Linux Kernel 2.6.X에서 SCTP API 분석

민경주* · 정옥조* ·강신각 *
*한국전자통신연구원

Analysis of SCTP API on Linux Kernel 2.6.X

Kyoung-Ju Min* · Ok-Jo Jung* · Shin-Gak Shin
*Electronic and Telecommunications Research Institute
E-mail : {meanrace, okjo, sgkang}@etri.re.kr

요 약

TCP/UDP가 차지하고 있던 전송 프로토콜에서 멀티미디어와 같은 대용량 데이터전송을 위해 개발되어 차세대 전송프로토콜로 지칭되는 SCTP가 최근의 리눅스 커널 2.6.0-testX에 탑재되었다. 리눅스 커널 2.6에 탑재된 SCTP는 lksctp라는 패키지 형태로 구현하고 배포하였는데, 실제 커널에서 SCTP API를 분석하고, TCP, UDP와의 비교가 시급한 실정이다. 또한 안정화 상태를 점검하고 검증할 가치가 있는 것으로 인식되고 있다. 이에 본 논문에서는 새로운 리눅스 커널에 탑재된 SCTP API를 분석하고, 테스트에 대한 구체적인 사항을 기술한다.

ABSTRACT

SCTP is a next generation transmission protocol that is more efficient for large scale data transmission like as multimedia data. SCTP protocol is supported in recent linux kernel 2.6.0-testX. SCTP which is supported in linux kernel 2.6 implemented in lksctp package styles. It is necessary that SCTP API is analyzed and is compared with TCP, UDP protocol. So, in this paper describes the analyzing and testing SCTP API that is supported in linux kernel 2.6.0-testX.

키워드
SCTP, Linux kernel 2.6, ngTCP, lksctp

1. 서 론

20년 이상 전송 프로토콜로 사용되던 TCP와 UDP는 사용하려는 응용에 따라 다소 불편이 있었던 것이 사실이다. 가령 TCP의 경우 커넥션 오리엔트드 방식이기 때문에 그 커넥션이 비정상적인 용인에 의해 종료 되었을때, 끝까지 종료하지 못한 데이터 전송에서 문제가 생기는 구조를 갖게 되었다. 이러한 단점을 보완하고 기존의 TCP, UDP의 장점을 동시에 살릴 수 있는 차세대 전송프로토콜의 필요성이 대두되면서 나온 것이 SCTP(Stream Control Transmission Protocol)[1,2]이다. SCTP는 'Super TCP[2]' 또는 차세대 TCP(ngTCP)[4]로 불려질 만큼 TCP의 개념을 수용하면서 단점을 보완한 프로토콜이다. 이러한 SCTP에 발맞추어 최근 개발되어 보급되기 시작한 Linux Kernel 2.6에서 SCTP가 기본 프로토콜로 자리매김하게 되었다. 현재까지의 커널 개발과 달리 이번에 출시된 커널 2.6에서는 release 버전이 아닌 테스트 버전이 나오면서 현재 2.6.0-test6까지 나와 있는 상황이다. 차후에 가장 많은 사용이 예상되는 전송 프로토콜이기 때문에, 이러한 커널을 설치하고, SCTP API를 검증하는 것은 필수적이라 할 수 있다. 이러한 이유로 본 논문에서는 새로운 커널의 설치와 SCTP API에 대한 테스트 및 검증에 관한 자세한 사항을 다루고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 SCTP의 기본 개념을 기술한다. 3장에서는 API를 테스트하고 검증하는 과정을 자세히 나룬다. 결론 및 향후 과제는 4장에서 다룬다.

2. SCTP 개요

SCTP의 기본 특징은 TCP의 특성과 비교되어 설명이 된다. 가장 대표적인 특징들은 다음과 같다. 첫째 TCP가 connection 개념을 사용하는 대신 SCTP는 association 개념을 사용한다. 둘째, TCP가 바이트 스트림 기반인데 반해, SCTP는 메시지 기반으로 동작을 한다. 셋째, Half-open 방식을 허용하지 않고, 쿠키 메커니즘을 사용함으로써, DoS (Denial of Service) 공격으로부터 보호된다. 넷째 앞서가던 패킷이 처리되지 않으면 뒤의 패킷도 처리할 수 없는 TCP의 방식과는 달리, SCTP는 스트림마다 독립적으로 처리되기 때문에 HOL(Head of Line) 블락킹이 없다. 다섯째 멀티호밍 방식으로 네트워크 라인에 문제가 있을 때 복구할 가능성이 커진다. 또한 흐름 및 혼잡 제어가 가능해진다[4,5].

SCTP의 특징을 살펴보면 다음과 같다.

2.1 멀티 스트리밍

SCTP의 멀티스트리밍 특성은 하나의 세션을 통해 다양한 종류의 응용 데이터를 보낼 수 있도록 한다. 세션 초기화 단계에서 송신자는 자신이 전송할 스트림의 개수를 수신자에게 통보하며, 전송 단계에서 각 스트림별로 독립적인 순서화(ordering) 기능이 제공된다. 그림 1과 같이 송신자와 수신자 사이에 n개의 스트림이 제공된다. 데이터 복구 및 재전송 과정 또한 스트림 ID별로 수행되어, 기존의 TCP에서 문제시 되었던, HOL 블락킹 문제를 해결한다. 이 방식은 어플리케이션이 동시에 여러 오브젝트(텍스트, 이미지, 오디오 등)를 전송할 수 잇도록 하며, 데이터 처리량을 향상시킨다.

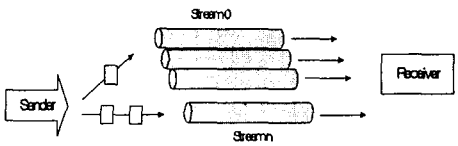

그림 1. SCTP의 멀티 스트리밍

2.2 멀티 호밍

SCTP의 멀티호밍 특성은 SCTP 세션이 여러 개의 IP 주소를 동시에 사용할 수 있도록 하며, 세션 도중 네트워크 장애가 발생하는 경우 대체 경로를 통해 세션이 유지되도록 한다. 세션 초기화 단계에서, 각 SCTP 종단은 가용한 IP 주소 목록을 교환하며, 이 주에 메인 IP 주소를 선정해 준다. 상대편 SCTP는 메인 주소를 수신주소로 하여 데이터를 전송하게 된다. SCTP 경로관리에 의해 메인 IP 주소에 이상이 있다고 판단되는 경우 SCTP는 다른 대체 IP 주소로 데이터 전송을 계속한다. 멀티 호밍의 Heartbeat 방식은 TCP의 keep-alive와 비슷한 개념으로 송신자가 수신자에게 주기적으로 heartbeat를 전송함으로써 활성 IP의 정보를 얻는다.

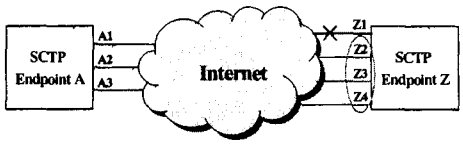

그림 2. 멀티 호밍

그림 2는 멀티호밍 방식을 보여준다. 송신자 A는 수신자 Z에게 데이터를 전송하기 위해 먼저 가용 IP 주소를 찾는다. 수신자 Z는 Z1~Z4까지의 주소 중 메인 주소인 Z1을 송신자 A에게 알려준다. 송신자 A는 A3와 Z1을 통해 통신하던 중 Z1 경로에 문제가 발생하면 나머지 가용 IP 주소를 찾는다. Z3가 활성 IP 주소라는 정보를 얻게 되면 송신자 A는 Z3를 통해 계속 통신을 하게 된다.

2.3 세션 초기화

초기화 과정에서 SCTP는 기존 TCP가 3-way handshake를 하던 것과는 달리, 4-way handshake 절차를 사용하여, DoS공격을 방어한다. 또한 verification tag와 COOKIE의 사용으로 블라인드 공격을 방어한다.

3. SCTP API의 테스트 및 검증

본 논문에서 설치하고 테스트한 환경은 다음과 같다. 리눅스 커널은 커널 2.6.0-test4에서 커널과 lksctp에서 제공하는 API를 테스트 하였다.

SCTP API의 검증을 위해서는 참고문헌 4의 내용을 충실히 테스트하여야 한다. 문서에서는 기본 오퍼레이션과 확장 오퍼레이션으로 구분하여 설명하고 있다. 기본 오퍼레이션으로 요구하는 것은, 서버를 위한 socket(), bind(), listen(), accept(), close() 와, 클라이언트를 위한 socket(), connect(), close()로 쉽게 구분할 수 있다. 뿐만아니라 데이터 전송과 수신을 위해 send(), recv(), sendto(), recvfrom(), read(), write()등도 TCP 스타일과 UDP 스타일에 따라 구분할 수 있다. 따라서 서버 클라이언트 프로그램에서 동작하는 것을 검증할 필요가 있다.

3.1 기본 오퍼레이션을 위한 서버-클라이언트

TCP의 기본 서버-클라이언트와 마찬가지로 SCTP를 이용한 파라미터를 이용하여, SCTP기반 서버-클라이언트 샘플을 만들어본 결과는 다음의 그림 3과 같다.

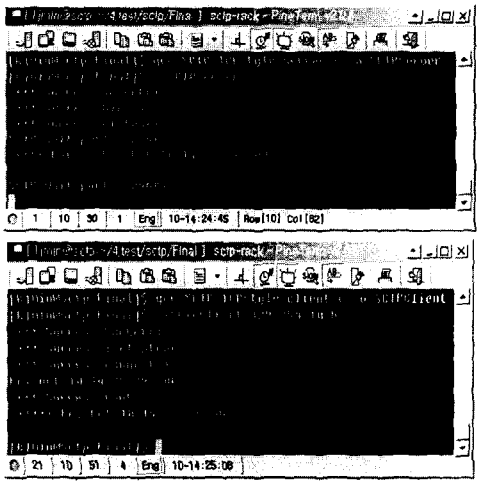

그림 3. 기본 SCTP 서버와 클라이언트

그림 3은 클라이언트가 서버에게 메시지를 전달하면 서버는 자신의 시간을 클라이언트에게 반환하는 간단한 샘플을 보인다. 이때 서버의 형태는 fd = socket(AF_INNET, SOCK_STREAM, IPPROTO_SCTP)와 같이 socket()을 열때 네 번째 파라미터에 프로토콜 타입이 SCTP임을 명시해주는 것을 제외하면 TCP 커넥션과 차이가 없고, IPPROTO_SCTP는 132번으로 정의 되었다.

3.2 확장 오퍼레이션 테스트 및 검증

확장 오퍼레이션을 위해 정의된 API는 sctp_bindx(), sctp_peeloff(), sctp_getpaddrs(), sctp_freepaddrs(), sctp_getladdrs(), sctp_freeladdrs(), sctp_sendmsg(), sctp_recvmsg(), sctp_connectx() 등이 있다. bindx는 기존의 bind이후에 다른 주소를 ADD하기 위한 API로 파라미터로 SCTP_BINDX_ADD_ADDR과, SCTP_ BINDX_REM_ADDR 두가지를 두어 주소를 ADD하거나 REMOVE할 수 있다.peeloff는 association에서 브랜치를 떼어내는데 사용하고, getpaddrs와 getladdrs는 peer address와 locally bound address에 대해 association에 있는 모든 해당 주소를 반환하는데 사용한다. 반대로 API앞에 free가 붙은 경우는 메모리 해제를 시키는데 사용한다. 나머지의 경우는 해당 API이름과 동일한 역할을 수행한다. 로컬 머신에서 간단히 테스트한 API들에 대한 결과를 보면 다음의 그림 4와 같다.

```
****TEST PRINT MESSAGE****

SNDRCV
sinfo_stream 2
sinfo_ssn 0
sinfo_flags 0x0
sinfo_ppid 1804289384
sinfo_context 0
sinfo_tsn 909145753
sinfo_cumtsn 0

SNDRCV
sinfo_stream 2
sinfo_ssn 1
sinfo_flags 0x0
sinfo_ppid 1804289384
sinfo_context 0
sinfo_tsn 909145754
sinfo_cumtsn 0

LOCAL ADDRESSES:
        127.0.0.1
PEER ADDRESSES:
        127.0.0.1
```

그림 4. SCTP를 이용한 정보들

SCTP를 이용하여 서버-클라이언트 프로그램을 통해 메시지의 전달과정을 Ethereal을 통해 검사한 결과가 다음의 그림 5에 나타나 있다. 서버-클라이언트 프로그램을 통해 메시지의 전달을 통해 INIT, INIT_ACK, COOKIE_ECHO 등에 대한 메시지 전달 과정을 볼 수 있다.

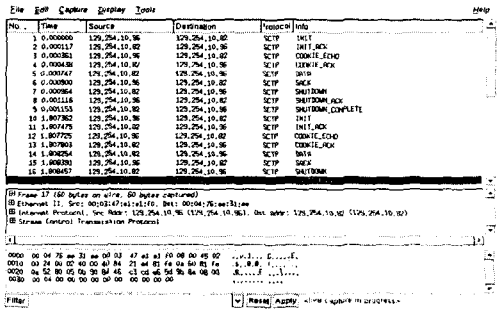

그림 5. Ethereal을 통해 본 SCTP 전달 과정

5. 결론 및향후 과제

지금까지 살펴본 바와 같이, SCTP 프로토콜은 TCP, UDP를 대체할 차세대 전송프로토콜로 인식되고 있다. 이러한 SCTP 프로토콜이 최근의 커널 2.6에 탑재되면서, SCTP의 활용도가 더욱 요구될 전망이다. 이에 커널에 올라있는 SCTP API를 테스트하고 검증하였다. 테스트와 검증을 통해 알 수 있듯이, SCTP 커널이 문제가 없는 것으로 보인다. 하지만 실제로는 커널 컴파일 과정에서 일부 문제점을 드러내고 있는 것이 사실이다. 따라서, 현재는 리눅스 커널이 보다 안정화 되고 테스트 버전이 아닌 릴리즈 버전으로 안정화되어야한다. 이를 기반으로 하여 대용량 데이터 전송이 요구되는 응용에 활용하면 그 성능 면이나 안정성면에서 우수할

것으로 예상된다.

향후 과제로는 이러한 SCTP 프로토콜과 함께, 커널이 안정화 된후 SIP 프로토콜등에 실제 활용함으로써 안정성을 실제 응용에서 검증하여야 한다.

참고 문헌

[1] RFC 2960, "Stream Control Transmission Protocol", draft-ietf-sctp-rfc2960.txt

[2] Radall R. Stewart, and Qiaobing Xie, "Stream Control Transmission Protocol (SCTP) : A Reference Guide", Addison-Wesley, 2002

[3] J.Rosenberg, H. Schulzrinne, G.Camarillo, "The Stream Control Transmission Protocol as a Transport for the Session Initiation Protocol", draft-ietf-sip-sctp-03.txt

[4] 고석주, 정회영, 민재홍, 박기식, "SCTP 표준 기술 분석 및 전망", 전자통신동향분석지 제 18권 제3호 , 20003.6

[5] 고석주, "SCTP:차세대 TCP", 한국전자통신연구원 정보화기술 동향분석지, 2003년 8월