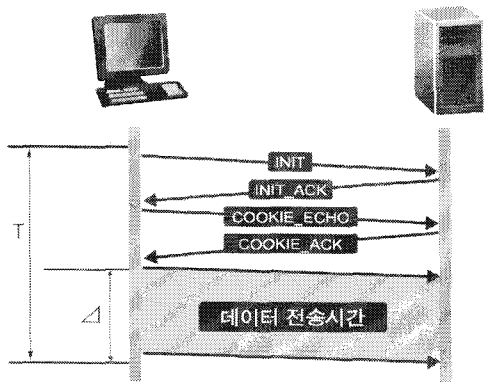
또한 대역폭 이용률(ρ_u)은 처리율(r)을 링크의 대역폭(B)로 나누면 되므로 식 (2)와 같다[8].

$$\rho_u = \frac{r}{B} \tag{2}$$

식 (1)과 식 (2)를 이용하여 전송 처리율 r과 대역폭 이용률 ρ_u를 측정하면 TCP, $SCTP_{\_single\text{-}homing}$ 그리고 $SCTP_{_multi\text{-}homing}$의 성능을 비교할 수 있다.



(a) TCP



(b) SCTP

그림 3. TCP와 SCTP의 시뮬레이션 시간과 데이터 전송시간

Ⅳ. 시뮬레이션 결과 및 해석

1. 링크다운이 없는 경우

먼저 링크가 다운되지 않았을 때의 처리율과 대역폭 이용률을 알아보기 위해 시뮬레이션 시간을 5 초, 100 초, 500 초로 하여 측정하였다. 대역폭과 RTT는 모든 시뮬레이션 시간에 동일하게 0.5 Mbps와 200 ms로 설정하였다. [그림 4]는 링크 다운이 없는 경우의 처리율을 보여준다.

[그림 4]에서 SCTP(single)은 single-homing을 사용한 SCTP를 SCTP(multi)는 multi-homing을 사용한 SCTP를 각각 나타낸다(이후에 나오는 그림과 표에서도 동일하다).

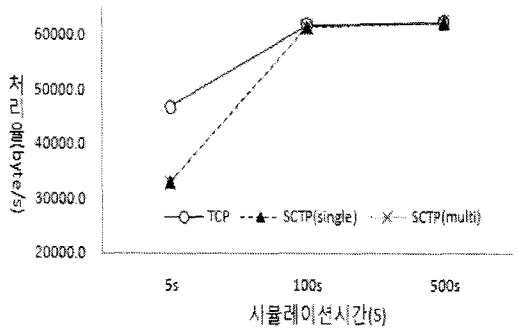


그림 4. 링크다운이 없는 경우의 처리율

시뮬레이션 시간을 5초로 설정한 경우, TCP의 처리율은 46,778 바이트, $SCTP_{single\text{-}homing}$의 처리율은 33,000 바이트, $SCTP_{multi\text{-}homing}$의 처리율은 33,000 바이트로 TCP가 단위 시간동안 가장 많은 양의 데이터를 전송하는 것으로 나타났다. 그 이유는 SCTP가 4-way-handshake로 연결 설정을 하는 데 비해, TCP는 3-way-handshake로 연결 설정을 하기 때문에 TCP의 초기 연결 설정 시간이 SCTP보다 짧기 때문이다. 즉, 시뮬레이션 5초 동안 TCP의 CWND 값은 약 67,000 바이트까지 올라가는데 반하여, SCTP는 약 54,000 바이트로 TCP보다 적게 올라가므로 이러한 혼잡 윈도우의 차이가 처리율의 차이로 나타난 것이다.

이제, 시뮬레이션 시간을 100 초로 증가시키면 TCP, $SCTP_{single\text{-}homing}$과 $SCTP_{multi\text{-}homing}$ 모두 약 61K 바이트로 유사한 결과 값이 나오는 것을 알 수 있다. 다시 시뮬레이션 시간을 500 초로 증가시키면 TCP, $SCTP_{single\text{-}homing}$과 $SCTP_{multi\text{-}homing}$는 모두 약 62K 바이트로 나오게 된다. 위와 같이 $SCTP_{single\text{-}homing}$과 $SCTP_{multi\text{-}homing}$이 같은 결과가 나오는 이유는 링크 다운이 없는 경우 $SCTP_{single\text{-}homing}$과 $SCTP_{multi\text{-}homing}$이 1차 경로를 통해서 경로 변경 없이 전송을 하기 때문이다.

[표 3]은 링크 다운이 없는 경우의 대역폭 이용률을 보여준다. 시뮬레이션 시간 100 초와 500 초에서 TCP, $SCTP_{single\text{-}homing}$ 그리고 $SCTP_{multi\text{-}homing}$의 대역폭 이용률이 처리율에서와 마찬가지로 동일한 결과가 나오고 있음을 확인할 수 있다.

표 3. 링크다운이 없는 경우의 대역폭 이용률

시뮬레이션 시간 / 프로토콜	5초	100초	500초
TCP	0.094	0.123	0.125
SCTP(single)	0.066	0.123	0.125
SCTP(multi)	0.066	0.123	0.125

따라서 시뮬레이션 시간이 짧은 경우(5 초)에는 링크가 다운되었을 때, SCTP와 TCP의 CWND가 다른 위치에 있기 때문에 공평한 비교를 할 수 없게 된다. 따라

서 본 연구에서는 동일한 처리율과 대역폭 이용률이 나오는 시뮬레이션 시간인 100 초를 시뮬레이션 시간(*T*)으로 설정한다.

2. 링크다운 시간에 따른 처리율과 대역폭 이용률

이 실험에서는 링크다운 시간을 5 초, 10 초, 20 초로 하고, 대역폭과 RTT는 0.5 Mbps와 200 ms로 설정하였다.

시뮬레이션 시간을 100초로 하여 5초 동안 링크를 다운시킨 경우 TCP와 $SCTP_{single-homing}$은 하나의 경로로 인해 링크다운 시간만큼 전송이 되지 못하고 송신 호스트와 수신 호스트 모두 유휴 상태와 대기 상태로 들어간다. 하지만 $SCTP_{multi-homing}$에서는 1차 경로의 링크가 다운됨에 따라 대체 경로를 찾아서 약 1.4 초 후에 재전송이 되었다. 따라서 TCP와 $SCTP_{single-homing}$의 처리율이 각각 54,186 바이트와 55,126 바이트인데 비해, $SCTP_{multi-homing}$의 처리율은 59,345 바이트로 많은 차이를 보인다.

100 초의 시뮬레이션 시간 동안에 10 초를 링크 다운하는 경우 TCP의 처리율은 48,574 바이트이고 $SCTP_{single-homing}$의 처리율은 49,812 바이트이지만 $SCTP_{multi-homing}$의 처리율은 59,345 바이트로 TCP와 $SCTP_{single-homing}$의 차이는 미미하지만, $SCTP_{multi-homing}$과 다른 프로토콜사이에는 현저히 차이가 나고 있음을 알 수 있다.

시뮬레이션 시간을 100 초로 하여 20 초를 링크다운하면 5 초와 10 초의 링크다운에 비하여 $SCTP_{multi-homing}$의 처리율이 월등하게 우수하게 나타난다. 즉 TCP의 처리율은 38,030 바이트이고, $SCTP_{single-homing}$의 처리율은 38,939 바이트인데 비해, $SCTP_{multi-homing}$의 처리율은 무려 59,345 바이트이다. 여기에서 $SCTP_{multi-homing}$은 10 초건 20 초건 링크 다운한 시간에 관계없이 동일한 처리율이 나오는 것에 유념해야 하겠다. 그 이유는 링크다운 시간의 크기에 관계없이 링크다운 초기에 대체 경로를 이용하여 전송이 이루어지기 때문이다.

링크다운 시간이 0 초, 5 초, 10 초, 20 초인 경우의 처리율이 [그림 5]에 나타나 있다.

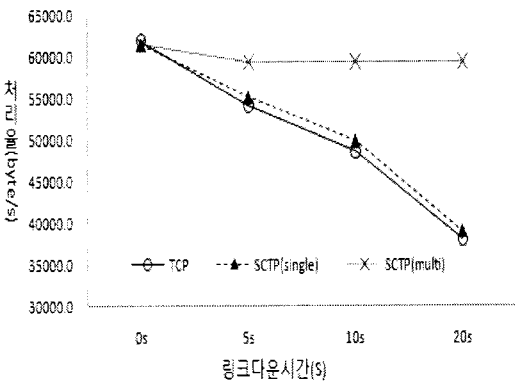
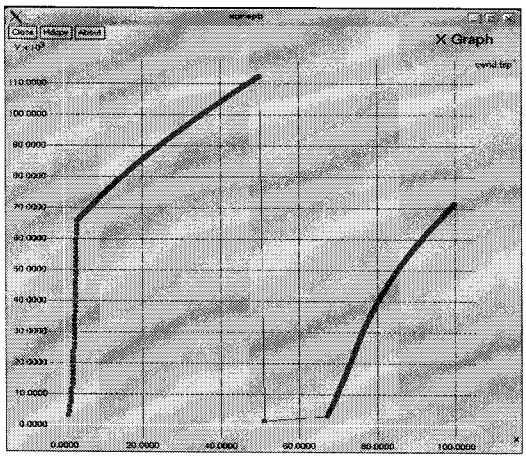


그림 5. 링크다운 시간에 따른 처리율

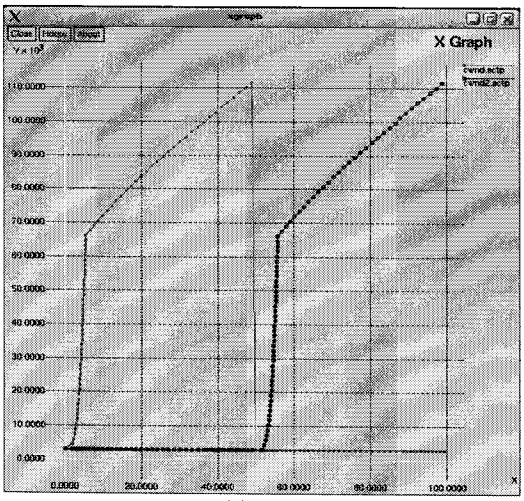
$SCTP_{multi-homing}$의 경우는 [그림 5]에서 보듯이 링크다운 시간에 거의 영향을 받지 않고 유사한 결과 값을 나타내고 있음을 알 수 있다. 하지만 TCP와 $SCTP_{single-homing}$의 경우에는 단일 경로로 송수신을 하다 보니 링크 다운된 시간만큼 처리율이 줄어들고 있음을 보여주고 있다. 링크다운이 없는 시뮬레이션 시간이 100 초인 경우의 TCP 처리율이 61,893 바이트인데 비해 $SCTP_{multi-homing}$의 처리율은 링크 다운된 시간이 5 초, 10 초, 20 초임에도 불구하고 약 59K 바이트로 TCP와 유사한 것을 확인할 수 있다. 따라서 링크다운 시간이 늘어날수록 $SCTP_{multi-homing}$의 처리율은 TCP와 $SCTP_{single-homing}$의 처리율에 비해 훨씬 더 우수하게 된다.

CWND의 변화를 보기 위해, 시뮬레이션 시간을 100 초로 하고, 50 초와 60 초 사이의 10 초간 링크를 다운시켰다. [그림 6] (a)는 TCP의 CWND의 변화를 보여주고, [그림 6] (b)는 $SCTP_{multi-homing}$의 CWND의 변화를 보여준다. ssthresh 값은 TCP의 경우 65,700(45x1460), SCTP의 경우 65340(45x1452)이다.

TCP의 경우 초기 윈도우가 RTT마다 지수적으로 증가하다가 ssthresh가 65,700 바이트가 되면 윈도우 크기를 지수적으로 증가하던 것을 멈추고 선형으로 증가하고 있다. 50초에서 60초까지 10초간 링크다운 상황이 발생되면 윈도우 값을 다시 초기 윈도우까지 내리고, 링크가 업(up)되면 다시 윈도우의 크기를 늘려가는 것을 알 수 있다.



(a) TCP



(b) SCTP

그림 6. TCP와 SCTP의 CWND 크기의 변화

그에 반해 SCTP_multi-homing의 경우는 [그림 6] (b)에서 보는 바와 같이 초기 윈도우에서 지수적으로 급격히 올라가다가 ssthresh가 65,340 바이트가 되면 윈도우의 크기를 지수적으로 늘리는 것에서 멈추고 다시 선형으로 증가하고 있음을 알 수 있다. 50초에서 60초까지 10초간 링크다운 상황이 발생하면서 1차 경로의 윈도우 크기가 초기 윈도우까지 떨어지게 되고 대체 경로의 윈도우가 1차 경로의 윈도우가 증가할 때와 같이 늘어나고 있음을 알 수 있다. 50초부터 60초사이의 링크가 다운되는 경우 NS-2 시뮬레이션에서 출력되는 tr (trace) 파일을 표시하면 [그림 7]과 같다.

```
r 49.99248 1 0 ack 40 ------- 0 1.0 0.0 1988 4013
+ 49.99248 0 1 tcp 1500 ------- 0 0.0 1.0 2033 4030
- 49.99984 0 1 tcp 1500 ------- 0 0.0 1.0 2006 3976
d 50 1 0 ack 40 ------- 0 1.0 0.0 1989 4015
v 50 link-down 1 0
d 50 0 1 tcp 1500 ------- 0 0.0 1.0 2007 3978
v 50 link-down 0 1
v 60 link-up 1 0
v 60 link-up 0 1
+ 67.04248 0 1 tcp 1500 ------- 0 0.0 1.0 1989 4034
- 67.04248 0 1 tcp 1500 ------- 0 0.0 1.0 1989 4034
r 67.26648 0 1 tcp 1500 ------- 0 0.0 1.0 1989 4034
```

(a) TCP

```
r 49.996448 4 1 sctp 48 -------S 0 4.0 1.0 1 1944 2962 65535 65535
+ 49.996448 1 4 sctp 1500 -------D 0 1.0 4.0 1 1987 2975 0 1986
+ 49.996448 1 4 sctp 1500 -------D 0 1.0 4.0 1 1988 2976 0 1987
d 50 4 1 sctp 48 -------S 0 4.0 1.0 1 1946 2965 65535 65535
v 50 link-down 4 1
d 50 1 4 sctp 1500 -------D 0 1.0 4.0 1 1962 2937 0 1961
v 50 link-down 1 4
+ 51.467468 2 5 sctp 1500 -------D 0 2.0 5.0 1 1945 2977 0 1944
- 51.467468 2 5 sctp 1500 -------D 0 2.0 5.0 1 1945 2977 0 1944
r 51.791468 2 5 sctp 1500 -------D 0 2.0 5.0 1 1945 2977 0 1944
```

(b) SCTP

그림 7. 링크다운시 TCP와 SCTP의 tr

[그림 7]의 r, +, -, d는 큐의 상태를 나타내는 것으로 +는 큐에 들어오는 패킷을, -는 큐에서 나가는 패킷을, d는 큐에서 드롭된 패킷을, r은 패킷이 수신노드에 도착했음을 나타낸다. [그림 7](a)의 첫 줄 r은 패킷이 들어왔음을 나타낸다. 다음으로 나타나는 숫자 49.99248은 시간을 표시하는 부분으로 패킷의 이동시간을 나타낸다. 1 0 은 from/to node를 나타내는 것으로 패킷의 이동 경로를 나타내며, ack는 패킷의 타입을 표시한다. 40은 패킷의 크기를 나타내고 뒤의 0은 fid를 나타내고 1.0 0.0 역시 from/to node를 나타내며, 1988은 네트워크 계층에서 이용하는 패킷의 순서번호를 의미한다. 4013은 패킷의 ID를 표시하는 필드이다.

[그림 7] (a)에서와 같이 TCP는 링크가 다운되는 50초부터 60 초까지 전혀 데이터를 전송하지 못하다가 링크가 업이 되고 7.0 초 이후에 데이터를 전송하기 시작하고 7.2 초가 되어서야 데이터가 수신측에 도착하게 된다. SCTP_multi-homing은 [그림 7] (b)에서와 같이 링크 다운이 발생하면 1차 경로를 대체 경로로 변경하고 1.4 초 후에 링크가 업이 되어 1.7 초 후에 데이터가 수신측에 도착하는 것을 알 수 있다. 이러한 차이가 TCP와

SCTP_multi-homing의 처리율 차이로 나타나고 있다.

SCTP_multi-homing이 재전송에 걸리는 시간이 1.4 초밖에 걸리지 않는 이유는 NS-2 시뮬레이션에서 출력되는 SCTP_multi-homing의 tr (trace) 파일을 나타낸 [그림 8]에서 찾아볼 수 있다.

```
r 64.33614 2 5 sctp 1500 -------D 0 2.0 5.0 1 2412 3652 0 2411
r 64.344908 5 2 sctp 48 -------S 0 5.0 2.0 1 2404 3692 65535 65535
r 64.346897 4 1 sctp 56 -------H 0 4.0 1.0 1 -1 3693 65535 65535
r 64.36014 2 5 sctp 1500 -------D 0 2.0 5.0 1 2413 3654 0 2412
r 64.38414 2 5 sctp 1500 -------D 0 2.0 5.0 1 2414 3655 0 2413
r 64.392908 5 2 sctp 48 -------S 0 5.0 2.0 1 2406 3696 65535 65535
                          .
                          .
                          .
                          .
r 65.17614 2 5 sctp 1500 -------D 0 2.0 5.0 1 2447 3706 0 2446
r 65.20014 2 5 sctp 1500 -------D 0 2.0 5.0 1 2448 3707 0 2447
r 65.201036 1 4 sctp 56 -------B 0 1.0 4.0 1 -1 3708 65535 65535
r 65.208908 5 2 sctp 48 -------S 0 5.0 2.0 1 2440 3748 65535 65535
r 65.225036 2 5 sctp 1500 -------D 0 2.0 5.0 1 2449 3710 0 2448
```

그림 8. Heartbeat 메시지 전송

[그림 8]에서 네모로 둘러싸인 부분은 링크가 업된 후에 수신측에서 송신측에게 Heartbeat request (H) 메시지를 보내서 링크다운 되었던 경로의 상태가 회복되었다는 메시지를 보내는 부분이다. 그 후에 송신측에서 회복되었다는 사실을 인지하였다는 Heartbeat Ack (B) 메시지를 보내고, 송신측에서는 다운되었던 경로가 회복되어 현재의 1차 경로가 다운되면 언제든지 대체 경로로 바꿀 수 있다고 인지하고 있다.

TCP와 SCTP_single-homing의 처리율이 같은 단일 경로를 사용함에도 불구하고 차이를 보이는 이유는 다음 [표 4]와 같은 이유이다. [표 4]는 링크가 다운되었을 때 큐에 들어있는 패킷수와 다시 링크-업 되는 시간, 재전송해서 목적지 노드에 들어오는 시간을 보여준다.

표 4. 링크다운시 링크의 상태

링크의 상태 / 프로토콜	큐(queue)에 들어있는 패킷수(s)	링크-업(link-up) 되는 시간 (s)	재전송시 목적지 노드에 들어오는 시간(s)
TCP	37	7.0	7.2
SCTP (single)	36	6.4	6.7
SCTP (multi)	36	1.4	1.7

링크가 다시 업이 되어 재전송하는데 걸리는 시간이 TCP의 경우에는 7.0 초인데 반하여 SCTP_single-homing의 경우에는 6.4 초로 짧다. 이로 인해 SCTP_single-homing의 처리율이 TCP 보다 약간 더 좋게 나타난 것이다. 그렇지만 이 결과는 NS-2 시뮬레이터 상에서 나타난 것으로 SCTP_multi-homing과 달리 TCP와 SCTP_single-homin의 프로토콜의 성능차이를 보여주는 것은 아니다.

표 5. 링크다운 시간에 따른 대역폭 이용률

링크다운시간 / 프로토콜	0s	5s	10s	20s
TCP	0.123	0.108	0.097	0.076
SCTP(single)	0.123	0.110	0.100	0.078
SCTP(multi)	0.123	0.119	0.119	0.119

[표 5]는 링크다운 시간에 따른 대역폭 이용률을 보여준다. 링크다운시간 0 초, 5 초, 10 초, 20 초에 따른 결과를 살펴보면, SCTP_multi-homing의 경우 링크다운 시간에 거의 영향을 받지 않고 약 0.12로 동일한 결과 값을 나타내고 있음을 알 수 있다. 하지만 TCP와 SCTP_single-homing의 경우에는 대역폭 이용률이 0.12에서 0.07로 떨어지고 있음을 보여주고 있다. TCP의 경우 링크다운 없이 시뮬레이션 시간 100초의 대역폭 이용률이 약 0.12이라면 SCTP_multi-homing의 경우 링크다운 시간이 5 초, 10 초, 20 초인 경우에도 약 0.12로 유사한 값이 나오는 것을 확인할 수 있다. 또한 링크다운 시간이 커질수록 SCTP_multi-homing과 TCP 및 SCTP_single-homing의 대역폭 이용률 차이가 점점 커짐을 예측할 수 있다.

링크다운 시간을 준 경우(5초, 10초, 20초)가 링크다운을 주지 않은 경우에 비하여 SCTP_multi-homing의 처리율이 TCP의 처리율 보다 평균 18% 우수한 것으로 판명되었다.

3. RTT에 따른 처리율과 대역폭 이용률

RTT에 따른 처리율과 대역폭 이용률의 변화를 살펴보기 위해 시뮬레이션 시간 100 초 동안 대역폭을 0.5 Mbps로 고정하고, RTT를 10 ms, 50 ms, 100 ms, 200 ms, 500 ms, 700 ms, 1000 ms로 변경하면서 실험하였

다. 링크를 다운하지 않은 경우와 동일 조건에서 10 초간 링크를 다운한 경우로 나누어 총 42 회에 걸쳐서 실험을 하였다. [그림 9]는 RTT에 따른 처리율을 나타낸다.

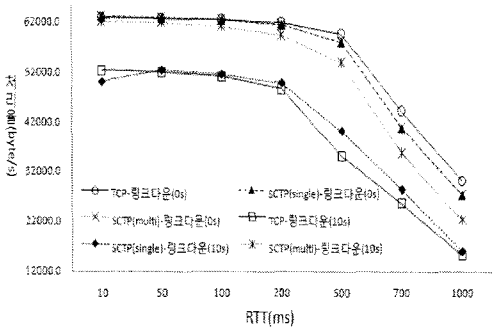


그림 9. RTT에 따른 처리율

[그림 9]에서 보는 바와 같이 가장 우수한 결과를 보여준 것은 링크를 다운하지 않은 TCP (TCP-링크다운(0s))로, RTT가 증가해도 처리율이 SCTP에 비해서 급격하게 감소하지는 않는 것으로 나타났다. 즉, 링크다운이 발생하지 않는다면, TCP가 가장 성능이 우수하다. 왜냐하면 500 ms이상으로 RTT가 길어지면 초기 전송량에서 TCP가 SCTP보다 많은 데이터를 전송을 하는데, RTT가 200 ms 일 때에는 TCP의 CWND가 ssthresh까지 가는 시간이 3.3초이고, SCTP의 CWND가 ssthresh까지 가는 시간이 5.3초로 2초의 차이가 나는데, 1000 ms로 RTT가 길어지면 TCP의 CWND가 ssthresh까지 가는 시간이 12.9초이고, SCTP의 CWND가 ssthresh까지 가는 시간이 21.06초로 8.16초의 차이가 난다. 이 차이가 TCP와 SCTP의 처리율의 차이를 나타내는 것이다.

그 다음으로 우수한 것은 SCTP(multi)-링크다운(0s)과 SCTP(single)-링크다운(0s)으로 링크가 다운되지 않고 손실이 없는 환경에서는 SCTP의 싱글호밍과 멀티호밍의 성능이 동일하다.

SCTP(multi)-링크다운(10s)은 TCP-링크다운(10s)과 SCTP(single)-링크다운(10s)보다 우수한 처리율을 보이고 있다.

RTT에 따른 TCP와 $SCTP_{\_single\text{-}homing}$ 그리고 $SCTP_{_multi\text{-}homing}$의 평균 처리율을 계산해보면 RTT가 짧을수록 $SCTP_{_multi\text{-}homing}$의 성능이 TCP보다 좋은 것으로 확인되었다.

[표 6]은 RTT에 따른 대역폭 이용률을 나타낸 것이다. 가장 대역폭 이용률이 우수한 것은 링크 다운이 없는 경우의 TCP-링크다운(0s)이다. 그 다음 우수한 것은 SCTP(multi)-링크다운(0s)과 SCTP(single)-링크다운(0s)으로 링크다운하지 않았기 때문에 동일한 대역폭 이용률을 보이고 있다. SCTP(multi)-링크다운(10s)의 경우는 링크 다운한 경우의 TCP와 $SCTP_{_single\text{-}homing}$보다 우수한 대역폭 이용률을 보이고 있다.

표 6. RTT에 따른 대역폭 이용률

RTT / 프로토콜	10ms	50ms	100ms	200ms	500ms	700ms	1000ms
TCP 링크다운(0s)	0.126	0.125	0.125	0.123	0.119	0.089	0.061
SCTP(single) 링크다운(0s)	0.126	0.126	0.125	0.123	0.116	0.082	0.055
SCTP(multi) 링크다운(0s)	0.126	0.126	0.125	0.123	0.116	0.082	0.055
TCP 링크다운(10s)	0.105	0.104	0.102	0.097	0.070	0.051	0.031
SCTP(single) 링크다운(10s)	0.100	0.105	0.103	0.100	0.080	0.057	0.032
SCTP(multi) 링크다운(10s)	0.124	0.124	0.122	0.119	0.108	0.072	0.045

이상의 결과를 종합하면, RTT 변화에 따른 $SCTP_{_multi\text{-}homing}$의 처리율과 대역폭 이용률은 TCP의 그것보다 평균 27% 우수한 것으로 측정되었다.

4. 대역폭에 따른 처리율과 대역폭 이용률

이 실험에서는 시뮬레이션 시간 100초 동안 RTT를 200 ms로 고정한 후 대역폭을 0.1 Mbps, 0.5 Mbps, 1 Mbps, 5 Mbps, 10 Mbps로 변경해 가면서 처리율과 대역폭 이용률을 측정하였다. RTT 실험에서와 마찬가지로 링크다운하지 않은 경우와 10초간 링크 다운한 경우에 대해 총 30회에 걸쳐 실험하였다. [그림 10]은 대역폭에 따른 처리율을 보여준다.

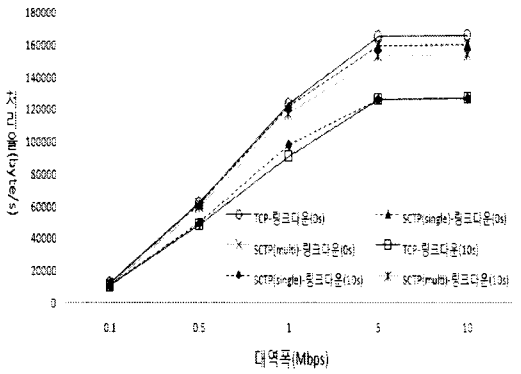


그림 10. 대역폭에 따른 처리율

[그림 10]에서 보듯이, 링크다운이 발생하지 않는다면 TCP의 성능이 가장 우수하다. 그러나 링크 다운이 있는 경우에는 SCTP(multi)-링크다운(10s)이 TCP-링크다운(10s)보다 우수한 처리율을 보이고 있다.

0.1 Mbps의 낮은 대역폭에서는 TCP와 SCTP의 처리율이 유사하지만, 대역폭이 커질수록 TCP-링크다운(0s)을 제외하고, $SCTP_{_multi\text{-}homing}$의 처리율은 늘어나고 있는데 비해, SCTP(single)-링크다운(10s)과 TCP-링크다운(10s)의 처리율은 $SCTP_{_multi\text{-}homing}$보다 많이 증가하지 않음을 알 수 있다.

[표 7]은 대역폭에 따른 이용률의 변화를 보여 준다.

표 7. 대역폭에 따른 이용률

대역폭 / 프로토콜	0.1Mbps	0.5Mbps	1Mbps	5Mbps	10Mbps
TCP 링크다운(0s)	0.125	0.123	0.123	0.033	0.017
SCTP(single) 링크다운(0s)	0.124	0.123	0.122	0.032	0.016
SCTP(multi) 링크다운(0s)	0.124	0.123	0.122	0.032	0.016
TCP 링크다운(10s)	0.106	0.097	0.091	0.025	0.013
SCTP(single) 링크다운(10s)	0.106	0.100	0.098	0.025	0.013
SCTP(multi) 링크다운(10s)	0.106	0.119	0.117	0.031	0.015

[표 7]에서 보듯이 대역폭이 커질수록 이용률은 떨어지고 있음을 알 수 있다. 다만, SCTP와 TCP 모두 0.1 Mbps에서 1 Mbps로 대역폭이 커질 때가 가장 큰 대역폭 이용률을 보여 준다. $SCTP_{_multi\text{-}homing}$의 경우 링크의 다운과 관계없이 대역폭 이용률이 유사하게 나오고 있고, 10 초 링크 다운한 경우에는 오히려 0.1 Mbps에서 0.5 Mbps로 대역폭이 커질 때 이용률이 좋아지다가 대역폭이 5 Mbps로 증가하면서부터는 대역폭 이용률이 많이 감소하고 있음을 알 수 있다. 왜냐하면 대역폭 이용률 식 (2)에서 B의 값이 대역폭에 따라 달라지기 때문에 [그림 10]과 같이 처리율은 선형적으로 늘어나는데 비하여 대역폭 이용률은 그렇지 않게 나타났다. 따라서 $SCTP_{_multi\text{-}homing}$은 대역폭 0.5 Mbps와 1 Mbps에서 대역폭의 이용 효율이 가장 좋은 것으로 나타났다. 그에 반하여 TCP-링크다운(10s)은 SCTP-링크다운(10s)에 비하여 대역폭이 커질수록 이용률이 떨어지고 있음을 확인할 수 있다.

이상에서 살펴본 바와 같이, 링크다운이 발생한 경우 대역폭이 커질수록 SCTP의 성능이 TCP의 성능보다 우수한 것으로 확인되었다. 결과를 요약하면 대역폭을 변경시켰을 때, $SCTP_{_multi\text{-}homing}$의 처리율이 TCP의 그것보다 평균적으로 9% 우수한 것으로 측정되었다.

V. 결론

본 연구에서는 링크다운 환경에서 NS-2 시뮬레이션을 이용하여 SCTP의 멀티호밍 효과를 측정하였다. 본 연구의 실험 방법은 다음과 같다.

먼저 링크를 다운하지 않은 초기 실험에서 시뮬레이션 시간을 100 초와 500 초로 하여 실험한 결과 SCTP와 TCP의 처리율과 대역폭 이용률이 동일하게 나왔다. 따라서 시뮬레이션 시간을 100 초로 하여 링크다운 시간, RTT 그리고 대역폭을 변경해 가면서 실험을 하였고 CWND(congestion window)의 크기를 관찰하면서 SCTP와 TCP의 처리율과 대역폭 이용률을 관찰하였다.

본 연구에서 얻어진 결과는 다음과 같다.

첫째, 링크다운 시간이 5초, 10초, 20초인 경우 $SCTP_{_multi\text{-}homing}$의 처리율과 대역폭 이용률이 TCP보다 평균적으로 18% 우수한 것으로 확인되었다.

둘째, 링크가 다운된 시점에서 CWND가 어떠한 값을 갖느냐에 따라 처리율과 대역폭 이용률이 달라짐을 알 수 있었다.

셋째, RTT와 대역폭을 변화시킨 링크다운 실험에서는 SCTP_multi-homing의 처리율과 대역폭 이용률이 TCP보다 각각 27%, 9% 우수한 것으로 판명되었다.

넷째, 모든 실험에서 SCTP_single-homing과 TCP 사이에는 뚜렷하게 성능차이가 발견되지 않았다. 그러나 링크다운이 발생하지 않는 경우 TCP가 SCTP_multi-homing보다 처리율이 2% 우수하였다.

다섯째, SCTP_multi-homing의 처리율이 TCP와 SCTP_single-homing의 처리율보다 평균 12% 우수하였다.

이상의 결과를 종합하면 SCTP_multi-homing의 처리율과 대역폭 이용률이 TCP보다 평균적으로 18% 우수한 것으로 판명되었다. 본 연구의 결과는 시뮬레이션에 의한 결과로 유동성이 있음을 감안해야 하고, 실제 네트워크 환경에서의 실험 결과와 다를 수 있다. 그렇지만 본 연구의 결과는 실제 인터넷의 경로상의 링크가 다운되는 경우 TCP에 대한 SCTP의 멀티호밍 효과를 추정하는데 기초 자료로 활용될 수 있을 것이다. 본 연구에서는 링크다운 환경에서 FTP 트래픽을 사용하였지만 추후 연구에서는 인터넷에서 주로 이용되는 WEB 트래픽을 사용한 연구가 기대된다.

참 고 문 헌

[1] 김주회, *HTTP를 응용계층 프로토콜로 사용한 TCP와 시뮬레이트 SCTP의 성능 평가*, 우송대 정보산업대학원, 석사학위 논문, 2005.

[2] 김주현, *리눅스 환경에서의 SCTP 혼잡제어 분석*, 한국교원 대학교, 석사학위 논문, 2008.

[3] 박재성, 고석주, "리눅스 환경에서 SCTP와 TCP 프로토콜의 성능 비교", 한국통신학회지, 제33권, 제8호, 2008.

[4] 배성수, 한종수, *네트워크 시뮬레이션(NS-2 기초와 활용)*, 세화출판사, 2005.

[5] 송정화, 이미정, 고석주, "SCTP의 멀티호밍 특성에 대한 성능 평가", 정보처리학회논문지, 제11-C권, 제2호 통권 제91호, 2004.

[6] 이용진, "초기 슬로우 스타트 구간에서 웹 객체의 평균 전송 시간 추정을 위한 수학적 모델", 대한공업교육학회지, 제33호 통권 제2호, 2008.

[7] 하종식, 고석주(2005). *리눅스 기반 SCTP & TCP 성능 비교 분석*, 2005. http://protocol.knu.ac.kr/tech/CPL-TR-05-02.pdf에서 인출

[8] F. Nadei, Mir, *Computer and Communication Networks*, Pearson Education, 2007.

[9] R. Stewar, Q. Xie, *Stream Control Transmission Protocol*, IETF RFC 2960, 2000.

[10] R. Stewart, Q. Xie, *Stream Control Transmission Protocol(SCTP) : A Reference Guide*, Addison Wesely, 2001.

[11] http://www.isi.edu/nsnam/ns

[12] http://nsnam.org

저 자 소 개

최 용 운(Yong-Woon Choi) 정회원

- 2004년 2월 : 연세대학교 물리학과(이학사)
- 2006년 8월 : 연세대학교 대학원 컴퓨터교육과(교육학석사)
- 2008년 3월 ~ 현재 : 한국교원대 기술교육과 석사과정

<관심분야> : 네트워크 시뮬레이션, 컴퓨터 교육

이 용 진(Yong-Jin Lee) 정회원

- 1995년 2월 : 고려대학교 전산과학과(이학박사)
- 1995년 3월 ~ 2005년 8월 : 우송대학교 컴퓨터정보학과 부교수
- 2005년 9월 ~ 현재 : 한국교원대학교 기술교육과 부교수

<관심분야> : 무선 및 이동 네트워크, 인터넷 QoS, 정보통신기술 교육