大 韓 工 業 敎 育 學 會 誌
第 32卷 第 2號 pp.199-216

초기 슬로우 스타트 단계에서 SCTP의 평균 전송 시간*

김 주 현** · 이 용 진***

<국문초록>

SCTP(stream control transmission protocol)는 데이터 전송을 위한 전송 계층 프로토콜로서, 많은 부분에서 TCP(transmission control protocol) 방식을 따른다. 하지만 멀티 호밍(multi-homing)과 멀티 스트리밍(multi-streaming)의 특징을 가짐으로 성능의 차이를 갖는다.

이 논문에서는 SCTP 혼잡제어 중에서 초기 슬로우 스타트 단계에 초점을 맞추어 데이터 전송을 분석하고, 대역폭, 지연시간 및 데이터 크기에 따른 SCTP와 TCP 평균 전송 시간을 측정하고 비교하였다.

아울러 SCTP와 TCP의 평균 전송시간에 영향을 미치는 요인인 초기 윈도우 크기를 데이터 크기에 따라 측정하였다.

실험을 위한 서버와 클라이언트 프로그램은 SCTP socket API를 이용하여 C 언어로 작성되었고, 전송 시간은 이더리얼 프로그램을 사용하여 측정되었다. 서버와 클라이언트 사이의 데이터 전송 방법은 라운드 로빈(round robin) 방법을 사용하였다.

실험 결과, SCTP는 초기 슬로우 스타트 단계에서 TCP 보다 평균 전송 시간에 있어 약 15% 정도 향상된 성능을 보였으며, 그 이유는 SCTP 초기 윈도우 크기가 TCP 보다 크기 때문으로 확인되었다.

주제어 : 초기 슬로우 스타트 단계, 초기 윈도우, 혼잡제어

* 이 논문은 2006년 정부의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2006-521-D00399).
** 주엽공업고등학교
*** 교신저자 : 이메일(lyj@knue.ac.kr) 한국교원대학교 기술교육과

Ⅰ. 서 론

사용자가 인터넷에 브라우저를 이용하여 접속하면 웹 화면이 나타난다. 이 때 사용자가 보는 웹 화면을 구성하는 다양한 이미지, 플래시 및 텍스트 등은 웹 서버로부터 사용자의 컴퓨터로 전송된 것이다. 각각의 이미지, 플래시 및 텍스트는 파일이다. 일반적으로 현재 주로 사용되고 있는 웹 화면은 하나의 파일로 이루어진 것이 아니라, 수많은 파일들로 구성되어 있다. 이 많은 파일들이 한번에 사용자에게 전송되면 좋겠지만, 수많은 사용자들이 함께 사용하는 인터넷 환경에서는 중간 매체의 용량을 고려하여 전송해야 한다. 만약 중간 매체 용량을 고려하지 않고 파일을 전송한다면 네트워크에 혼잡이 일어나 손실이 발생할 수 있다. 손실이 발생 함에도 불구하고 계속 데이터양을 줄이지 않는다면, 결국 네트워크에 데이터가 폭주하여 인터넷을 사용할 수 없을 것이다.

이런 네트워크의 혼잡을 미연에 방지하고, 만약 발생할 경우 네트워크 혼잡 상황을 복구하기 위해 종단 시스템에 있는 전송 계층 프로토콜은 혼잡제어(congestion control) 알고리즘을 통해 송수신자간에 전송되는 패킷률을 조절한다(강현국, 안상현, 신용태, 최종원, 2005, p. 29).

현재 가장 널리 사용되고 있는 전송 계층 프로토콜은 TCP(transmission control protocol)이다. 하지만 TCP는 초기 유선 환경을 기반으로 개발되었기에, 최근 사용이 증가하고 있는 실시간 서비스, 멀티미디어 서비스와 무선 및 모바일 통신에는 한계가 있다. 이에 대한 대안으로 IETF의 SIGTRAN 그룹에서 IP 기반 네트워크를 통한 신호화 전송을 위해 설계되었던 SCTP(stream control transmission protocol)가 제안되었다.

SCTP는 기존 전송 계층 프로토콜인 TCP와 가장 큰 차이점인 멀티 호밍(multi-homing)과 멀티 스트리밍(multi-streaming) 특성을 가짐으로 TCP의 HOL(head of line) 블로킹, 네트워크 손실, SYN 공격 등의 한계점을 극복할 수 있다.

이전 SCTP 혼잡제어 관련 논문들(송정화, 이미정, 고석주, 2003, Ye, Saadawi, & Lee, 2002, Donato, Salvatore, Pescape, & Ventre, 2006)들은 혼잡제어 전체에 관심을 두고 실험하였지만, 본 연구에서는 초기 슬로우 스타트 단계에 초점을 맞추어 실험하였다.

이를 위해 본 논문에서는 서버와 클라이언트를 라우터로 연결하는 네트워크 환경을 구성하였다. 또한 SCTP 서버 및 클라이언트 프로그램을 SCTP socket API를 이용하여 C 언어로 작성한 후, 실험을 통해 초기 슬로우 스타트 단계에서 전송량을 분석하였다. 더불어 TCP의 전송시간을 측정하여 SCTP와 비교하여, SCTP 전송시간이 TCP 보다 평균 15 % 우수함을 확인했다.

이 연구는 패킷 손실이 없는 환경에서 SCTP를 사용했을 때 어느 정도의 효과가 있는지 확인할 수 있고, 새로운 응용 프로그램 개발 시 전송 프로토콜을 선택할 때 기

초 자료로 활용될 수 있다.

전체 5장으로 구성되는 본 연구는 2장에서 SCTP의 초기 슬로우 스타트에 대해 설명하고, 3장에서 시스템 환경의 구성 및 실험 방법에 대해 설명한다. 4장은 실험 측정 결과이고, 마지막 5장에서 결론을 맺는다.

Ⅱ. SCTP의 초기 슬로우 스타트 단계

SCTP는 원래 IP 네트워크를 통해 PSTN 시그널링 메시지를 전송하기 위해 설계되었으나, 모바일 서비스의 등장 및 소비자의 다양한 컨텐츠의 요구 등으로 새로운 전송 계층 프로토콜로 제안되고 있다(RFC 2960).

지속적인 무선 및 모바일 단말기의 사용 증가로 인해 SCTP에 대한 연구가 다방면으로 진행 중이지만 본 논문에서는 혼잡제어 알고리즘 관련 논문에 대해서만 언급한다.

Alamgir와 Ivancic(2005)와 Donato 등(2006)의 논문은 다양한 변수를 조절하여 측정하였지만, 네트워크 시뮬레이터 ns-2를 이용하여 실험하였다. ns-2는 너무 이상적인 환경을 제공하기에 실제 네트워크에 결과를 적용하기에는 무리가 있다.

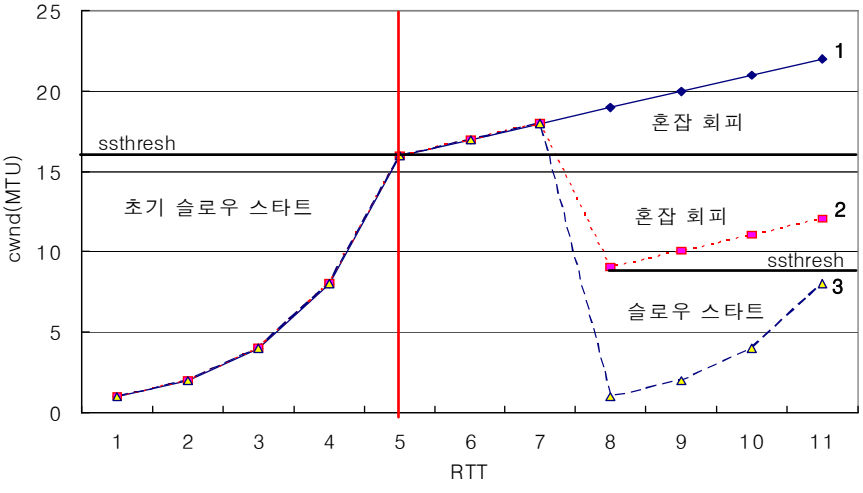
하종식, 고석주(2005)는 두 대의 단말기를 통해 실험 환경을 구성하여, 전송 버퍼 사이즈를 조정하여 전송시간, 데이터양 및 총 수신 패킷 수를 측정하였다. 그리고 SCTP와 TCP의 성능을 비교하여, 전송 버퍼 사이즈를 고려하지 않고 무작위로 전송하는 경우에 TCP가 SCTP 보다 처리율이 높다는 것을 확인했다.

하지만 이 논문에서는 단지 전송 버퍼 사이즈만 변경하여 실험함으로써, 실제 네트워크 상태를 반영하지 못한 단점이 있다. 그래서 본 논문에서는 직접 라우터에서 대역폭과 지연시간 조정을 통해 실험하였다.

위에서 언급한 논문들은 혼잡제어 전체 부분을 다루고 있다. 하지만 본 논문에서는 SCTP 초기 슬로우 스타트 과정에서 전송되는 데이터양을 측정하여 분석한다. 더불어 대역폭, 지연시간 및 데이터 크기에 따른 전송시간을 TCP와 비교하였다. 이 때 네트워크가 혼잡에 이르지 않도록 하기 위해 송신자 측에서 보내는 데이터양을 혼잡 윈도우(congestion control window, cwnd)라고 한다. 초기(initial) cwnd는 서버에서 클라이언트로 처음 보내는 데이터의 양을 뜻한다.

데이터 전송을 처음 시작할 때, [그림 1]과 같이 초기 슬로우 스타트 상태로 시작한다. [그림 1]은 RTT(round trip time)에 따른 cwnd 크기를 나타낸다. 슬로우 스타트에서는 cwnd가 RTT 마다 2배씩 증가하며, 혼잡회피 구간에서는 1*MTU(maximum transmission unit)씩 증가한다. 슬로우 스타트와 혼잡 회피 사이의 경계를 슬로우 스타트 한계치(slow start threshold, ssthresh)라 한다. 수신측에서 cwnd가 ssthresh 보다 작

거나 같으면, 슬로우 스타트 알고리즘을 사용하고, cwnd가 ssthresh 보다 크면 혼잡 회피 알고리즘을 사용한다(RFC 2960).

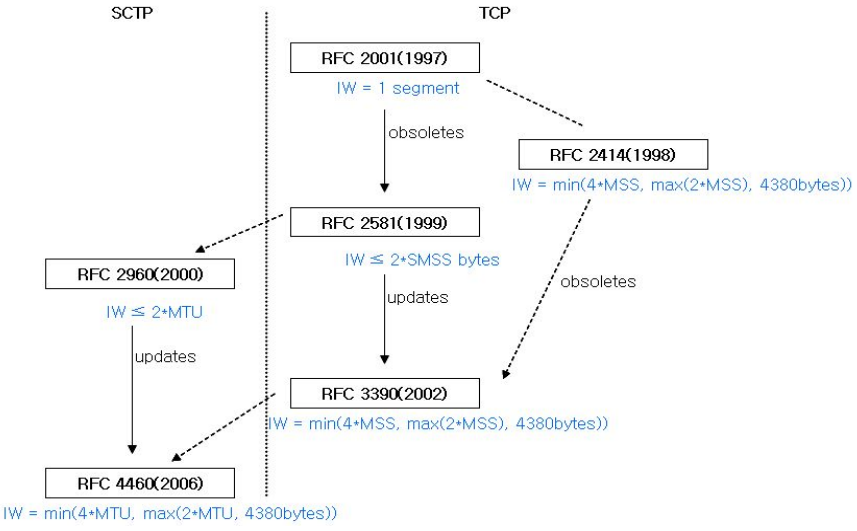


[그림 1] 슬로우 스타트와 혼잡 회피의 도식화(단, 1 : 무손실, 2: SACK에 의해 손실 보고, 3: 타임아웃으로 손실 감지)

본 논문에서는 [그림 1]에서 3과 같이 타임아웃으로 인해 서버가 손실을 감지해서 cwnd가 1*MTU로 감소하여 시작하는 슬로우 스타트 부분을 제외한 **초기 슬로우 스타트** 단계에 맞추어 실험하였다.

일반적으로 초기 ssthresh는 상대방의 수신 윈도우 크기로 구성되기 때문에, 매우 큰 값이다. 그래서 손실이 없는 네트워크 환경에서는 슬로우 스타트에서 대다수의 전송의 끝나기 때문에 초기 cwnd 값은 데이터 전송 시간에 많은 영향을 미칠 것이다. [그림 2]는 초기 cwnd 크기 변화에 따른 RFC 표준 문서의 발전 과정이다. 초기 cwnd 크기가 점차 증가하는 것을 확인할 수 있다.

초기 cwnd가 점차 증가하고는 있지만 그것에 단점 또한 존재한다. 큰 초기 cwnd의 단점은 혼잡이 계속 일어나는 네트워크 환경에서 큰 초기 cwnd로 인해 손실이 더욱 크게 발생하는 것이다. 하지만 손실이 없을 경우에는, 본 논문의 실험을 통해 확인한 바와 같이 전송 시간을 줄일 수 있는 장점이 있다. 일반적으로 MTU가 1500 bytes이고, 이메일 또는 웹 화면이 4 KB 보다 작다면 1 RTT 동안 전송이 완료된다(RFC 2414).



[그림 2] 초기 cwnd 변화에 따른 표준안 발전과정(단, IW는 초기혼잡윈도우)

SCTP 초기 cwnd 크기는 [그림 2]에 나타난 바와 같이 RFC 2960에서 "2*MTU 보다 작거나 같다"로 정의되었으나 RFC 4460에서 식 (1)으로 변경되었다.

$$initial\_cwnd = min(4*MTU, max(2*MTU, 4380\ bytes)) \quad (1)$$

위 식에서 초기 cwnd는 MTU 값으로 결정되는 것을 알 수 있다. 각 MTU 값에 따른 초기 cwnd 값은 다음과 같다.

만약 MTU <= 1095bytes 이면 초기 cwnd <= 4*MTU ,
만약 1095 bytes < MTU <= 2190 bytes 이면 초기 cwnd <= 4380 bytes ,
만약 2190 bytes < MTU 이면 초기 cwnd <= 2*MTU 이다.

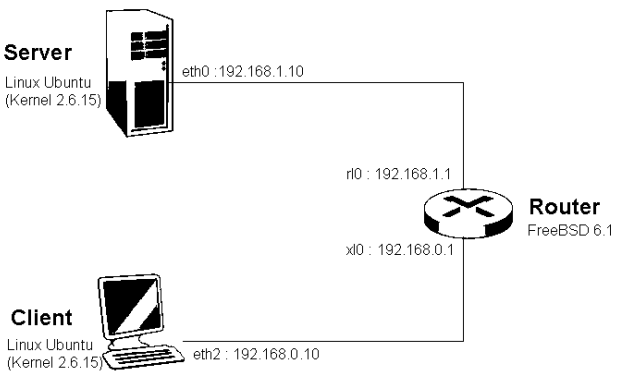
본 논문에서는 실험을 통해 식 (1)과 비교하여 어떻게 데이터가 전송되는지를 직접 확인하고 전송 시간을 측정하여, 초기 cwnd 크기가 전송 시간에 어떤 영향을 미치는지를 확인했다. 더불어 SCTP와 TCP의 전송 시간을 측정하여 성능을 비교하였다.

Ⅲ. 연구 방법

이 연구에서는 IP 주소를 하나만 갖는 서버와 클라이언트, 즉 싱글홈드 호스트(single-homed host)에서 SCTP 혼잡제어 분석을 하고 아울러 변수를 조정하여 TCP와 전송시간을 비교하였다.

1. 시스템 환경 및 실험 구성

본 연구의 실험 환경은 [그림 3]와 같이 서버와 클라이언트가 라우터로 연결되어, 라우터에서 직접 대역폭과 지연시간을 조정하고 서버에서 데이터 크기를 변경하여 실험하였다. 실험에 사용되는 서버, 클라이언트 및 라우터의 운영체제와 이더넷 인터페이스 주소는 <표 1>과 같다.



[그림 3] 실험 장비의 구성

<표 1> 실험 환경

구분	서버	라우터	클라이언트
하드웨어	펜티엄 D 3.4G RAM 1.28G HDD 300G	펜티엄 P3 500M 128M 528M	펜티엄 P4 2.4G RAM 1.28G HDD 40G
OS	linux ubuntu (커널 2.6.15)	FreeBSD 6.1	linux ubuntu (커널 2.6.15)
SCTP	lksctp-2.6.15-1.0.5		lksctp-2.6.15-1.0.5
IP	192.168.1.10	192.168.1.1(to server) 192.168.0.1(to client)	192.168.0.10

서버와 클라이언트의 애플리케이션 프로그램은 C 언어를 이용하여 네트워크 소켓 프로그래밍으로 직접 코딩하여 구현하였다. SCTP 서버와 클라이언트 애플리케이션의 의사코드(pseudo code)는 각각 [그림 4], [그림 5]와 같다.

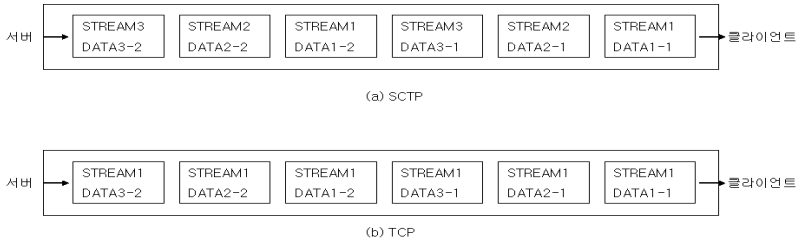
```
listenSock=socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP); //create a SCTP socket
bind();
listen(listenSock, ..);
while() {
        scanf(.. , &send_stream); //input the number of data or stream
        connectSock=accept(listenSock, ..); //await a new client connection
        for() {                         //new client socket has connected
                fp[i]=open("test.dat", ..); //open files
        }
        t[i]=read(fp[i], ..);
        while() {
                ret=sctp_sendmsg(connSock, ..); //send a file(test.dat) to client
                for() {
                        buffer[i][..]=0;
                }
        ret=0;
        i=i+1;
        if(i==send_stream) i=0; //round robin
        t[i]=read(fp[i], ..);
        close(connSock); //close the client connection
}
return 0;
}
```

[그림 4] SCTP 서버 애플리케이션 의사코드

```
sock=socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP); // create a SCTP socket
connect(sock, ..); //request connection to server
do{
        sctp_recvmsg(sock, ..); //read a 'test. dat' received from server
} while(1);
close(sock);
```

[그림 5] SCTP 클라이언트 애플리케이션 의사코드

서버 애플리케이션은 데이터의 수, 즉 스트림의 개수를 결정하고 클라이언트에서의 연결 설정을 기다린다. 연결 요청(connect( ))이 들어오면 이를 수락(accept( )) 하고, 연결 설정 완료 후 바로 "test.dat" 파일을 전송한다. 데이터의 전송은 라운드 로빈 방식으로 실험했다. 라운드 로빈은 [그림 6]과 같이 전송되는 기법이다. 이는 데이터 전송을 하는데 있어서 서비스 우선순위가 존재하지 않기 때문에, 정해진 시간 동안에 차례대로 번갈아 가면서 전송하게 된다(강현국, 안상현, 신용태, 최종원, 2005, p. 611). [그림 3]에서 라운드 로빈 전송 과정을 SCTP와 TCP로 나누어 나타냈다. SCTP는 스트림 1로 데이터 1의 첫 번째 부분을 보내고, 스트림 2로 데이터 2의 첫 번째 부분을 전송한다. 그 다음 자신의 순번이 되어서 남지 데이터를 전송한다. 이에 비해 TCP는 단일 스트림이기 때문에, 하나의 스트림을 통해 라운드 로빈 방식으로 데이터를 전송한다. 스트림 1로 데이터 1의 첫 번째 부분을 보내고, 동일 스트림으로 데이터 2의 첫 번째 부분을 전송한다. 이렇게 동일 스트림으로 나머지 부분의 데이터도 다음 순번에 전송한다.

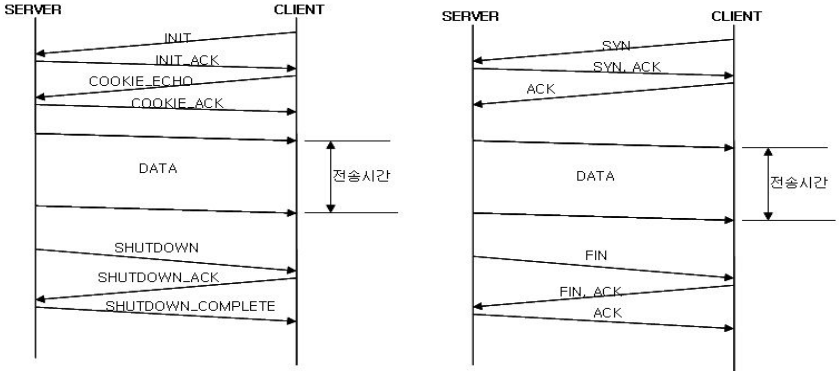


[그림 6] 라운드 로빈 데이터 전송

TCP 서버 및 클라이언트 애플리케이션은 SCTP와 동일한 방식으로 구현되었다.

2. 실험 방법 및 절차

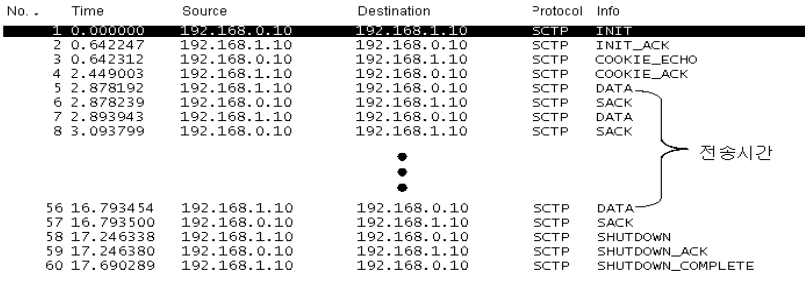
전송 시간은 SCTP와 TCP 프로그램의 실행에서 연결 설정 부분과 연결 해지 부분을 제외하고 [그림 7]과 같이 클라이언트 측에서 처음 데이터가 들어오는 시점부터 마지막 데이터가 들어오는 시간까지를 측정했다.



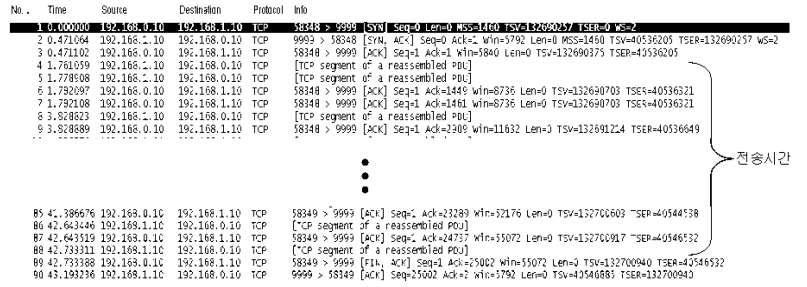
(a) SCTP 전송시간 (b) TCP 전송시간

[그림 7] 전송시간 측정

실제 실험을 통해 SCTP 패킷을 캡처한 화면은 [그림 8]와 같다. [그림 8]에서와 같이 처음 데이터가 들어오는 시간부터 마지막 데이터가 들어오는 시간까지를 전송시간이라 하였다. 이와 마찬가지로 TCP 패킷 캡처 화면은 [그림 9]과 같다.



[그림 8] SCTP 데이터 전송시간 측정 이더리얼 캡처



[그림 9] TCP 데이터 전송시간 측정 이더리얼 캡처

Ⅳ. 실험 결과 및 분석

이 장에서는 손실이 없는 환경에서 라운드 로빈 방식으로 데이터 전송을 하는 SCTP 서버와 클라이언트 프로그램의 실행을 통해 초기 슬로우 스타트에서 전송량을 분석했다. 더불어 대역폭, 지연시간 및 데이터 크기에 따른 전송시간을 측정하여 TCP와 비교하였다. 마지막으로 전송시간 차이점의 이유라고 생각되는 초기 cwnd 크기를 TCP와 비교하였다. 각 실험 결과는 10번의 측정을 통해 평균값을 나타냈고, 전송시간은 소수점 셋째 자리에서 반올림하였다. 또한, 실험에서 SCTP와 TCP 송신 버퍼 사이즈는 1460 bytes로 동일하고, 1*MTU는 1500 bytes이다.

1. 초기 슬로우 스타트 단계에서 전송량 분석

[그림 10]에 나타난 바와 같이, IP 헤더가 20 bytes, SCTP 일반 헤더가 12 bytes, SCTP 데이터 청크의 헤더가 16 bytes가 되므로, MTU가 1500 bytes에서 전송할 수 있는 최대 데이터의 크기는 1452 bytes이다.

IP	SCTP 일반헤더	SCTP 데이터 청크
(20 bytes)	(12 bytes)	(헤더 16 bytes+데이터 1452 bytes=1468 bytes)

[그림 10] SCTP 패킷의 구조

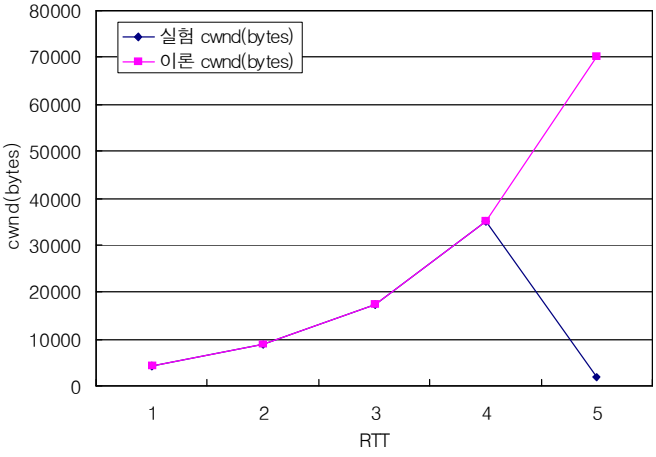
이 실험에서 전송 버퍼 사이즈가 1460 bytes이므로 이 크기로 전송되어야 하나, 최대로 전송 가능한 데이터 크기가 1452 bytes이므로 각 데이터는 1452 bytes와 8 bytes로 나뉘어져 전송된다.

대역폭 10 Mbps, 지연시간 10 ms인 환경에서 13.5 KB의 데이터를 5개 전송하는 실험의 결과는 <표 2>와 같다. <표 2>를 보면 각 RTT 마다 전송된 데이터의 SID 번호, SSN 번호와 데이터 크기를 나타냈다. 각 데이터가 라운드 로빈 방식에 따라 전송되다가 마지막 데이터가 전송되는 RTT 5 구간에서 SID 0/SSN 9, SID 1/SSN 9와 SID 2/SSN 9가 한 패킷에 번들링(bundling)되어 전송되고, 다음 SID 3/SSN 9와 SID 4/SSN 9가 함께 전송되었다.

<표 2>에 나타난 실험 cwnd 크기와 식 (1)로 구한 초기 cwnd로 시작하여 두 배씩 증가하는 슬로우 스타트 알고리즘을 비교하여 [그림 11]에 나타냈다.

<표 2> RTT에 따른 데이터 전송

RTT	SID	SSN	data (bytes)	RTT	SID	SSN	data (bytes)	RTT	SID	SSN	data (bytes)
1	0	0	1452	3	1	3	1452	4	2	6	8
	0	0	8		1	3	8		3	6	1452
	1	0	1452		2	3	1452		3	6	8
	1	0	8		2	3	8		4	6	1452
	2	0	1452		3	3	1452		4	6	8
	2	0	8		3	3	8		0	7	1452
전송 데이터 양(1 RTT)			4380		4	3	1452		0	7	8
2	3	0	1452		4	3	8		1	7	1452
	3	0	8		0	4	1452		1	7	8
	4	0	1452		0	4	8		2	7	1452
	4	0	8	전송 데이터 양(3 RTT)			17520		2	7	8
	0	1	1452	4	1	4	1452		3	7	1452
	0	1	8		1	4	8		3	7	8
	1	1	1452		2	4	1452		4	7	1452
	1	1	8		2	4	8		4	7	8
	2	1	1452		3	4	1452		0	8	1452
	2	1	8		3	4	8		0	8	8
	3	1	1452		4	4	1452		1	8	1452
	3	1	8		4	4	8		1	8	8
전송 데이터 양(2 RTT)			8760		0	5	1452		2	8	1452
3	4	1	1452		0	5	8		2	8	8
	4	1	8		1	5	1452		3	8	1452
	0	2	1452		1	5	8		3	8	8
	0	2	8		2	5	1452		4	8	1452
	1	2	1452		2	5	8		4	8	8
	1	2	8		3	5	1452	전송 데이터 양(4 RTT)			35040
	2	2	1452		3	5	8	5	0	9	360
	2	2	8		4	5	1452		1	9	360
	3	2	1452		4	5	8		2	9	360
	3	2	8		0	6	1452		3	9	360
	4	2	1452		0	6	8		4	9	360
	4	2	8		1	6	1452				
	0	3	1452		1	6	8	전송 데이터 양(5 RTT)			1800
	0	3	8		2	6	1452				



[그림 11] RTT에 따른 cwnd의 변화

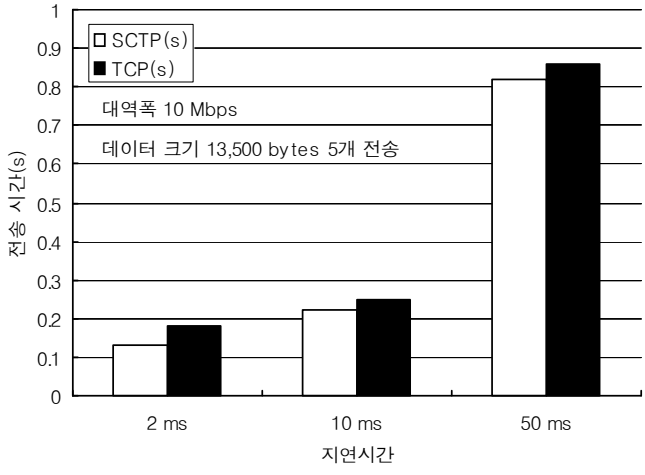
[그림 11]을 보면, 이 실험에서 이론 cwnd와 실험 cwnd는 동일하다. 처음 RTT 1 구간에서 4380 bytes를 전송되는데, 이는 식 (1)에서 구한 값과 같다. RTT 2 구간에서는 이전 구간 cwnd의 두 배인 8760 bytes가 전송된다. RTT 3 구간에서는 RTT 2 구간의 두 배인 17520 bytes가 전송된다. RTT 4 구간에서는 RTT 3 구간의 두 배인 35040 bytes가 전송되고, 마지막 구간인 RTT 5 구간에서는 패킷의 나머지 1800 bytes가 전송되기 때문에, 그래프에서 이론 cwnd와 달라진 것이다.

2. SCTP와 TCP의 평균 전송시간 비교

본 논문에서는 SCTP 혼잡제어 분석에 추가적으로 지연시간, 대역폭 및 데이터 크기에 따른 전송시간을 TCP와 비교하였다.

가. 지연 시간에 따른 전송 시간

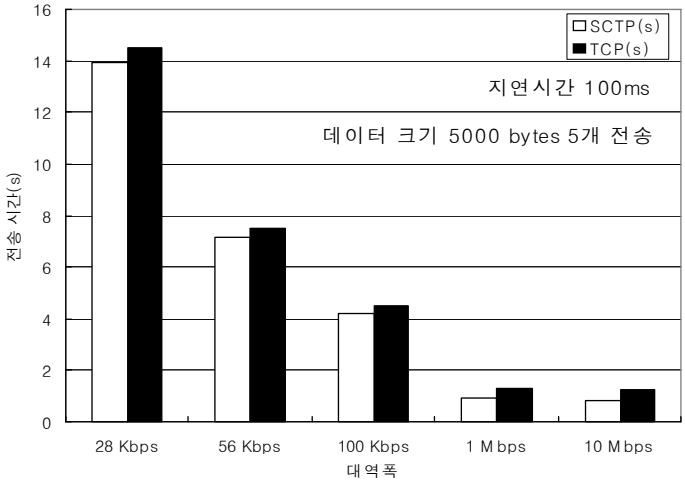
대역폭 10 Mbps이고, 13.5 KB 데이터 5개를 전송하는 경우에 지연시간에 따른 실험 결과는 [그림 12]과 같다. [그림 12]에서 지연 시간이 2 ms, 10 ms, 50 ms인 각각의 경우에 SCTP가 TCP 보다 39 %, 12 %, 5% 정도 전송시간이 짧다. 실험 환경이 서버와 클라이언트가 라우터 하나로 연결되어 있어 타이머 지정 시간보다 지연 시간이 길 경우 손실이 발생해 응답이 오지 않는다고 생각해 서버에서 재전송을 실시하였다. 지연시간이 2 ms인 경우에도 재전송이 발생하는 데, 이는 너무 빠른 데이터 전송에 의해 중복 응답(duplicate acknowledge)이 발생하고 이에 대응하여 빠르게 재전송이 발생하기 때문이다. 또한 마지막 데이터에 대한 동일한 SACK(selective acknowledge)의 폭주가 일어난다.



[그림 12] 지연시간에 따른 전송 시간 측정

나. 대역폭에 따른 전송 시간

라우터의 지연 시간이 100 ms일 때, 5000 bytes 데이터를 5개 전송하는 경우의 대역폭에 따른 실험 결과는 [그림 13]와 같다.

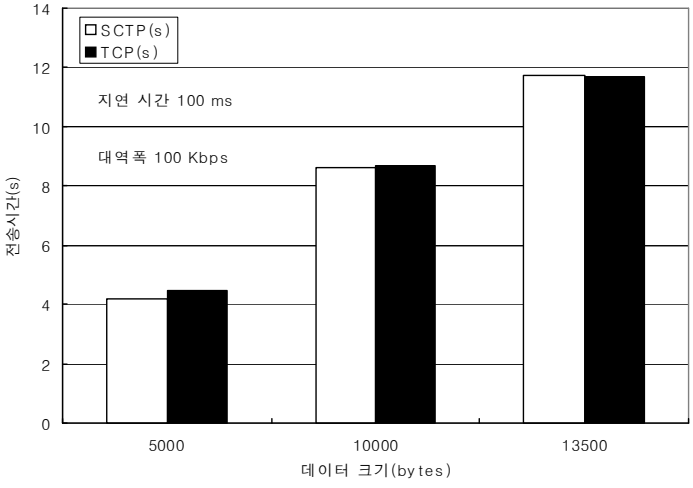


[그림 13] 대역폭에 따른 전송 시간

대역폭이 28 Kbps, 56 Kbps, 100 Kbps, 1 Mbps, 10 Mbps일 경우, 각각 4 %, 5 %, 7 %, 37 %, 49 % 정도 SCTP가 TCP 보다 전송 시간이 짧다. SCTP가 TCP 보다 대역폭이 커질수록 상대적 효율이 높은 것을 알 수 있다.

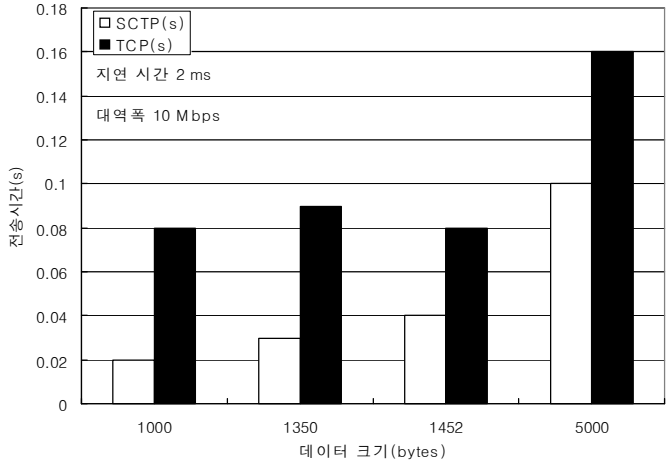
다. 데이터 크기에 따른 전송 시간

지연 시간이 100 ms, 대역폭이 100 Kbps인 경우, 데이터 크기에 따른 실험 결과는 [그림 14]와 같다. 각 데이터들은 5개씩 전송된다.



[그림 14] 데이터 크기에 따른 전송 시간

지연 시간이 위의 실험보다 짧은 2 ms, 대역폭이 상대적으로 큰 10 Mbps인 경우의 각 데이터 크기에 따른 전송 시간의 결과는 [그림 15]과 같다. 이 때, 각 데이터들은 10개씩 전송된다.



[그림 15] 데이터 크기에 따른 전송 시간

위 실험 결과를 보면 데이터 크기가 작을수록 SCTP의 효율이 TCP에 비해 상대적

으로 높은 것을 확인할 수 있다.

3. SCTP와 TCP의 초기 cwnd 비교

대역폭, 지연시간 및 데이터 크기에 따른 전송 시간 측정 실험에서 SCTP가 TCP 보다 더 나은 성능을 보이는 이유는 초기 윈도우 값이 더 크고, 멀티 스트림을 사용하기 때문이다. 그 중 손실이 없는 환경에서 큰 영향을 미친다고 생각되는 초기 cwnd 크기를 실제 실험을 측정하여 이론 크기와 비교하였다.

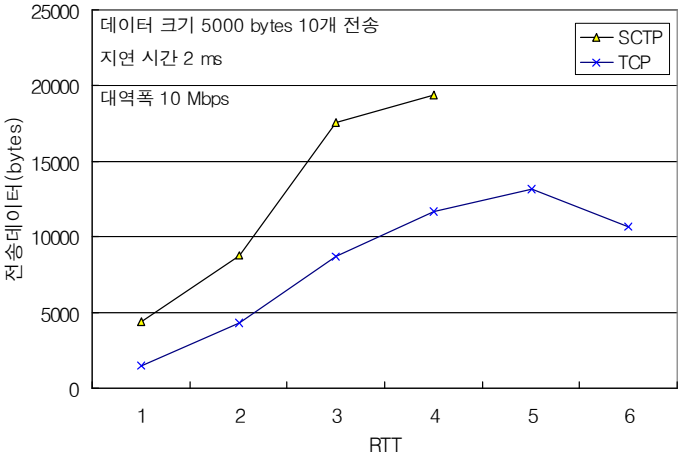
본 연구의 실험에서는 1*MTU 값이 1500 bytes이므로 식 (1)의 결과에 따라 초기 cwnd는 4380 bytes이다. <표 3>은 각각의 데이터를 10개씩 전송했을 경우에, 초기 cwnd 크기가 몇 bytes 인지를 나타냈다. 하지만 실제 전송에서는 항상 이론값으로 전송되지 않고 <표 3>와 같이 전송되었다.

<표 3> TCP와 SCTP의 초기 cwnd 크기(단위: bytes)

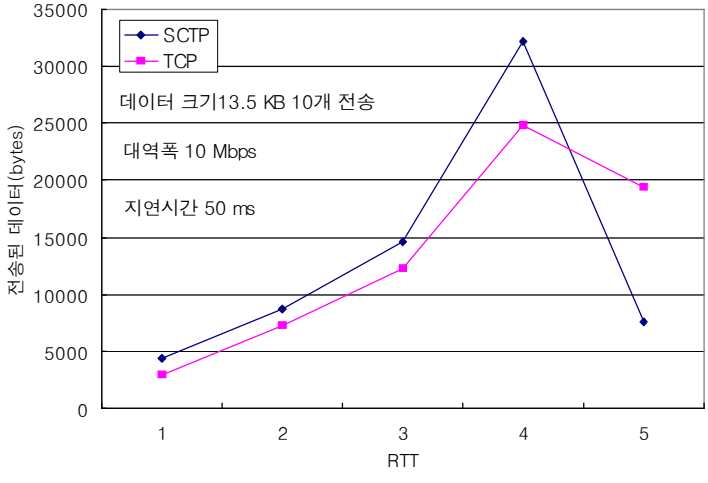
데이터크기 / 프로토콜	1000 bytes	1350 bytes	1448 bytes	1452 bytes	5000 bytes
TCP	3896	4246	4344	2900	2908
SCTP	5000	5400	5792	5808	4380

<표 3>을 보면 TCP는 초기 cwnd 크기보다 데이터양을 작게 전송하지만, SCTP는 이를 초과하여 하나의 데이터를 더 전송할 수 있다. SCTP에서 5000 bytes 이상의 데이터를 전송하는 경우에는 초기 cwnd 크기가 4380 bytes로 동일하였다. <표 3>을 통해 전송하는 데이터의 크기에 따라 초기 cwnd 값이 달라짐을 확인할 수 있다.

이에 따라 전송 시간 또한 달라진다. 데이터 크기가 작을수록 초기 cwnd 값이 전송 시간에 크게 영향을 미치기 때문에 [그림 16]과 같이 SCTP가 TCP 보다 먼저 전송이 끝나거나, [그림 17]과 같은 경우처럼 마지막 RTT 구간에서 SCTP가 적은 데이터양이 전송되기 때문에 전송시간이 짧아진다.



[그림 16] RTT에 따른 전송된 데이터양



[그림 17] RTT에 따른 전송된 데이터의 양

V. 결 론

본 연구는 리눅스 환경에서의 SCTP 초기 슬로우 스타트 단계에서 데이터 전송 시간을 측정하고, 초기 cwnd가 이에 미치는 영향을 분석하였다. 기존 연구에서는 혼잡제어 전체에 관심을 두고 실험되었지만, 본 연구는 초기 슬로우 스타트 단계에서 데이터 전송에 초점을 둔다. 그리고 SCTP socket API를 이용하여 C 언어로 SCTP 서버와 클라이언트 프로그램을 작성하여, 실제로 초기 슬로우 스타트에서 전송되는 데이터양을 분석했다. 더불어 라우터에서의 대역폭과 지연시간을, 서버에서 데이터 크기를 조정하여 클라이언트 측에서 전송시간을 측정하여 TCP와 비교하였다.

초기 슬로우 스타트 단계에서 SCTP 데이터 전송은 RFC 4460과 거의 유사하였다. 그리고 라우터에서 대역폭과 지연시간 조정으로 SCTP와 TCP의 평균 전송시간을 비교하여 평균적으로 TCP 보다 SCTP가 약 15 %정도 우수한 성능을 보임을 확인하였다. 이는 실제 실험을 통해 측정된 SCTP와 TCP의 초기 cwnd 크기를 비교하여 확인한 결과, SCTP는 이론 cwnd 크기와 거의 유사하게, TCP는 그 보다 작게 전송하기 때문이다.

이 연구는 앞으로 새로운 네트워크 응용 프로그램을 개발하는 데 있어 어떤 전송계층 프로토콜을 선택 하는가에 대한 배경으로 사용될 수 있고, SCTP 혼잡제어 관련 연구의 기본 자료로 활용될 수 있다.

하지만 본 논문은 손실 상황을 고려하지 않아서 무선 네트워크에 적용하기에는 무리가 있다. 그래서 손실이 발생했을 때, 네트워크의 신뢰성을 높일 수 있는 SCTP 멀티 호밍에서 각 주소에 따른 데이터 전송량을 확인하고 전송시간을 측정하는 연구가 이루어져야 할 것이다.

참 고 문 헌

강현국, 안상현, 신용태, 최종원(2005). *컴퓨터 네트워킹 제 3판*. 서울 : 피어슨에듀케이션코리아. [원저: Kurose, J. F. & Ross, K. W. (2005). computer networking. Addison Wesley].

송정화, 이미정, 고석주(2003). SCTP의 멀티호밍 특성 및 재전송 정책에 대한 성능 평가. 한국정보과학회 추계학술발표논문집, 30(2). 88-90.

하종식, 고석주(2005). 리눅스 기반 SCTP & TCP 성능 비교 분석. 2005. 5., http://protocol.knu.ac.kr/tech/CPL-TR-05-02.pdf에서 인출

Alamgir R., Ivancic W. (2005). Effect of Congestion Control on the Performance of SCTP and TCP in a Wireless Environment. *WSEAS TRANSACTIONS ON COMMUNICATIONS : Issue 6, Vol. 4,* 256-263.

Allman M., Floyd S., & Partridge C.(1998). *Increasing TCP's Initial Window.* RFC 2414.

Allman M., Floyd S., & Partridge C.(2002). *Increasing TCP's Initial Window.* RFC 3390.

Allman, M., Paxson, V., & Stevens, W.(1999). *TCP Congestion Control.* RFC 2581.

Donato E., Salvatore L., Pescape A., & Ventre G.(2006). Measuring SCTP Throughput and Jitter over Heterogeneous Networks. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications(AINA'06) : Vol. 2,* 395-399.

Natarajan, P., Iyengar, J. R., Amer, P. D. & Stewart, R.(2006). SCTP : An innovative transport layer protocol for the web. *15th International World Wide Web Conference,* Edinburgh, Scotland. 615-624.

Stevens W. (1997). *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.* RFC 2001.

Stewart, R., Arias-Rodriguez, I., Poon, K., Caro, A. & Tuexen, M. (2006). *Stream Control Transmission Protocol(SCTP) Specification Errata and Issues.* IETF RFC 4460.

Stewart R., Xie Q., Morneault K., Sharp C., Schwarzbauer H., Taylor T., Rytina I., Kalla M., Zhang L., & Paxson V.(2000), *Stream Control Transmission Protocol.* IETF RFC 2960.

◎ 논문접수: 2007. 10. 30 / 1차수정본 접수: 2007. 11. 29 / 게재승인: 2007. 12. 12

<Abstract>

Mean Transfer Time for SCTP in Initial Slow Start Phase*

Ju-Hyun Kim · Yong-Jin Lee*****

Stream Control Transmission Protocol(SCTP) is a transport layer protocol to support the data transmission. SCTP is similar to Transmission Control Protocol(TCP) in a variety of aspects. However, several features of SCTP including multi-homing and multi-streaming incur the performance difference from TCP.

This paper highlights the data transfer during the initial slow start phase in SCTP congestion control composed of slow start phase and congestion avoidance phase.

In order to compare the mean transfer time between SCTP and TCP, we experiment with different performance parameters including bandwidth, round trip time, and data length. By varying data length, we also measure the corresponding initial window size, which is one of factors affecting the mean transfer time.

For the experiment, we have written server and client applications by C language using SCTP socket API and have measured the transfer time by ethereal program. We transferred data between client and server using round-robin method.

Analysis of these experimental results from the testbed implementation shows that larger initial window size of SCTP than that of TCP brings the reduction in the mean transfer time of SCTP compared with TCP by 15 % on average during the initial slow start phase.

Key words : initial slow start phase, initial window, congestion control

---

* This work was supported by the Korea Research Foundation Grant funded by the Korea Government(KRF-2006-521-D00399).

** Juyop Technical High-school

*** Correspondence : email(lyj@knue.ac.kr), Korea National University of Education