SCTPLIB 기반 mSCTP 핸드오버 설계서

2006년 8월

경북대학교 통신프로토콜연구실

이동화 (yiffie9819@gmail.com)

요 약

본 문서는 RAW socket을 이용하여 user level에서 구현한 SCTP STACK인 SCTPLIB와 그 추가 확장 규격인 Dynamic Address Reconfiguration(ADD-IP)을 이용하여 수송 계층에서 IP의 이동성을 제공함으로써 이 기종 망에서의 끊김 없는 핸드오버를 지원할 수 있는 시스템을 설계함이 그 목적이다.

목 차

1. 서론

본 문서는 RAW socket을 이용하여 user level에서 구현한 SCTP STACK인 SCTPLIB와 그 추가 확장 규격인 Dynamic Address Reconfiguration(ADD-IP)을 이용하여 수송 계층에서 IP의 이동성을 제공함으로써 이 기종 망에서의 끊김 없는 핸드오버를 지원할 수 있는 시스템을 설계할 수 있음이 그 목적이다. 기본적으로 SCTPLIB의 모든 API는 RFC 2960의 10장 "Interface with Upper Layer"의 내용에 따라서 크게 ULP-to-Upper 인터페이스와 SCTP-to-ULP Notification, 보조함수로 구분되어서 작성이 되었으며, 각 함수가 사용하는 인자들은 부가적인 설명이 없이도 개발자가 쉽게 이해할 수 있게 설계되었다. 이동성 지원을 위해 추가된 함수 역시 기존 SCTPLIB의 철학을 따라서 사용자에게 친숙한 형태를 가지고 있다. 본 문서에서는 SCTPLIB 기반 mSCTP 핸드오버 시스템의 구조에 대해서 설명하고 그 시나리오에 대하여 분석하도록 한다.

2. SCTPLIB 기반 IP 이동성 지원 모델

SCTP는 TCP/UDP에 이은 3번째 수송계층 프로토콜로써 종단간 데이터 신뢰전송 기능을 제공한다. 주요 특징으로는 '멀티스트리밍(multi-streaming)' 및 '멀티홈잉(multi-homing)' 기능이 있다. 먼저 SCTP 멀티스트리밍을 통해 하나의 세션에서 여러 개의 응용 스트림을 식별하여 전송할 수 있으며, 멀티호밍 특성을 통해 여러 개의 IP 주소를 SCTP 세션에 바인딩 할 수 있다.

그림 1은 SCTP 멀티호밍 개념을 보여준다. SCTP는 등록된 여러 IP 주소 중의 하나로 데이터를 전송할 수 있다. 특히, 최근에 개발중인 '동적 IP 주소설정(Dynamic Address Configuration)' 방식에서는, 세션 도중에 새로운 IP 주소를 추가하고(Add-IP), 주요 데이터 전송 경로를 변경하거나(Primary-Change), 기존 IP 주소를 삭제하는(Delete-IP) 기능이 추가되었다.

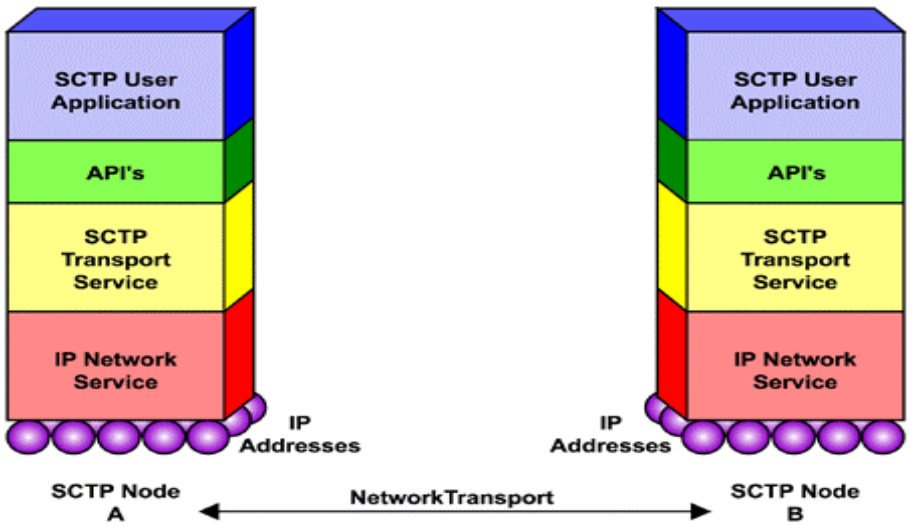

그림 1. SCTP 멀티홈잉

SCTP 이동성 기법은 상기와 같은 '멀티홈잉' 특성 및 '동적 IP 주소설정' 기능을 활용하여, 이동단말의 IP 주소 변경 시에 핸드오버를 지원하는 기술이다.

2.1 SCTP 이동성 기법

SCTP 주요 확장 기능으로 ADD-IP 확장을 들 수 있다. SCTP 기본 규격에서는 세션에 사용되는 IP 주소를 세션초기화 단계에서만 지정하도록 되어있었으나, 본 확장 규격에서는 세션 도중에도 신규 IP 주소를 세션에 등록하거나 혹은 삭제하는 기능을 제공한다. 또한 세션 도중에 primary IP 주소를 변경하는 기능도 포함한다.

이러한 SCTP 세션에 대한 IP 주소의 재구성이 필요할 경우, 해당 SCTP는 관련 주소 정보를 ASCONF (Address Configuration Change) 제어 Chunk에 실어 상대방에 전송하며, 상대방은 ACSONF-ACK Chunk로 응답할 수 있다.

위 기능은 특히 이동단말이 세션 도중에 다른 IP 망으로 이전하게 되는 경우에, seamless handover 기능 지원을 위해 필수적으로 요구되는 사항이다. 특히 기존의 Mobile IP 경우, 핸드오버 기능이 취약하였으나 SCTP를 통해 핸드오버 기능이 상당히 개선될 것으로 기대된다. 한편 완전한 IP 이동성 지원을 위해서는 핸드오버와 함께, IP 단말에 대한 위치관리 기능이 제공되어야 하며, 이를 위해 Mobile IP의 위치등록 기능이 함께 사용될 수 있을 것으로 전망된다. 본 절에서는 ASCONF 기능을 활용한 seamless handover 절차에 대해 기술한다.

그림 2에 보여지듯이, 이동 단말이 SCTP를 사용하며 세션 도중에 IP 지역을 바꾸는 경우 핸드오버 절차는 다음과 같다.

(1) 먼저 그림에서처럼, 세션 초기화 단계에서 이동단말(Mobile Node, MN)은 주소 2를, 상대단말(Correspondent Node, CN)은 주소 1를 사용하여 SCTP 세션을 설정하였다고 가정한다.

(2) MN이 다른 IP 영역으로 이동하는 경우, 중첩지역(overlapping region)에서 신규 주소 3을 하위 네트웍 계층으로부터 받게 되면, 이를 ASCONF chunk를 통해 CN 에게 통지한다. 이를 통해 MN은 dual-homing 상태가 되며, CN으로부터의 데이터를 주소 2 뿐만 아니라 주소 3을 통해서도 받을 수 있게 된다.

(3) MN은 무선계층의 신호세기에 따라 primary 주소를 주소 3으로 변경할 수 있다.

(4) MN이 중첩지역을 벗어나는 경우 ASCONF를 통해 기존 주소 2를 SCTP 세션에서 삭제할 수 있다.

(5) 이러한 절차가 SCTP 세션 도주에 IP 지역을 바꿀 때 마다 되풀이 된다.

SCTP는 이동단말 MT의 이동으로 IP 주소가 변경된 경우에도, 바뀐 주소를 세션에 바인딩함으로써 세션이 중단되지 않는 핸드오버 기능을 제공한다. 그림 2는 SCTP 핸드오버 모델을 보여준다.

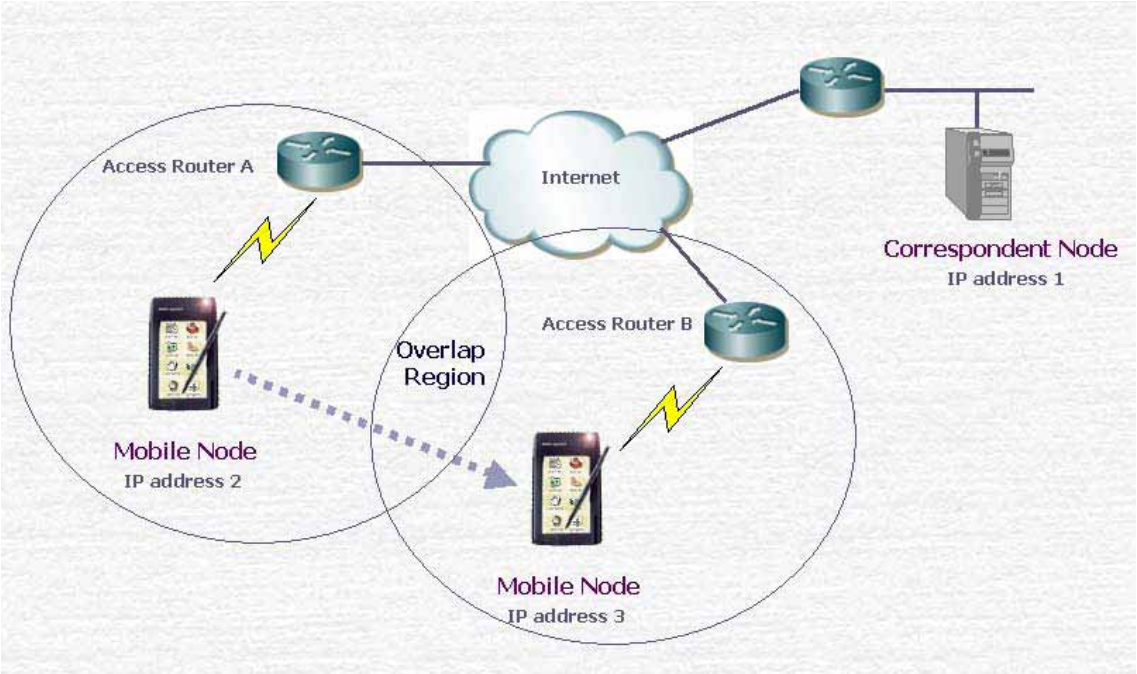

그림 2. SCTP 핸드오버

2.2 SCTP 핸드오버 시나리오

상기 핸드오버 알고리즘은 네트워크 환경에 따라 다음 두 가지 시나리오에 적용될 수 있다.

A. 시나리오 1 (그림 3(a) 참조): Dual-homing MT

3G-WLAN 혹은 3G-WiBro 연동처럼 이종망간 'Vertical Handover' 상황에 적용될 수 있다. MT 단말에 두 개의 NIC가 탑재되고, 핸드오버 동안에 동시에 두 개의 NIC가 활성화된다. 즉, MT는 dual-homing 상태에서 패킷 송수신을 수행한다.

B. 시나리오 2(그림 3(b) 참조): Single-homing MT

3G 혹은 WiBro 등의 동일 망에서의 'Horizontal Handover' 상황에 적용될 수 있다. MT는 한 순간에 하나의 NIC 및 IP 주소를 사용하며, single-homing 상태에서 패킷 송수신을 수행한다.

그림 3에서처럼 '시나리오 1'에서는 중첩영역에서 두 개의 NIC가 동시에 활성화될 수 있는 이종망간 Vertical 핸드오버를 가정한다. 반면에 '시나리오 2'에서는 동일망간 Horizontal 핸드오버를 가정하고, 한 순간에 NIC를 통해 하나의 IP 주소만 사용할 수 있다. 즉, 하위 L2/L3 계층에서 Link-Up 및 Link-Down이 동시에 발생하는 Hard Handoff를 가정한다. 이 경우, SCTP 계층에서 Add-IP, Primary-Change 및 Delete-IP 절차가 거의 동시에 수행되어야 한다.

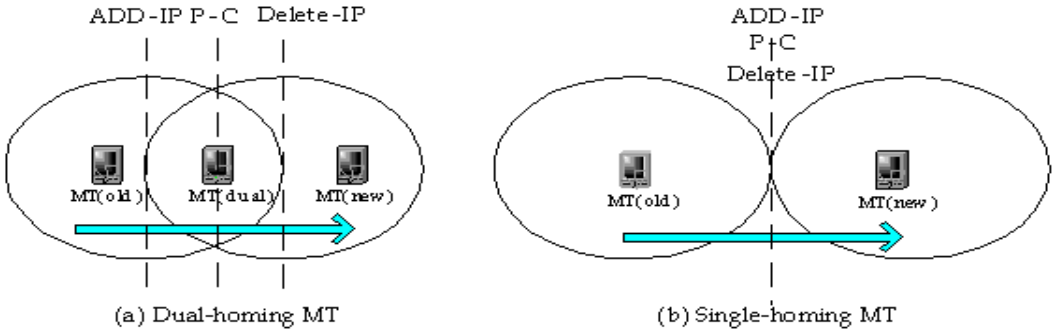

그림 3. SCTP 핸드오버 적용 시나리오

SCTP 기반의 핸드오버는 기존의 MIP (mobile IP) 와는 달리 종단간 수송계층에서 이루어진다는 특징이 있으며, 따라서 터널링 등의 네트워크 라우터의 도움이 없이도 적용 가능 하다. SCTP 핸드오버에 대한 유일한 요구사항은 MN 및 CN에 SCTP가 사용되어야 한다는 점이다. 이러한 mobile SCTP 기능은 차세대 all-IP 기반 이동통신망의 IP 이동성 관리 기법으로 적용될 수 있을 것으로 전망된다.

3. 핸드오버 구현을 위한 SCTPLIB APIs

기본적으로 SCTPLIB의 모든 함수의 API는 RFC 2960의 10장 "Interface with Upper Layer"의 내용에 따라서 크게 ULP-to-Upper 인터페이스와 SCTP-to-ULP Notification, 보조함수로 구분되어서 작성이 되었으며, 각 함수가 사용하는 인자들은 부가적인 설명이 없이도 개발자가 쉽게 이해할 수 있게 설계되었다. 이동성 지원을 위해 추가된 함수 역시 기존 SCTPLIB의 철학을 따라서 사용자에게 친숙한 형태를 가지고 있다.

Int sctp_bindx(int associationID, unsigned char addresses[][SCTP_MAX_IP_LEN],

Int noOfAddresses, int flags);

Int sctp_setPeerPrimary(int associationID,

unsigned char address[SCTP_MAX_IP_LEN]);

3.1 SCTPLIB의 ASCONF 모듈 인터페이스 설계

사용자 인터페이스로서 다음과 같은 APIs의 설계가 필요하다.

int sctp_getLocalAddress(unsigned int assoc_id, struct sockaddr **addrs);
int sctp_getPeerAddress(unsigned int assoc_id, struct sockaddr **addrs);
int sctp_freeLocalAddress(struct sockaddr *addrs);
int sctp_freePeerAddress(struct sockaddr *addrs);

3.1.1 sctp_bindx()

sctp_bindx()의 내부 구현의 전체적인 흐름은 리눅스 커널 코드를 참조한다.

리눅스 커널 코드

```
SCTP_STATIC int sctp_setsockopt_bindx(struct sock* sk,
                                      struct sockaddr __user *addrs,
                                      int addrs_size, int op)
{
        ...
        switch (op) {
        case SCTP_BINDX_ADD_ADDR:
                err = sctp_bindx_add(sk, kaddrs, addrcnt);
                if (err)
                        goto out:
                err = sctp_send_asconf_add_ip(sk, kaddrs, addrcnt);
                break;
        case SCTP_BINDX_REM_ADDR:
                err = sctp_bindx_rem(sk, kaddrs, addrcnt);
                if (err)
                        goto out;
                err = sctp_send_asconf_del_ip(sk, kaddrs, addrcnt);
                break;
        default:
                err = -EINVAL;
                break;
        }
        ...
}
```

위와 같은 형식을 참조로 sctp_bindx()의 내부 구현을 다음과 같이 정의한다.

```
int sctp_bindx(int SCTP_instance, struct sockaddr *addrs, int addrcnt, int flags)
{
        ...
        switch ( flags ) {
        case SCTP_BINDX_ADD_ADDR:
                asc_add_ip();
      |=> mdi_addAddressToInstance()          ? mdi_xxx가 맞는지 고찰.
      |=> mdi_addLocalAddressToAssoc()
                asc_buildParam();
                asc_sendRequest();
        case SCTP_BINDX_REM_ADDR:
```

```
                asc_del_ip();
        |=> mdi_delAddressFromInstance()
        |=> mdi_delAddressFromAssoc()
                asc_buildParam();
                asc_sendRequest();
            default:
            }
    }
```

3.1.2 sctp_setPeerPrimary()

sctp_setPeerPrimary()의 리눅스 커널 구현은 다음과 같다.

```
SCTP_STATIC int sctp_setsockopt(struct sock *sk, int level, int optname,
                                char __user *optval, int optlen)
{
            ...
            switch (optname) {
            case SCTP_SET_PEER_PRIMARY_ADDR:
            retval = sctp_setsockopt_peer_primary_addr(sk, optval, optlen);
            break;
            ...
            }
            ...
}

static int sctp_setsockopt_peer_primary_addr(struct sock *sk, char __user *optval,
                                             int optlen)
{
            ...

            chunk = sctp_make_asconf_set_prim(asoc,
                                        (union sctp_addr *)&prim.sspp_addr);
            if (!chunk)
                    return -ENOMEM;

            err = sctp_send_asconf(asoc, chunk);
            ...
}
```

위와 같은 형식을 참조로 sctp_setPeerPrimary()의 내부 구현은 다음과 같이 구현한다

```
int sctp_setPeerPrimary(int assoc_id, unsigned char address[SCTP_MAX_IP_LEN)
{
        ...
        asc_buildParam();
        asc_sendRequest();
        ...
}
```

3.2 SCTPLIB API 사용 순서

다음 그림은 3.1절의 SCTP 핸드오버 알고리즘을 구현하기 위한 SCTPLIB API 사용법을 순서대로 기술하고 있다.

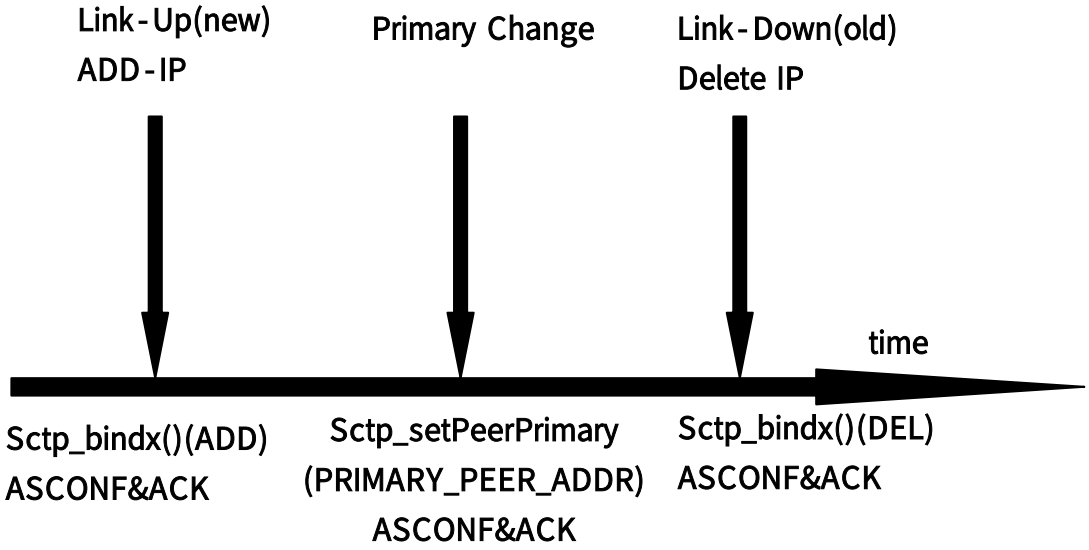

그림 4.핸드오버를 위한 SCTPLIB APIs 호출 순서

그림에서 알 수 있듯이, MT의 이동으로 인해 새로운 영역의 Link-Up 신호가 감지되고 IP 계층에서 새로운 IP 주소를 얻게 되면, SCTP에서 'sctp_bindx()' 함수를 호출하여 Add-IP 기능을 수행한다. Add-IP 함수호출 후에, SCTP ASCONF 패킷이 FS에게 전달되며, FS는 ASCONF-ACK 패킷으로 MT에게 응답한다.

새로운 Link 신호가 강해지는 경우 혹은 별도로 정한 규칙에 의해, MT는 Primary-Change 패킷을 FS에게 전송하고 이를 위해 sctp_setPeerPrimary() 함수를 호출한다. 구체적인 Primary-Change 발생 시점은 구현 및 적용 시나리오에 따라 다를 수 있다.

MT의 추가적인 이동으로 기존 Link의 신호를 감지하지 못하는 경우, 기존 IP 주소는 SCTP

세션에서 삭제되며 이를 위해 'sctp_bindx()' 함수가 사용된다. 함수호출 후에 기존 IP 주소는 더 이상 SCTP 세션에서 사용되지 않는다.

4. 결론

지금까지 본 고에서는 SCTPLIB를 이용한 핸드오버와 그에 해당하는 APIs에 대하여 살펴보았다. SCTP는 TCP 이후의 차세대 수송계층 프로토콜로써 지속적인 표준확장 및 보급이 이어질 것으로 전망된다. 시간이 지남에 따라 SCTP 보급이 확대되면, 기존에 TCP를 통해 제공되던 응용들도 SCTP를 통해 보다 효율적으로 제공될 수 있을 것으로 전망된다.

특히, 실시간 멀티미디어 전송 및 고도의 신뢰성이 요구되는 응용에 대해서는 SCTP의 적용이 선호된다. 또한, 차세대 이동통신망에서의 IP 이동성 제공 측면에서도 SCTP의 사용이 긍정적으로 검토될 수 있을 것이다.

참고 문헌

[1] Stewart R., et al., "Stream Control Transmission Protocol", IETF RFC 2960, October 2000

[2] Pastor J. and Belinchon M., "SCTP Management Information Base", IETF Internet Draft, draft-ietf-sigtran-sctp-mib-08.txt, November 2002

[3] SCTP implementations by Linux, http://rivus.sourceforge.net/

[4] SCTP tutorial, http://www.iec.org/online/tutorials/sctp/

[5] Stewart, R., et al., " SCTP Dynamic Address Reconfiguration", IETF Internet Draft, draft-ietf-tsvwg-addip-sctp-19.txt, Jul 2007