무선 인터넷 환경에서 SCTP 프로토콜의 성능 최적화 방안

Performance Optimization of SCTP in Wireless Internet Environments

민경욱 · 고석주
Kyeong-Wook Min · Seok-Joo Koh

지금까지의 Stream Control Transmission Protocol(SCTP) 관련 연구는 주로 유선 인터넷 기반의 고정 단말 위주로 진행되어 왔으나 최근에는 인터넷 통신 환경이 휴대 단말 위주로 변화됨에 따라 모바일 단말에서의 SCTP 프로토콜 활용 및 성능 최적화에 대한 연구가 요구된다. 이에 본 논문에서는 안드로이드 기반의 무선 인터넷 환경에서 SCTP 프로토콜을 사용하는 방법에 대해서 기술하고 나아가 유무선 환경에서의 다양한 실험을 통해 TCP 프로토콜과의 성능을 비교 분석하고자 한다. 성능 비교 분석을 위한 환경 변수로는 Maximum Segment Size(MSS) 및 멀티스트리밍(multi-streaming)을 사용하는 경우 스트림(stream) 수를 사용하였다. 실험을 위해 테스트베드를 구축하고 실제 망과 유사하도록 패킷 손실이 있는 네트워크 환경을 구성하여 프로토콜의 전송 성능을 비교하였다. 실험 결과, 유선 환경에서는 SCTP가 무선 환경에서는 TCP가 전반적으로 우수한 성능을 보여주었다. 또한 무선 환경에서 패킷 손실률이 높아질수록 TCP의 성능은 떨어졌고 SCTP는 일정한 수준을 유지함을 확인하였다. 상기 실험 결과는 향후 모바일 단말에서의 SCTP 응용 개발시에 적정한 파라미터 설정에 참고로 활용될 수 있을 것으로 기대된다.

주제어: SCTP, TCP, Android, 모바일, 성능 비교

The existing works on Stream Control Transmission Protocol (SCTP) was focused on the fixed network environment. However, the number of smart phone users and services has increased, and thus the study on SCTP in the mobile network environment is crucially required. In this paper, we describe how to install the SCTP over Android platform for mobile devises and provide a comparative analysis for the performance of SCTP and TCP in the various fixed and wireless network environments. In the experiments, the following parameters are considered for performance comparison: Maximum Segment Size (MSS) and the number of streams in the SCTP multi-streaming case. A small-scale testbed is constructed for performance analysis, and some packet losses are generated to emulate the realistic network environment. From the experimental results, we can see that SCTP is better than TCP in the fixed network, whereas TCP seems to give better performance than SCTP in the wireless network. On the other hand, TCP performance is severely degraded when the packet loss rate increases in the network, however, SCTP tends to provide a consistent performance even in the network with some packet losses. It is expected that this study on SCTP performance optimization in the wireless networks can give useful information to the application developers using the Android platform.

Keywords: SCTP, TCP, Android, Mobile, Performance comparison

논문번호: TR12-107, 논문접수일자:2012.11.19, 논문수정일자:2013.04.05,논문게재확정일자:2013.04.22
민경욱: 삼성전자
고석주: 경북대학교

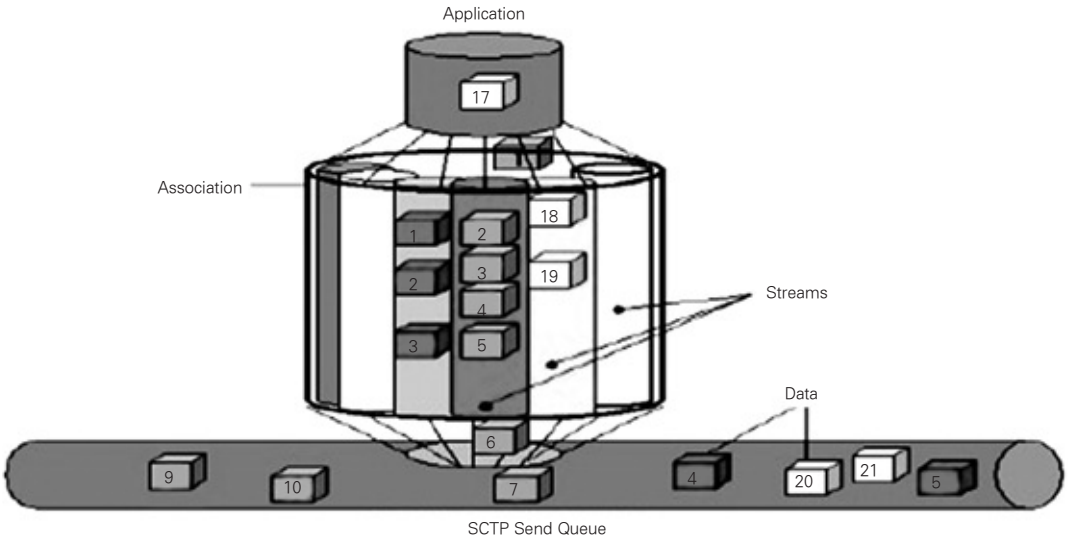


그림 1. SCTP 멀티스트리밍

I. 서 론

최근 스마트폰의 보급과 함께 모바일 기술이 급격하게 발전하고 있으며 사용자들은 유선 인터넷 환경에서 제공받던 서비스를 모바일 환경에서도 동일하게 제공받길 원하고 있다. 하지만 무선 네트워크 환경은 기존 유선 환경에 비해 높은 패킷 손실률, 핸드오버 발생 제한된 무선 자원 등의 제약조건을 지니고 있어 이에 대한 효과적인 대비가 요구된다.

한편 인터넷 표준화 기구인 Internet Engineering Task Force(IETF)에서는 TCP 및 UDP에 이어 세번째 수송계층 프로토콜로서 Stream Control Transmission Protocol(SCTP)을 표준으로 개발하였다[1]. 현재 SCTP 프로토콜은 인터넷 전화 서비스를 위한 시그널링 게이트웨이간 트래픽 전송, 멀티미디어 트래픽 전송 등의 실시간 신뢰 전송이 요구되는 응용에서 널리 사용되고 있다[2]. 또한 무선 단말의 핸드오버 지원을 위해 SCTP를 사용하는 연구도 진행되어 왔으며[3],[4] SCTP의 고유 특징인 멀티홈잉(multi-homing)을 활용하여 트래픽 전송의 효율성을 도모하는 연구도 이루어졌다[5],[6]. 아울러 SCTP의 멀티스트리밍(multi-streaming) 특성을 활용하여 음성, 영상, 텍스트 등의 멀티미디어 데이터를 미디어별로 다른 스트림을 사용하는 연구도 진행되고 있다[7]~[9].

이처럼 TCP에 비해 많은 장점이 있는 SCTP는 기존 PC 기반의 유선 네트워크 환경 뿐만 아니라 최근 급속도로 확산되고 있는 모바일 환경에서 더욱 나은 성능을 보여 줄 것으로 기대가 된다. 하지만 지금까지의 연구 개발 및 실험은 주로 PC 및 노트북 단말을 토대로 수행되어서 기존의 실험 결과들이 실제 안드로이드 혹은 iOS 등의 스마트폰 단말에 적용 가능한지 그리고 스마트폰 단말에 적용되었을 때 실제 PC 환경과 같은 성능을 제공할 수 있는지에 대한 연구는 거의 전무한 실정이다.

이에 본 논문에서는 최근 전세계 스마트폰 시장의 70% 이상을 차지하고 있는 안드로이드 플랫폼 기반의 스마트폰 단말기를 대상으로 SCTP 프로토콜을 설치하고 이를 활용하는 방법에 대하여 논의한다. 아울러 유무선 네트워크 환경에서 다양한 실험을 통해 TCP와 SCTP의 전송 성능을 비교 분석함으로써 스마트폰 단말 기반의 응용 프로그램 개발시에 적정한 파라미터 설정에 도움을 제공하고자 한다.

본 논문의 구성은 다음과 같다. II장에서 SCTP에 대한 개요와 관련 연구를 알아보고 III장에서는 안드로이드 플랫폼 환경에서 SCTP 프로토콜 모듈을 설치하고 활용하는 방법을 기술한다. 다음 IV장에서 유무선 네트워크 환경에서 TCP와 SCTP의 성능 분석을 위한 V장에서는 실험 결과를 토대로 유무선 환경에서의 두 가지 프로토콜의 성능을 비교 분석한다. 마지막으로 VI장에서 본 논문의 결론을 맺는다.

II. SCTP 개요 및 관련 연구

1. SCTP 프로토콜 개요

SCTP는 인터넷 망에서 시그널링 메시지 등의 시급하고 중요한 데이터를 전송하기 위해 개발된 프로토콜로써 UDP의 메시지 지향(message-oriented) 특성과

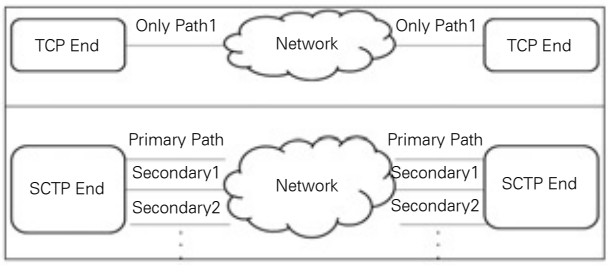


그림 2. SCTP 멀티홈잉

TCP의 연결 지향형(connection-oriented) 및 신뢰 전송 (reliable delivery) 특성을 조합하여 설계되었다. TCP와 구분되는 가장 큰 특징으로는 멀티스트리밍(multi-streaming) 기능과 멀티홈잉(multi-homing) 기능이 있다.

먼저 멀티스트리밍 특징에서는 하나의 연결에서 여러 개의 스트림을 동시에 독립적으로 전송할 수 있는 기능을 제공한다. 그림 1에서 보여지듯이 각각의 스트림에 아이디를 할당하고 스트림별로 신뢰성 있는 데이터를 전송함으로써 하나의 스트림 데이터 전송에 문제가 발생해도 다른 스트림에서는 정상적인 데이터 전송이 가능하도록 한다. 이를 통해 TCP의 HoLB(Head of Line Blocking) 문제를 방지할 수 있다.

다음으로 멀티홈잉 특징을 통해 종단 호스트에서 여러 개의 IP 주소를 동시에 사용할 수 있다. 그림 2에서 보여지듯이 TCP에서는 오직 하나의 IP주소만 사용하는 반면에 SCTP에서는 여러 개의 IP 주소를 사용한다. 실제 데이터 전송시에는 이 중 하나의 IP 주소만 우선경로(primary path)로 사용한다. 나머지 IP 주소는 보조경로(secondary path)로서 만약 우선 경로에 장애가 발생한 경우에 자동으로 보조 경로를 우선 경로로 변경하는 기능을 제공한다. 이를 통해 네트워크 장애가 발생했을 경우에도 데이터 전송을 지속할 수 있는 장점을 제공한다.

2. SCTP 관련 연구

SCTP 프로토콜이 표준으로 제정된 이후에 많은 연구가 수행되었으며 주요 결과를 정리하면 다음과 같다. 먼저 동일한 대역폭을 갖는 망에서 성능 분석 연구를 살펴보면 사용자 데이터 크기가 클수록 TCP에 비해 SCTP가 높은 처리율을 보인다[10]. 그리고 ECN(Explicit Congestion Notification) 기법을 이용하면 손실이 많이 발생하는 환경에서도 SCTP 성능을 향상시킬 수 있다는 결과도 제시되었다[11].

한편 멀티스트리밍 특성을 이용한 멀티미디어 데이터 전송에 대한 연구에서는 음성, 영상, 텍스트 등의 멀티미디어 데이터를 미디어별로 다른 스트림을 이용해 전송하는 것이 효율적이라고 실험을 통해 밝히고 있다[7]. 다음으로 멀티홈잉을 활용한 연구에서는 TCP Reno, TCP SACK, SCTP 방식의 성능을 시뮬레이션을 통해 비교하였다. 실험 결과 멀티홈잉 SCTP의 재전송 정책이 우선경로와 보조 경로간의 특성에 따라 성능이 달라짐을 보여주었다[12]. 아울러 멀티홈잉 특성을 이용한 무선 단말의 핸드오버 연구도 진행되었는데 하위 무선 링크계층의 신호세기를 기준으로 기존 IP 주소 삭제와 신규 IP 추가의 효율적인 타이밍에 대한 실험 분석 결과도 기존 연구에서 찾아볼 수 있다[13]~[15].

마지막으로 SCTP에서 발전된 Enhanced-SCTP에 대한 연구도 진행되었다[16]. 해당 연구에서는 패킷 에러율 및 지연 등의 실험조건을 달리하며 SCTP, PR-SCTP, Enhanced-SCTP에 대해서 다양한 비교 실험을 진행하였다. 실험 결과, 음성이나 비디오 같은 실시간 멀티미디어 스트리밍 서비스를 위해서 Enhanced-SCTP 기법이 효율적임을 보여주고 있다.

Ⅲ. 안드로이드 플랫폼에서 SCTP 프로토콜의 설치 및 활용

안드로이드 플랫폼에서 SCTP 응용 프로그램을 작성하기 위해서는 먼저 커널에서의 SCTP 프로토콜 지원과 플랫폼 차원의 SCTP 라이브러리 제공이 필요하다. 기본적으로 안드로이드는 리눅스 커널 2.6 이상을 사용하고 있기 때문에 커널 컴파일 시에 SCTP 관련 옵션을 추가해주면 SCTP 프로토콜 스택을 사용할 수 있다. 또한 리눅스 개발 환경에서 SCTP 응용 프로그램을 작성할 때 사용하는 lksctp[17] 툴을 안드로이드 플랫폼에 맞게 컴파일해서 필요한 라이브러리들을 제공할 수 있다.

1. SCTP 설치

안드로이드에서 SCTP 프로토콜 스택을 지원하기 위해서는 커널에 포함된 네트워크 설정 파일에 SCTP를 위한 옵션을 추가해 주고 커널 컴파일 시에 모듈로

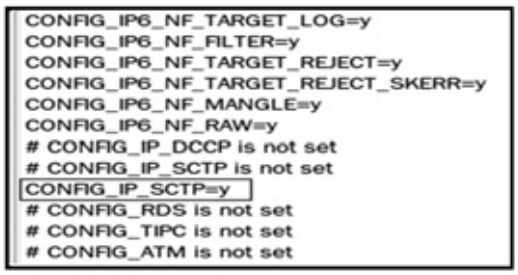


그림 3. 안드로이드 커널에 SCTP 프로토콜 스택 추가

그림 4. 안드로이드용 lksctp 라이브러리 생성을 위한 android.mk

생성해서 커널에 포함시키면 된다. 참고로 구글에서 제공하는 안드로이드 원본 소스에는 해당 옵션이 포함되어 있지 않다. 따라서 그림 3과 같이 수정해서 SCTP 프로토콜 스택을 추가할 수 있다.

또한 응용 프로그램을 작성하기 위해서는 SCTP 관련 라이브러리가 필요한데 리눅스에서는 이미 'SCTP for Linux Kernel'이라는 프로젝트가 진행 중이며 현재는 lksctp-2.6.28-1.0.10 패키지를 제공한다[17]. 하지만 lksctp 프로젝트는 리눅스 개발 환경을 위한 것이지 안드로이드 환경이 아니다. 따라서 해당 패키지를 안드로이드 환경에서 동작할 수 있게 변환 작업을 해야 한다. 변환 작업은 리눅스 환경에서 arm cross compiler로 컴파일하고 컴파일 과정을 분석한 후 lksctp 소스가 안드로이드에서도 컴파일 될 수 있도록 안드로이드 빌드 스크립터인 android.mk를 작성하는 것이 핵심이다[18],[19].

안드로이드용 lksctp 라이브러리 생성 과정은 다음과 같다. 먼저 리눅스용 lksctp 소스를 받고 arm cross-compiler를 준비한다. lksctp를 arm용으로 컴파일하기 위해서 configure 파일을 수정한다(configure-build=i386-linux -host=컴파일러 이름). 컴파일을 수행한 후 lkscp 소스가 안드로이드 환경에서 컴파일 될 수 있게 앞의 컴파일 과정을 분석해서 android.mk 파일을 작성한다. 즉 리눅스의 makefile 빌드 스크립트를 안드로이드에 맞는 android.mk 파일로 변환하는 과정이 필요하다. 그래야 안드로이드 풀(full) 빌드시 android.mk를 참조해서 lksctp를 빌드에 포함시키고 안드로이드에서 사용할 수 있는 라이브러리가 생성된다. 그림 4는 이에 대한 실행 화면을 보여준다.

그림 4에서 작성된 android.mk 파일과 lksctp 소스 파일을 안드로이드 소스의 android/external 위치에 복사하고 안드로이드 풀 빌드를 실행한다. android/external은 구글에서 제공하는 라이브러리 외에 필요한 라이브러리를 생성하는 장소이다. 풀 빌드가 성공적으로 끝나면/output 위치에 생성된 안드로이드 이미지를 단말기에 다운로드하고 그림 5처럼 Android SDK에서 제공해주는 툴을 통해 /system/libs에 lksctp 라이브러리가 생성되었는지 확인한다.

2. SCTP 활용

안드로이드 어플리케이션은 Java 언어로 작성이 되어야 한다. 하지만 lksctp는 C 언어로 작성된 헤더파일과 라이브러리를 제공한다. 따라서 어플리케이션은 Java로 작성을 하고 실제 lksctp 라이브러리를 사용하는 부분은 C 언어로 작성을 해야 한다. 그리고 그 둘 사이를 JNI(Java Native Interface)로 연결하는 작업이 필요하다.

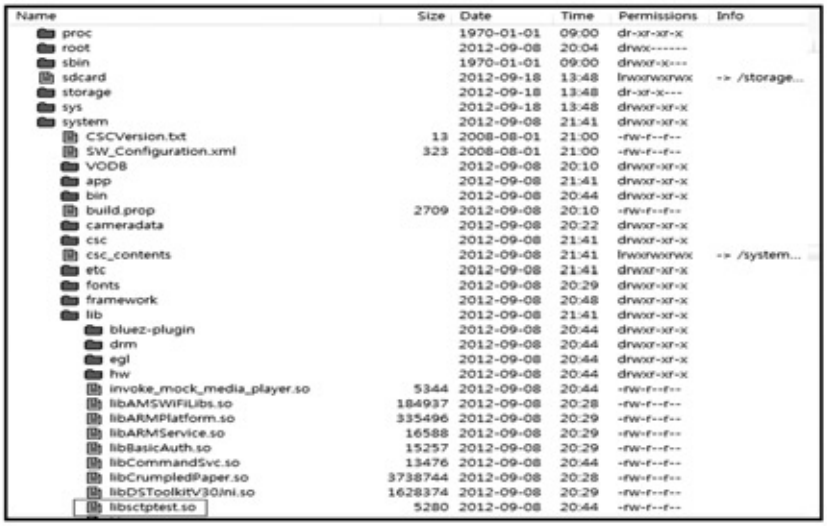


그림 5. 안드로이드용 lksctp 라이브러리 생성 확인

그림 6. SCTP 응용프로그램 작성 폴더 구조

본 절에서는 앞서 언급한 안드로이드용 lksctp 라이브러리를 이용해 윈도우 이클립스(eclipse) 개발환경에서 SCTP 안드로이드 어플리케이션을 작성하는 방법을 제시한다. 이를 위해 다음 3개의 파일이 사용된다. MainActivity.java는 메인 응용 프로그램이고 stcpclient.cpp는 lksctp 라이브러리를 이용해서 실제 소켓 통신을 수행하는 프로그램이다. 그리고 메인 응용 프로그램에서 C 함수들을 호출할 수 있게 JNI 기능을 제공하는 stcpclient.java가 있다.

이를 토대로 안드로이드 기반 SCTP 응용 프로그램 작성 순서를 정리하면 다음과 같다.

① sctpclient.java에서 native method를 정의한다.
② JDK에서 제공하는 javah를 이용해 native method의 헤더파일(sctp_client_sctpclient.h)을 생성한다.
③ 헤더 파일에 대응되는 sctpclient.cpp 파일을 작성한다.
④ android.mk 파일을 작성하고 ndk-build를 이용해 sctpclient.cpp를 컴파일해서 libsctpclient.so를 생성한다. 이때 sctpclient.cpp에서 lksctp 라이브러리를 사용하기 때문에 lksctptest.so과 sctp.h 파일이 필요하다. lksctptest.so는 SCTP 설치과정에서 생성된 파일로 단말기 연결을 통해/system/libs에서 로컬 PC로 다운이 가능하다.

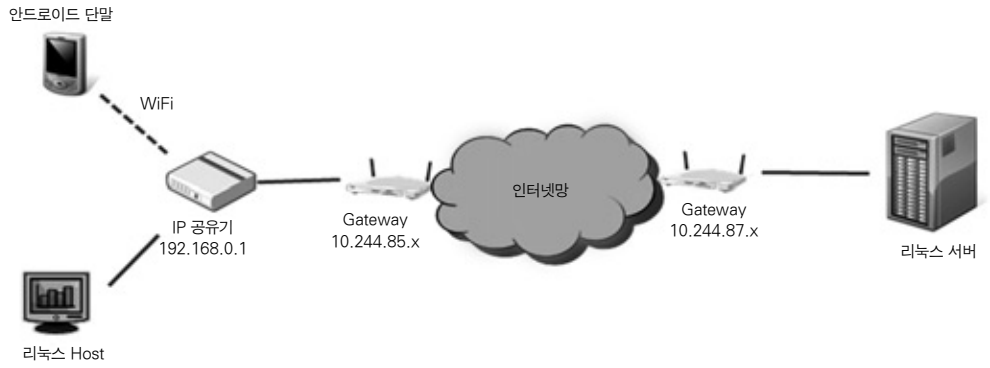


그림 7. 실험 환경 구성도

표 1. 안드로이드와 리눅스의 SCTP 개발 환경 비교

구분	리눅스	안드로이드
SCTP Protocol Stack	리눅스 설치시 옵션 추가 Networking->SCTP Configuration->SCTP Protocol-> y,(or m)	네트워크 설정파일에 CONFIG_IP_SCTP=y추가
SCTP Library	Lksctp RPM 설치	1. 리눅스용 lksctp 소스를 arm cross compiler 를 이용해 안드로이드용으로 컴파일 한 후 컴파일 과정을 분석함 2. 소스가 안드로이드 환경에서 빌드 될 수 있게 android.mk 빌드 스크립트를 작성함. 3. 소스와 빌드 스크립트를 /android/external에 위치시키고 안드로이드 full 빌드 후 플랫폼의 Library Layer에 포함시킴
SCTP Application	C, C++ 을 이용해 작성	대부분 Java로 작성하고 lksctp 라이브러리를 사용하는 부분은 C++로 작성 후 Java 와 JNI 로 연결하는 작업이 필요함.

⑤ 이후 sctpclient.java에서 libsctpclient.so 파일을 로드시켜서 응용 프로그램을 작성한다.

그림 6은 위에서 기술한 SCTP 응용 프로그램 작성 순서가 적용된 폴더 구조이다. 그림에서 com.example.sctpclient 패키지는 메인 응용 프로그램이고 sctp.client는 메인 응용에서 사용할 native method가 정의되어 있다. jni 폴더에는 native method가 직접 구현된 sctpclient.cpp가 위치하며 ndk-build를 이용해 sctpclient.cpp를 컴파일한 후 /libs/armeabi/libsctpclient.so를 생성한다. 그리고 sctpclient.cpp에서 실제 lksctp 라이브러리를 사용하는데 이를 위해 /netinet 폴더에 단말기에서 다운로드한 안드로이드용 lksctp 라이브러리를 위치시키고 /jni/android.mk에 lksctp 위치를 명시한 후(예: LOCAL_LDLIBS := -L./netinet/ -lsctptest) 사용하면 된다.

상기의 과정들을 통해 안드로이드 플랫폼에서 SCTP 프로토콜 기반의 응용 프로그램 작성이 가능하며 전체적으로 안드로이드와 리눅스에서의 SCTP 개발 환경을 비교하면 표 1과 같다.

IV. 실험 환경 및 시나리오

앞서 설치한 lksctp 라이브러리와 파일 전송 응용 프로그램을 이용해 TCP와 SCTP 프로토콜의 성능 분석 실험을 수행하였다. 본 논문에서는 무선 환경과 유선 환경에서 두 프로토콜 성능 비교 분석을 위해 안드로이드와 리눅스에 lksctp[17]를 설치하였고 데이터 전송 응용 프로그램을 이용하여 안드로이드와 리눅스 기반의 테스트베드를 구축하였다.

1. 실험 환경

SCTP와 TCP의 성능 비교 분석을 위해 파일 수신

표 2. MSS 크기에 따른 성능 비교를 위한 실험 설정값

구분	TCP	SCTP
MSS 크기	1460, 730, 365 byte	1460, 730, 365 byte
스트림수	Default - 1개	Default - 1개
패킷 손실률	Default - 0%	Default - 0%

표 3. SCTP 스트림수에 성능 비교 실험 설정값

	TCP	SCTP
MSS 크기	-	365 byte
스트림수	-	2개, 5개, 10개
패킷 손실률	-	Default - 0%

표 4. 패킷 손실률에 성능 비교 실험 설정값

	TCP	SCTP
MSS 크기	1460 byte	365 byte
스트림수	1개	5개
패킷 손실률	0%, 10%, 20%	0%, 10%, 20%

서버와 유선망 클라이언트에는 각각 ubuntu linux 10.04를 사용하고 무선망 클라이언트는 android ICS 4.0.3 버전을 사용하였다. 우선 안드로이드용 lksctp 라이브러리가 포함된 이미지를 단말에 다운로드 한다. 그리고 파일 전송 실험을 위해서 소켓 응용 프로그램을 설치한다. 또한 유선 환경 실험을 위해 리눅스 단말을 준비하고 lksctp와 파일 전송 응용 프로그램을 설치한다. 실험 서버도 lksctp를 설치하고 각 클라이언트에게서 파일을 수신받는 소켓 응용 프로그램을 설치한다. 아울러 수신된 패킷을 분석할 수 있는 Wireshark[20]도 준비한다. 안드로이드 단말과 리눅스 Host가 연결되어 있는 IP 공유기에서 MTU(Maximum Transmission Unit) 크기를 1500 byte로 고정한다. 이는 대부분의 망에서 기본적으로 사용되는 값이다.

그림 7은 실험 환경 구성도이다. 리눅스 서버, 리눅스 Host, 안드로이드 단말로 이루어져 있으며 파일을 전송하는 리눅스 Host와 안드로이드 단말은 하나의 IP 공유기에 연결되어 있다. 그리고 무선 환경 실험을 위해 안드로이드 단말과 IP 공유기는 서로 WiFi 통신을 하고, 파일 발신측 네트워크와 수신측 네트워크는 인터넷 망으로 연결되어 있다.

2. 성능 분석 시나리오

안드로이드 단말과 리눅스 Host에서 각각 100MB 텍스트 파일을 리눅스 서버로 전송하고 초당 전송량을 측정하였다. 이때 성능 기준으로는 패킷의 세그먼트 크기를 지정하는 MSS(Maximum Segment Size)와 SCTP의 스트림수를 사용하였다. 그리고 실험 결과 최적의 성능 비교 값들을 추출하여 실제 패킷 손실이 존재하는 망에 적용함으로써 TCP와 SCTP, 그리고 유선망과 무선망에서의 성능을 비교 분석하였다. 각 5회씩 실험을 수행하고 평균값을 성능 비교 수치로 사용하였다.

실험에 적용한 3가지 시나리오를 정리하면 다음과 같다.

2.1. MSS 크기에 따른 성능 비교

SCTP는 하나의 패킷에 여러 개의 데이터 청크를 포함 할 수 있기 때문에 MSS 값의 변화에 따라 프로토콜의 패킷수가 변하게 될 것이다. 기본적으로 MSS 크기는 MTU 크기보다 40 byte가 작은 1460(1500-40) bytes로 설정된다. 이 값을 1460, 730, 365 로 변경하면서 MSS 크기가 프로토콜 성능에 미치는 영향을 분석해본다. MSS 크기에 따른 성능 비교에 사용된 설정값을 정리하면 표 2와 같다.

2.2. SCTP 스트림수에 따른 성능 비교

본 실험에서는 스트림수가 SCTP 프로토콜 성능에 미치는 영향을 분석한다. 기본적인 설정값은 앞서 실험에서 추출한 최적의 값을 적용하고 스트림수를 10, 5, 2개로 변화시키면서 실험을 진행한다. SCTP 스트림수에 따른 성능 비교에 사용된 실험 설정값을 정리하면 표 3과 같다.

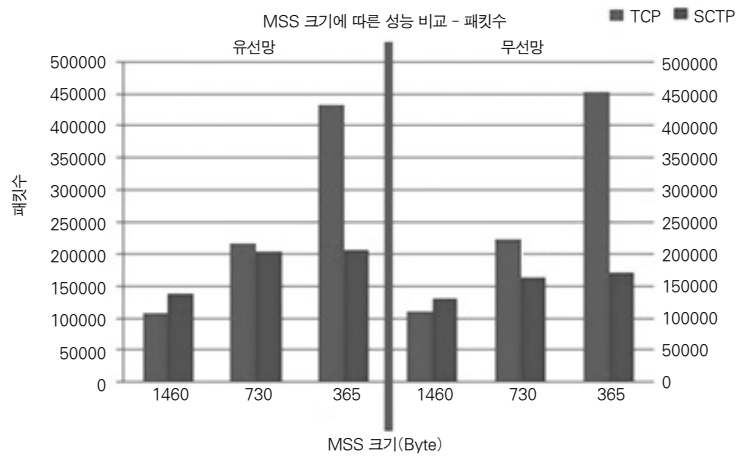


그림 8. MSS 크기에 따른 성능 비교 - 패킷수

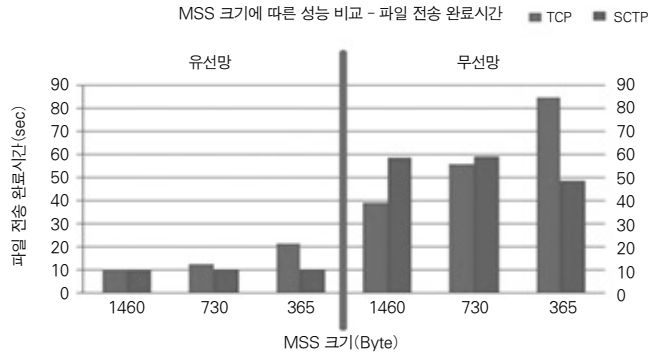


그림 9. MSS 크기에 따른 성능 비교 - 파일 전송 완료시간

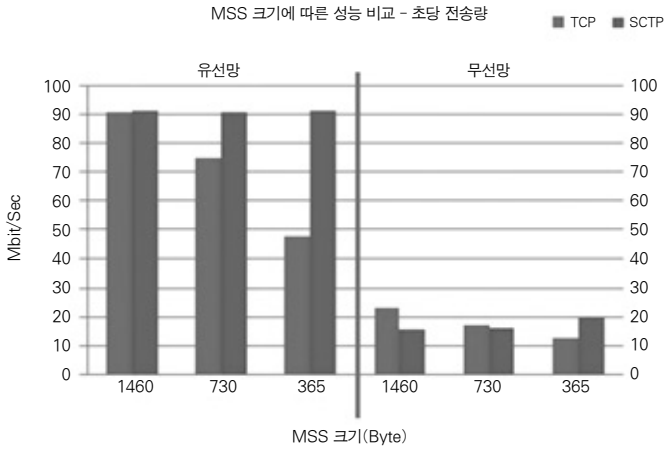


그림 10. MSS 크기에 따른 성능 비교 - 초당 전송량

2.3. 패킷 손실률에 따른 성능 비교

무선망의 경우 비트 에러율, 채널 간섭 및 대역폭 등의 채널 환경에 따라 상당한 양의 패킷 손실이 발생할 수 있다. 본 실험에서는 무선 채널 환경으로 인해 발생하는 패킷 손실률을 0%, 10%, 20%로 설정하고 성능 결과를 분석해본다. 성능 비교에 사용된 실험 설정값을 정리하면 표 4와 같다.

V. 실험 결과 분석 및 성능 비교

그림 8, 9, 10은 MSS 크기에 따른 성능 비교로서 생성된 패킷수와 파일 전송 완료시간 및 초당 전송량을 보여준다. MSS 크기를 50%씩 줄이면 TCP 패킷수도 50%씩 늘어남을 알 수 있다. TCP 헤더의 크기가 20 byte인점을 감안하면 (늘어난 패킷수 * 20바이트)의 오버헤드가 생기게 된다. 이로 인해 전체 전송해야 하는 데이터가 많아지고 초당 전송량도 현저히 떨어지게 된다.

반면에 SCTP의 경우에는 패킷수가 완만히 늘어나는데 이는 SCTP의 패킷 번들링(bundling) 효과에 의한 것이다. MSS가 1460일 경우 패킷수가 가장 적다. MSS가 줄어들 경우 SCTP는 MSS 크기의 데이터 청크(chunk) 2개 이상을 하나의 패킷으로 번들링하기 때문에 전체 패킷수와 초당 전송량에 급격한 변화는 나타나지 않는다. 그리고 초당 전송량 측면에서 TCP는 유, 무선망간에 차이가 없다. 반면에 SCTP의 경우 유선에서는 MSS 크기의 변화와는 관계없이 일정한 전송량을 유지하지만 무선에서는 MSS 크기가 작아질수록 초당 전송량이 증가하는 결과를 볼 수 있다. 유선망에 비해 상대적으로 낮은 품질을 가지는 무선망에서는 MSS를 크게 설정하고 데이터를 전송하는 도중에 오류를 만났을 때는 재전송해야 하는 데이터가 크지만 MSS를 작게 설정하면 작은 데이터만 재전송하면 된다. 따라서 무선 환경에서는 MSS 값을 작게 설정하는 것이 유리하다. 아울러 그림 10의 파일 전송 완료시간 실험 결과를 보면 초당 전송량 결과와 동일한 경향을 볼 수 있다. 이는 초당 전송량이 많을수록 파일 전송 완료시간도 빠르고 초당 전송량이 적을수록 전체 파일을 전송 완료하는데 시간이 많이 걸린다는 것을 보여준다. 결론적으로 MSS 크기가 작아질수록 TCP의 전송 효율은 떨어지는 반면에 SCTP는 원래의 효율을 일정하게 유지하거나 무선 환경에서는 오히려 효율이 좋아지는 것을 알 수 있다.

그림 11, 12는 MSS=365 바이트를 적용하고 스트림 개수를 10, 5, 2개로 변화시키며 실험한 결과이다. 스트림 개수를 5개로 설정했을 때 유/무선 모두 전송 효율이 가장 좋다. 하지만 그 이상이 되면 효율은 떨어진다. 이것은 여러 스트림으로 패킷이 전송되기 때문에 수신 측에서는 다시 패킷을 재배치해야 되는데 이때 생기는 패킷 재배치 오버헤드 때문인 것으로 풀이된다. 하지만 무선의 경우에는 망 자체의 속도가 늦기 때문에 패킷 재배치의 오버헤드가 줄어들어 10개 스트림과 5개 스트림의 경우 모두 비슷한 효율을 보인다. 다시 말해 유선의 경우 패킷 재배치 속도보다 수신측 큐에 데이터 쌓이는 속도가 빠르기 때문에 오버헤드가 많이 생기지만 유선망보다 상대적으로 느린 무선망에서는 패킷 재배치 속도와 데이터 쌓이는 속도간 차이가 매우 작기 때문에 유선망에서 실험 결과만큼 오버헤드가 발생하지 않는 것이다.

그림 13은 무선망에서 채널 환경으로 인한 패킷 손실률을 0%, 10%, 20%로 변경하면서 초당 전송량을 비교한다. 그림에서 알 수 있듯이 기본적으로 손실률이 0%일 때 TCP가 좋은 전송률을 보여준다. 하지만 손실률이 증가할수록 TCP의 성능은 떨어지는데 비하여 SCTP는 일정한 전송 효율을 유지하고 있다. 또한 패킷 손실률이 10%를 초과하는 네트워크 환경에서는 TCP 보다 SCTP가 더 좋은 성능을 제공하고 있다. 이는 손실률이 높은 망에서 SCTP의 Selective ACK 기법을 통해 오류 탐지 및 복구를 효율적으로 수행하기 때문이다. 결과적으로 손실률이 존재하는 망에서는 TCP에 비해 SCTP가 더 안정적이라고 할 수 있다.

그림 14는 무선망에서 패킷 손실률 별로 SCTP 스트림 수에 따른 초당 전송량을 보여준다. TCP는 스트림 1개를 사용한 것에 해당한다. 그림에서 패킷 손실률이 증가할수록 TCP는 성능이 급격히 떨어지고 SCTP는 일정한 수준을 유지하거나 오히려 좋아지는 모습을 볼 수 있다. 또한 패킷 손실률이 0~20% 상태의 망에서 SCTP 프로토콜을 사용할 경우 스트림 5개를 사용하는 것이 최적의 성능을 제공하고 있다. 이를 통해 무선 네트워크 환경에 따라 적정한 스트림 개수가 존재함을 할 수 있다. 특히 그림 14(c)에서 보여지듯이 패킷 손실률이 상대적으로 높을 때에는 SCTP가 TCP 성능을 능가할 수 있음을 알 수 있다.

VI. 결 론

지금까지 안드로이드 플랫폼 기반의 모바일 단말에서 SCTP 프로토콜을 사용하는 방법에 대해서 기술하고 유무선 환경에서의 다양한 실험을 통해 TCP 프로토콜과의 성능을 비교 분석하였다. 실험 결과 유선 환경에서는 SCTP가 무선 환경에서는 TCP가 전반적으로 우수한 성능을 보여주었다. 또한 무선 환경에서 패킷 손실률이 높아질수록 TCP의 성능은 떨어졌고 SCTP는 일정한 수준을 유지함을 확인하였다. 그리고 손실률이 증가하는 무선 환경에서 TCP는 효율이 급격히 떨어지는 반면에 SCTP는 일정하고 안정적인 전송 효율을 제공하며 스트림수에 따라 TCP보다 더 좋은 성능을 제공한다. 따라서 안드로이드 단말을 사용하는 경우에 어느

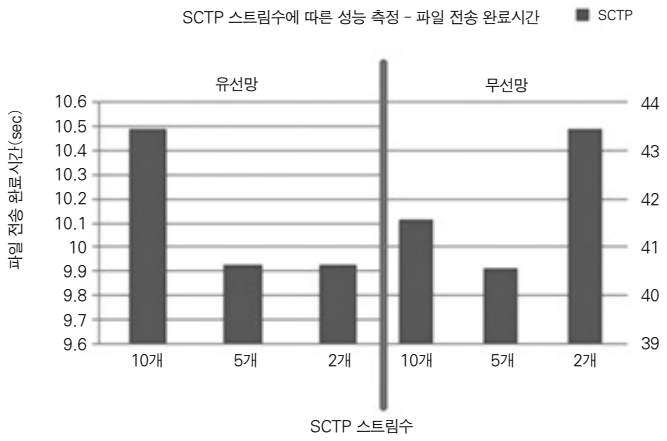


그림 11. SCTP 스트림수에 따른 성능 측정 - 파일 전송 완료시간

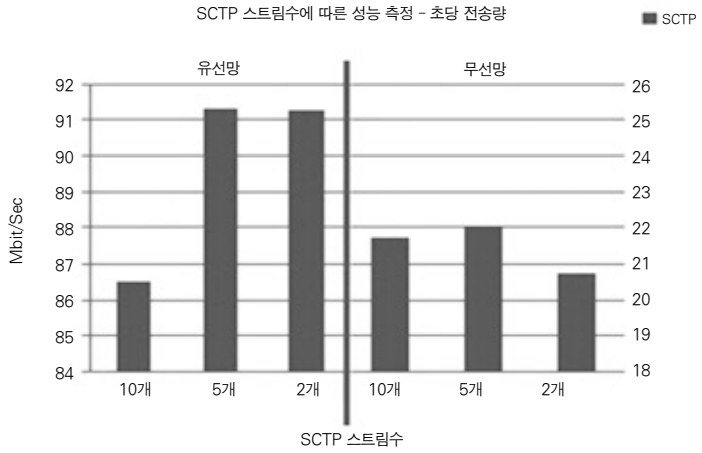


그림 12. SCTP 스트림수에 따른 성능 측정 - 초당 전송량

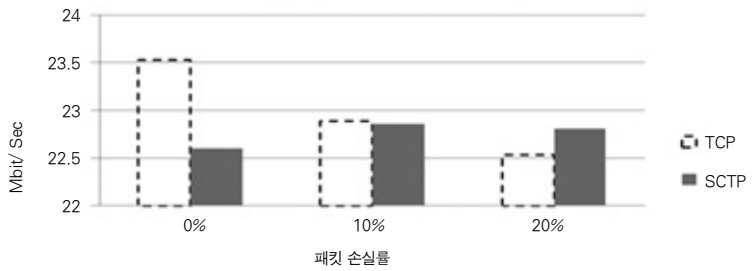
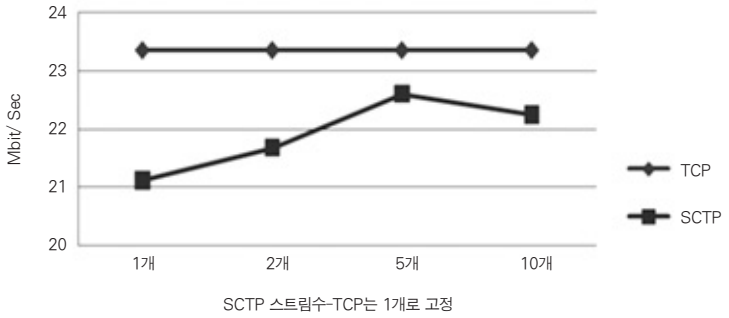
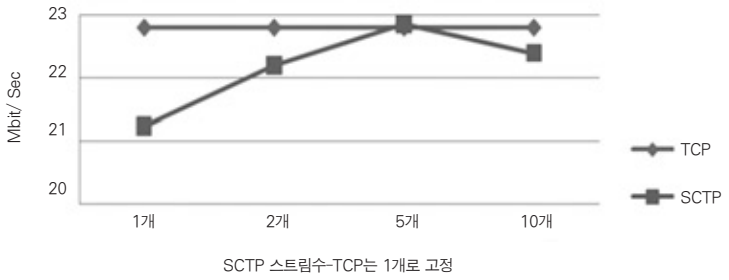


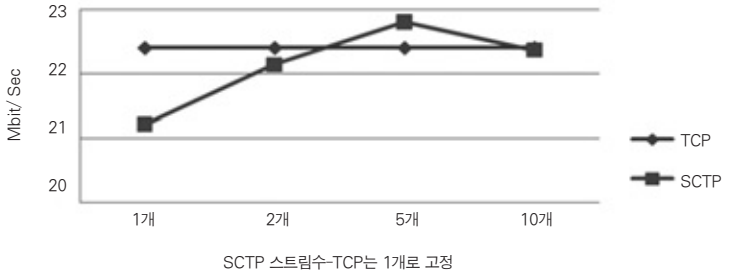
그림 13. 무선망에서 패킷 손실률에 따른 성능비교 - 초당 전송량



(a) 패킷 손실율 0%



(b) 패킷 손실율 10%



(c) 패킷 손실율 20%

그림 14. 무선망에서 패킷 손실률과 스트림수에 따른 성능 비교 - 초당 전송량

정도 손실률이 존재하는 무선 환경에서는 안정적인 통신에 SCTP가 좀 더 효율적일 수 있음을 알 수 있다. 상기 결과는 향후 모바일 단말에서의 SCTP 응용 개발 시에 적정한 파라미터 설정에 참고 자료로 활용될 수 있을 것으로 기대된다.

감사의 글

본 논문은 한국연구재단의 기초연구사업(NRF-2010-0020926)과 미래창조과학부 및 정보통신산업진흥원의대학IT연구센터지원사업(NIPA-2013-H0301-13-2004)의 지원으로 수행되었음.

〔참고문헌〕

[1] R. Stewart, et al., *Stream Control Transmission Protocol*, IETF RFC 4960, Sep. 2007.

[2] J. S. Ha, et al., "Performance Comparison of SCTP and TCP over Linux Platform," *Lecture Note in Computer Science*, Vol. 3645, Aug., 2005, pp. 396-404.

[3] D. P. Kim, et al., "Performance Enhancement of mSCTP for Vertical Handover across Heterogeneous Wireless Networks," *International Journal of Communication Systems*, Vol. 22, No. 12, Dec. 2009, pp. 1573 - 1591.

[4] D. P. Kim, et al., "mSCTP-DAC: Dynamic Address

Configuration for mSCTP Handover," *Lecture Note in Computer Science*, Vol. 4096, Aug. 2006, pp. 244 - 253.
[5] 최용운 외, "링크다운 환경에서 TCP에 대한 SCTP의 멀티호밍 효과", *한국콘텐츠학회논문지*, 제9권 제8호, 2009년 8월, pp. 73-83.
[6] 최용운 외, "링크다운 시간에 따른 TCP와 SCTP의 웹 트래픽 분석", *한국콘텐츠학회논문지*, 제10권 제3호, 2010년 3월, pp. 44-52.
[7] 민경주 외, "SCTP를 이용한 멀티미디어 품질 향상 방법", *정보과학회 추계학술대회 발표논문집*, 제30권 제2호, 2003년 10월, pp. 280-282.
[8] 이용진, "무선 인터넷 환경에서 HTTP over SCTP의 평균 응답시간 추정", *한국콘텐츠학회논문지*, 제8권 제6호, 2008년 6월, pp. 43-53.
[9] 강기철 외, "SCTP 기술을 이용한 효율적인 DLNA 미디어 스트리밍 환경 구축에 관한 연구", *한국방송공학회 하계학술대회 발표논문집*, 2011년 7월, pp. 212-215.
[10] A. Jungmajer, et al, "Performance Evaluation for the Stream Control Transmission Protocol," *Proceeding of the Joint ATM Workshop*, Jun. 2000, pp. 141-148.
[11] Guanhua Ye, et al., "Improving Stream Control Transmission Protocol Performance Over Lossy Links," *IEEE Journal on Selected Areas in Communications*, Vol. 22, Issue 4, May 2004, pp. 727-736.
[12] 송정화 외, "SCTP의 멀티홈잉 특성에 대한 성능 평가", *정보처리학회논문지*, 제11-C권 제2호, 2004년 4월, pp. 245-252.
[13] 장문정 외, "mSCTP를 이용한 종단간 이동성 지원 방안," *정보과학회논문지:정보통신*, 제 31권 제 4호, 2004년 8월, pp. 393 ~ 404
[14] M. Chang, et al., "Transport Layer Mobility Support Utilizing Link Signal Strength Information", *IEICE Transactions on Communications*, Vol. E87-B, No. 9, Sep. 2004, pp. 2548-2556.
[15] M. Chang, et al., "A Transport Layer Mobility Support Mechanism", *Lecture Note in Computer Science*, Vol. 3090, May 2004, pp. 287 - 296.
[16] 김경회, "Multimedia transmission protocols for distributed MAC and enhanced SCTP", *고려대학교 대학원: 전자컴퓨터공학과*, 2010년 8월
[17] Linux Kernel SCTP, Available from http://lksctp.sourceforge.net.
[18] Android Developer, Available from http://developer.android.com
[19] Android Platform, NDK, Compile, Available from http://www.aesop.or.kr
[20] Wireshark, Available from http://www.wireshark.org.

민 경 욱
(Kyeong-Wook Min)

2007: 동서대학교 컴퓨터공학과 학사
2013.2: 경북대학교 컴퓨터학부 석사
2007~현재: 삼성전자 무선사업부 책임연구원
관심분야: 통신 프로토콜, SCTP
E-mail: k.w.min@samsung.com

고 석 주
(Seok-Joo Koh)

1988~1992: KAIST 경영과학과 학사
1992~1994: KAIST 경영과학과 석사
1994~1998. 8: KAIST 산업공학과 박사
1998. 8~2004. 2: ETRI 표준연구센터 선임연구원
2004. 3~현재: 경북대학교 IT대학 컴퓨터학부 교수
2000~현재: ITU-T SG13 및 ISO/IEC JTC1/SC6 Editor
관심분야: 미래 인터넷, 이동성 관리, SCTP, 멀티캐스트, 국제 표준화
E-mail: sjkoh@knu.ac.kr