mSCTP-HSS: mSCTP Handover Supporting System

(CPL-TR-07-08)

2007년 8월

경북대학교 통신프로토콜연구실

김동필 (dpkim@cs.knu.ac.kr)

요 약

본 문서는 리눅스 플랫폼환경에서 이동 단말의 새로운 망으로의 이동을 탐지하고 mSCTP 핸드오버를 자동적으로 수행하는 시스템의 설계 및 개발에 관해 언급한다. 또한, 개발된 시스템을 노트북과 같은 이동 단말에 설치하는 과정을 상세히 설명한다.

목 차

1. 서론

본 문서는 리눅스 플랫폼 환경에서 두 개의 WLAN 인터페이스를 장착한 이동 단말이 새로운 망으로 이동하였을 때, 자동적으로 mSCTP 핸드오버를 수행할 수 있는 시스템의 설계 및 개발에 관한 내용이다. 또한, 개발된 시스템을 리눅스 플랫폼에서 설치하는 과정을 담고 있다. 본 시스템은 이동 단말이 서로 다른 IP 정보를 가지는 두 개의 WLAN 망을 이동할 때, 하부 계층(L2 레벨)의 Link-Up/Down을 탐지하여 새로운 망으로의 이동을 탐지하고, 해당 네트워크의 AR(Access Router)로부터 네트워크 정보를 수신한 후, mSCTP 핸드오버를 수행하도록 개발되었다. 본 문서에서는 mSCTP 핸드오버 지원 시스템의 구조를 설명하고, 구현된 시스템의 개발 코드를 간략히 분석한다. 또한, 개발된 시스템을 리눅스 플랫폼 환경에서 설치하는 과정을 가능한 자세히 설명하도록 한다.

2. 시스템 설계

본 시스템은 4개의 기능 모듈로써, 링크 정보 관리자(Link Information Manager), 이동성 탐지(Movement Detection), 핸드오버 결정(Handover Engine), 핸드오버 수행(Handover Manager), mSCTP API로 구성된다.
다음은 mSCTP 핸드오버 시스템의 구성도이다.

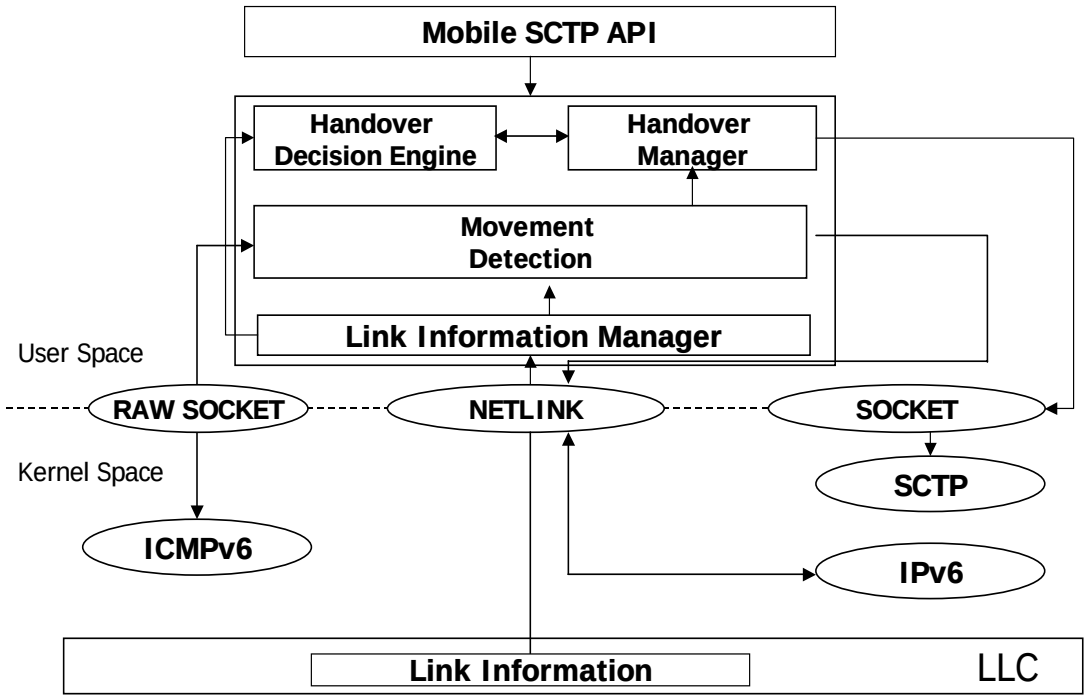


그림 1. SCTP 핸드오버 시스템 구성도

그림 1에서와 같이, SCTP 핸드오버 시스템은 User Space와 Kernel Space로 나누어 설계하였다. User Space는 SCTP 핸드오버 시스템을 지원하기 위해 요구되는 기능 모듈들이 요구되는 부분이고, Kernel Space는 운영체제에서 제공하고 있는 프로토콜 스택들이 존재하는 부분이

다. 실제로 SCTP 핸드오버 시스템이 동작하기 위해서는 운영체제에서 제공하고 있는 기존 프로토콜들 및 Kernel 레벨에서 제공하는 유틸리티들과 상호 연동이 이루어져야 한다. 예를 들어, 이동 단말 시스템이 이동성을 탐지하기 위해서는 L2 (Data Link Layer)레벨에서 RSSI (Received Signal Strength Indication) 정보를 수집하거나, L3 레벨에서 IPv6 주소의 Prefix 정보를 수신하여야 한다. 이를 위해 본 시스템에서는 User Space에서는 SCTP 핸드오버를 지원하기 위한 기능 모듈들을 설계하였고, Kernel레벨에서 제공하고 있는 프로토콜들 및 Kernel API들을 이용하여 Kernel 레벨과의 상호 협력할 수 있도록 설계하였다.
SCTP 핸드오버를 지원하기 위해 설계한 기능 모듈들은 링크 정보 관리자(Link Information Manager), 이동성 탐지 (Movement Detection), 핸드오버 관리자 (Handover Manager), 핸드오버 결정 엔진 (Handover Decision Engine) 등이다. 링크 정보 관리자는 단말의 이동성을 탐지하기 위해 요구되는 기능 모듈로써, L2 레벨의 정보를 지속적으로 수신한다. 이동성 탐지 기능 모듈은 링크 정보 관리자로부터 단말의 이동성을 감지하면 새로운 Prefix를 받아들여 IP 핸드오버의 수행을 결정하고, 핸드오버 결정 엔진과 핸드오버 관리자에 의해 SCTP 핸드오버를 수행하게 된다.
각 기능 모듈들의 세부적인 설명은 다음과 같다.

2.1 링크 정보 관리자 (Link Information Manager)

링크 정보 관리자는 단말의 이동 및 핸드오버 수행을 결정하기 위한 링크 정보를 탐지하는 역할을 수행한다. 또한, 이동 단말이 두개 이상의 서로 다른 미디어의 네트워크 인터페이스들을 탑재하고 있는 경우, 새로운 전송 미디어를 사용하는 영역으로 이동하였을 때, 해당 인터페이스가 활성화됨을 탐지하게 된다. 본 시스템에서 설계한 링크 정보 관리자는 NETLINK LIBRARY를 이용하여 새로운 영역의 링크, 즉 이동 단말과 AR간의 링크를 탐지한다. 단말이 이동하여 해당 AR과의 접속 링크가 끊어졌을 경우 링크 다운 메시지가 발생하고, 단말이 새로운 영역으로 이동하여 해당 AR과 접속 링크가 새로이 생성된 경우 링크 업 메시지가 발생되도록 설계하였다. 데이터 링크 계층과 링크 정보 관리자간의 링크 모니터링을 위해 사용된 메시지들은 다음과 같다.

```
RTM_NEWLINK        // 이동 단말과 해당 AR간의 링크가 활성화 되었을 경우
RTM_DELLINK        // 이동 단말과 해당 AR간의 링크가 비 활성화 되었을 경우
```

링크 정보 관리자가 L2 레벨로부터 NETLINK LIBRARY를 통하여 RTM_NEWLINK 메시지를 수신하였을 경우, 링크 정보 관리자는 곧바로 이동성 탐지 모듈을 호출하여 IPv6의 Router Discovery 절차를 수행하도록 한다. 반면, 링크 정보 관리자가 RTM_DELLINK 메시지를 수신하였을 경우에는 핸드오버 결정 엔진을 호출하여 핸드오버 수행 관리자에게 해당 링크의 IPv6 주소를 삭제하도록 (DELETE-IP) 하도록 요청한다. 또한, 링크 정보 관리자는 이동단말의 핸드오버 수행을 결정하는 변수로 사용하기 위해 해당 링크의 신호세기를 주기적으로 수집하여 핸드오버 결정 엔진에 전송하도록 한다.

2.2 이동 탐지 (Movement Detection)

이동 탐지 기능 모듈은 AR로부터 새로운 네트워크 Prefix를 수신하고, IPv6의 자동 주소 설정 방식에 의해 새로운 Stateless IPv6 주소를 생성하도록 하는 역할을 수행한다. 이를 위해, 링크 정보 관리자가 RTM_NEWLINK 메시지를 수신하여 이동 탐지 모듈을 호출하게 되면, 곧바로 IPv6의 Router Discovery 절차를 수행시킨다. 이동 탐지 모듈은 Kernel Space의 ICMPv6을 사용하기 위해 ICMPv6 Raw Socket API를 사용하여 Router Solicitation (RS) 메시지를 전송하고, AR로부터 Router Advertisement (RA) 메시지를 수신하게 된다. 수신한 RA 메시지에는 Network Prefix를 포함하고 있고, 이를 추출하여 현재 사용 중인 Prefix와 비교한다. 만일 수신 Network Prefix가 기존의 것과 다르다면, 이동 단말이 새로운 IP 영역으로 이동하였다고 판단하고 해당 IPv6 주소를 생성하게 된다. 생성된 IPv6 주소는 핸드오버 결정 엔진에 등록되고, 핸드오버 관리자에게 SCTP의 ADD-IP 기능을 수행하도록 요청한다.

2.3 핸드오버 결정 엔진 (Handover Decision Engine)

핸드오버 결정 엔진은 핸드오버 수행 여부(주요 IP 전송 경로 변경 요청)를 결정하는 모듈로써, 링크 정보 관리자로부터 수신한 링크 신호 강도, 네트워크 지연시간 및 사용되고 있는 대역폭 등을 고려한 핸드오버 수행 결정 알고리즘에 의거하여 핸드오버 수행 여부를 판단한다. 구체적인 핸드오버 수행 결정 알고리즘에 대해서는 언급하지 않는다.

2.4 핸드오버 관리자 (Handover Manager)

핸드오버 관리자는 링크 정보 관리자나 이동 탐지 기능 모듈로부터 SCTP의 핸드오버 수행을 요청 받으면, Kernel Space의 SCTP 프로토콜 스택을 참조하여 SCTP 핸드오버 수행을 위해 SCTP의 제어 메시지인 Address Configuration (ASCONF) Chunk을 생성하도록 한다. 즉, 이동 탐지 모듈로부터 새로운 IPv6주소가 생성되고, 이를 보고 받으면 핸드오버 관리자는 SCTP의 ADD-IP 기능을 수행하기 위해 SCTP_BINDX() 함수를 호출하여 SCTP 스택이 ASCONF Chunk를 생성하도록 한다. 이동 단말이 ADD-IP를 위해 ASCONF Chunk를 전송하면 상대 노드는 이에 대한 응답으로 ASCONF-ACK chunk를 전송하게 되고, 이동 단말이 이를 정상적으로 수신하게 되면 두 노드간의 SCTP 세션에서 새로운 주소가 Binding되게 된다. 더욱이, 링크 정보 관리자로부터 DELTET-IP 수행을 요청 받으면, 같은 방식으로 핸드오버 관리자는 ASCONF/ASCONF-ACK를 송수신 하여 SCTP DELTE-IP 기능을 수행하도록 한다. 한편, 핸드오버 결정 엔진이 지속적으로 수신한 링크 정보를 비교하여 새로이 이동한 영역의 링크 정보가 훨씬 좋다고 판단하였을 경우, 핸드오버 관리자는 SETSOCKOPT(PRIMARY CHANGE) API를 호출하여 주요 데이터 전송 경로를 변경하게 된다. 이와 같은 경우에도 역시 ASCONF/ASCONF-ACK Chunk를 송수신하여 수행하게 된다.

아래의 Flow Charter들은 SCTP 핸드오버 수행을 위한 ADD-IP, Primary-Change 그리고 DELTE-IP 기능에 따른 각 기능 모듈간의 수행 절차를 보여주고 있다.

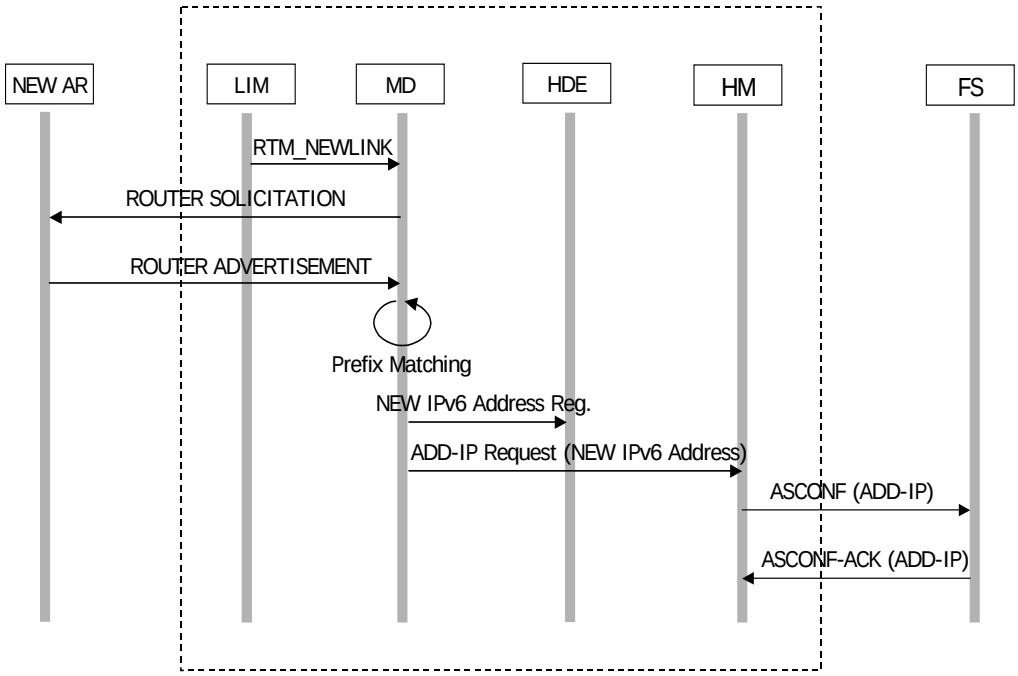


그림 2 SCTP 핸드오버를 위한 ADD-IP 수행 흐름도

위의 그림에서 같이, 링크 정보 관리자(LIM)가 RTM_NEWLINK 메시지를 수신하면 이동 탐지 기능 모듈 (MD)와 해당 AR (NEW AR)는 IPv6 RS/RA 메시지를 송수신하고, 이동 탐지 기능 모듈 (MD)은 새로이 수신한 Prefix를 기존의 것과 비교한다. 새로이 수신한 Prefix가 새로운 영역의 것이라면 IPv6 주소를 Stateless하게 생성하고 이를 핸드오버 결정 엔진 (HDE)에게 등록한 후, HM은 상대 노드 (FS)에게 ASCONF를 송신하고, 이에 대한 응답으로 ASCONF-ACK를 수신하면, SCTP 핸드오버 수행을 위한 ADD-IP가 완료되게 된다.

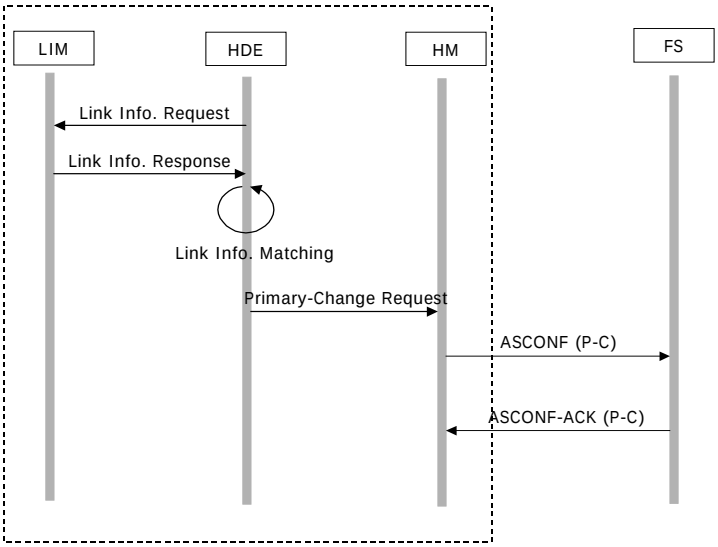


그림 3 SCTP 핸드오버를 위한 Primary-Change 수행 흐름도

위의 그림에서 같이, 핸드오버 결정 엔진(HDE)은 주기적으로 활성화된 네트워크 인터페이스에 대해 해당 링크 정보를 링크 정보 관리자(LIM)에게 요청한다. 링크 정보 관리자는 L2 레벨의 정보를 수집하여 이를 핸드오버 결정 엔진에 보고하고, 핸드오버 결정 엔진은 이를 비교하여 새로이 추가된 경로의 링크 정보가 좋을 경우, 핸드오버 관리자(HM)에게 주요 데이터 전송 경로를 새로운 경로로 변경 (Primary-Change)하도록 요청한다. 핸드오버 관리자는 상대노드(FS)에게 ASCONF를 보내고, ASCONF-ACK를 수신하면 이후 주요 데이터 전송 경로는 새로운 경로로 변경되어 데이터 송수신이 이루어지게 된다.

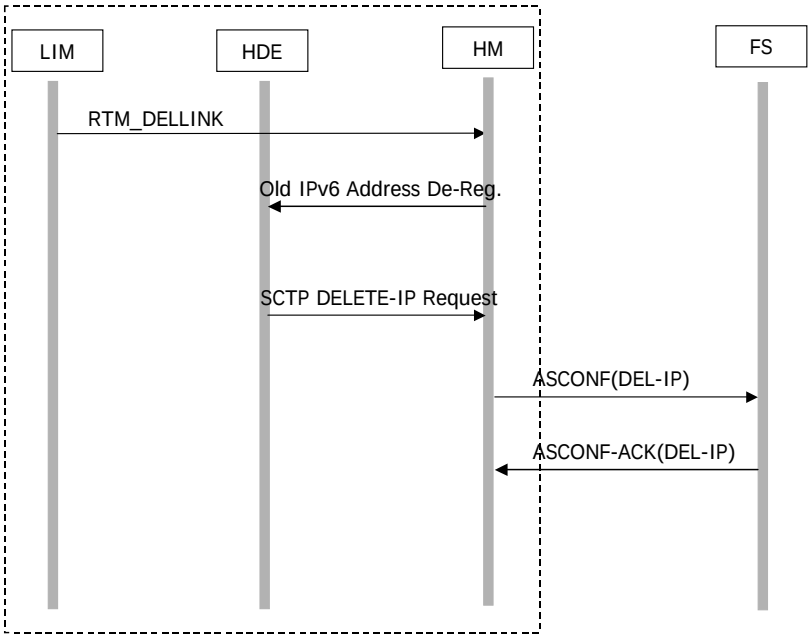


그림 4 mSCTP 핸드오버를 위한 DELETE-IP 수행 흐름도

그림 4와 같이, 링크 정보 관리자(LIM)가 RTM-DELLINK를 수신하면 핸드오버 관리자(HM)가 핸드오버 결정 엔진에 해당 IPv6 주소를 등록 풀(pool)에서 삭제하도록 요청한다. 이후, 핸드오버 결정 엔진은 핸드오버 관리자에게 SCTP DLETE-IP를 요청하고 핸드오버 관리자와 상대노드 (FS)사이에 ASCONF/ASCONF-ACK 메시지가 정상적으로 송수신되면 DLETE-IP 수행이 완료된다.

3. 시스템 개발 코드 분석

mSCTP 핸드오버 시스템은 앞서 언급하였듯이, NETLINK를 사용하여 하부 계층의 링크 정보 (e.g., Link-Up 또는 Link-Down)를 탐지하는 부분과, ICMPv6 소켓을 활용하여 IPv6 RS 및 RA메시지를 생성한 후 전송하고, 수신하는 부분, 그리고 mSCTP 핸드오버를 실제 수행하는 부분으로 구성된다.

본 시스템에서는 기존의 시스템 라이브러리인 NETLINK를 수정하여 이동 단말과 해당 액세스 포인트간의 링크 업/다운을 탐지하도록 하였다. 다음 그림은 링크 정보 관리자가 RTM_NEWLINK와 RTM_DELLINK를 수신하여 처리하는 루틴을 보여주고 있다.

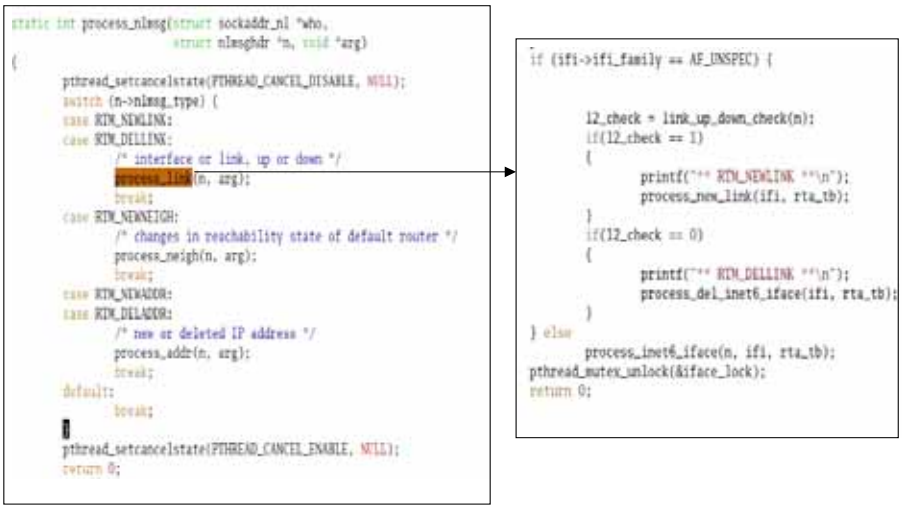
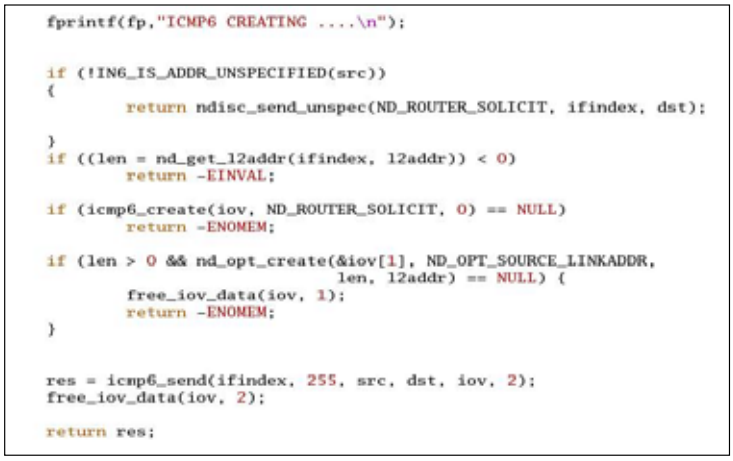


그림 5 링크 정보 관리자의 RTM_NEWLINK/RTM_DELLINK 처리 루틴

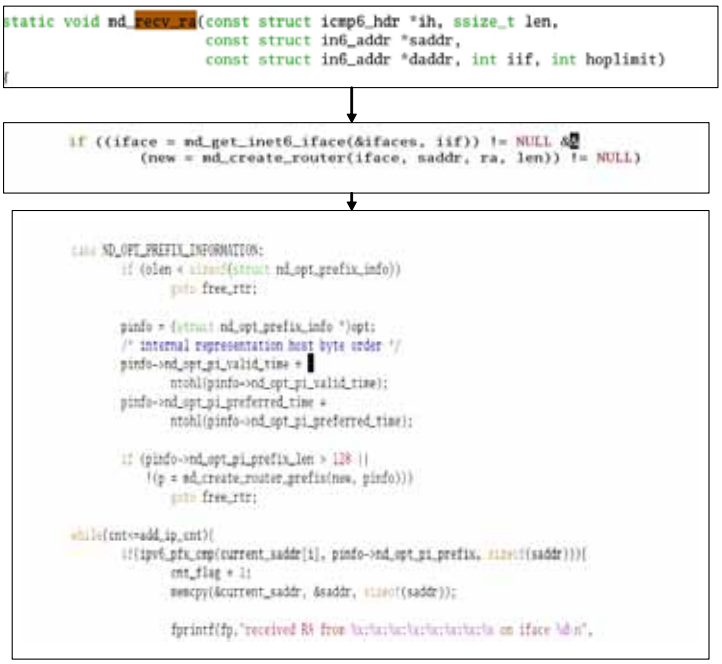
위의 그림에서와 같이, process_nlmsg() 함수가 본 시스템에서 데몬 프로그램으로 동작하게 되고, L2 레벨에서 새로운 링크가 활성화 되면 RTM_NEWLINK 메시지가 링크 정보 관리자에 수신하게 된다. RTM_NEWLINK나 RTM_DELLINK 메시지가 입력되면 링크 정보 관리자의 process_link() 함수에 의해 해당 루틴으로 연결되게 된다. RTM_NEWLINK의 경우에는 process_new_link()를 호출하게 되고, RTM_DELLINK는 process_del_link()를 호출하게 된다. process_new_link는 이동성 탐지 기능 모듈에 속하는 함수로써 ICMPv6 Raw Socket을 생성하여 IPv6 RS 메시지를 전송하도록 한다. 그림 6은 이동성 탐지 기능 모듈에서 ICMPv6 raw Socket을 이용하여 RS메시지를 전송하는 루틴을 보여준다.



```
fprintf(fp,"ICMP6 CREATING ....\n");


if (!IN6_IS_ADDR_UNSPECIFIED(src))
{
        return ndisc_send_unspec(ND_ROUTER_SOLICIT, ifindex, dst);

}
if ((len = nd_get_l2addr(ifindex, l2addr)) < 0)
        return -EINVAL;

if (icmp6_create(iov, ND_ROUTER_SOLICIT, 0) == NULL)
        return -ENOMEM;

if (len > 0 && nd_opt_create(&iov[1], ND_OPT_SOURCE_LINKADDR,
                             len, l2addr) == NULL) {
        free_iov_data(iov, 1);
        return -ENOMEM;
}


res = icmp6_send(ifindex, 255, src, dst, iov, 2);
free_iov_data(iov, 2);

return res;
```

그림 7 이동성 탐지 기능 모듈에서 RS 메시지 전송 루틴

위의 그림과 같이, ICMPv6 Raw Socket과 RS 메시지를 생성하여 해당 AR의 링크 주소로 전송하게 된다. 이동 단말은 이에 대한 응답으로 RA메시지를 수신하게 된다. 다음 그림은 송신한 RS에 대한 응답으로 수신할 RA메시지 처리 루틴을 보여주고 있다.



```
static void nd_recv_ra(const struct icmp6_hdr *ih, ssize_t len,
                       const struct in6_addr *saddr,
                       const struct in6_addr *daddr, int iif, int hoplimit)
{
```

```
    if ((iface = nd_get_inet6_iface(&ifaces, iif)) != NULL &&
            (new = nd_create_router(iface, saddr, ra, len)) != NULL)
```

```
    case ND_OPT_PREFIX_INFORMATION:
        if (olen < sizeof(struct nd_opt_prefix_info))
            goto free_rtr;

        pinfo = (struct nd_opt_prefix_info *)opt;
        /* internal representation host byte order */
        pinfo->nd_opt_pi_valid_time =
            ntohl(pinfo->nd_opt_pi_valid_time);
        pinfo->nd_opt_pi_preferred_time =
            ntohl(pinfo->nd_opt_pi_preferred_time);

        if (pinfo->nd_opt_pi_prefix_len > 128 ||
            !(p = nd_create_router_prefix(new, pinfo)))
            goto free_rtr;

    while(cnt<=add_ip_cnt){
        if(!ipv6_pfx_cmp(current_saddr[i], pinfo->nd_opt_pi_prefix, sizeof(saddr))){
            cnt_flag = 1;
            memcpy(&current_saddr, &saddr, sizeof(saddr));

            fprintf(fp,"received RA from %x:%x:%x:%x:%x:%x:%x:%x on iface %d\n",
```

그림 8 이동성 탐지 기능 모듈에서 수신 RA 메시지 처리 루틴

그림 8에서와 같이, recv_ra()함수가 thread로써 동작하게 되고, 수신한 RA가 이미 보낸 RS에 대한 응답 RA인지를 md_get_inet6_iface()를 통하여 확인한다. md_get_inet6_iface()의 반환값이 NULL이 아닐 경우, RA메시지가 RS의 응답으로 간주하고, RA에 포함된 Network Prefix를 추출한다. 추출된 Prefix가 새로운 IP 영역에서 수신된 것인지를 확인하기 위해 ipv6_pfx_cmp()를 호출한다. Ipv6_pfx_cmp()의 반환 값이 참이면 이는 새로운 Network Prefix로 간주하고 핸드오버 관리자에게 ADD-IP 수행을 요청한다.
다음 그림은 핸드오버 관리자 기능 모듈에서 SCTP 핸드오버를 수행하기 위한 루틴을 보여준다.

그림 9 핸드오버 관리자 기능 모듈에서 ADD-IP 수행 루틴

위의 그림에서와 같이, 핸드오버 관리자에서 sctp_bindx() 함수를 호출하여 새로이 생성된 IPv6 주소를 동적으로 SCTP 세션에 바인딩한다. 또한, setsockopt() 함수를 호출하여 Primary-Change를 수행하고 있다.
다음 그림은 핸드오버 관리자가 DELTE-IP를 수행하기 위한 처리 루틴을 보여준다.

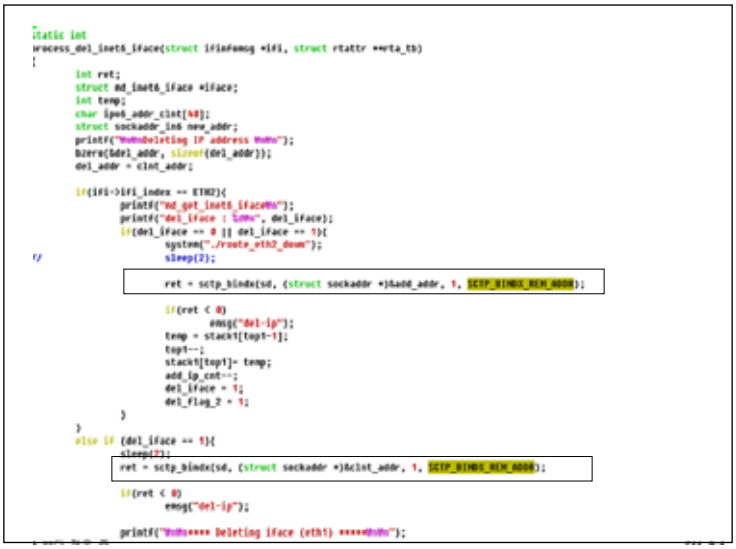


그림 10 핸드오버 관리자 기능 모듈에서 DLETE-IP 수행 루틴

위의 그림과 같이, 링크 정보 관리자가 RTM_DELLINK를 수신하면 핸드오버 관리자의 process_del_inet6_iface() 함수를 호출하고, process_del_inet6_iface()에서 sctp_bindx() API를 호출하여 SCTP DELETE-IP를 수행하게 된다.

4. 시스템 개발 및 설치

본 장에서는 mSCTP 핸드오버 지원 시스템 개발에 관해 설명한다.

4.1 프로그램 요구사항

본 프로그램은 이동 단말(MT)과 고정된 서버(FS)상에서 동작한다. MT는 FS와 SCTP 세션을 시작하고 IPTV 서비스를 제공받기 위해 서비스를 요청한다. 반면, FS는 IPTV 서비스를 제공하는 주체로써 SCTP 세션 요청을 기다리고 이를 수락하게 된다.
Mobile SCTP – IPTV 프로그램을 MT와 FS에 각각 동작 시키기 위해서는 다음과 같은 OS와 응용 프로그램들이 두 단말에 설치되어 있기를 권장한다.

4.1.1 Mobile Terminal (MT)

Fedora 5 with Kernel 2.6.16 (kernel 2.6.16으로 kernel compile)
Kernel 2.6.16으로 kernel compile 시, kernel option에서 가능한 SCTP 관련 부분들이 모두 built in (*)으로 설정되어 있어야 한다.

LKSCTP-TOOLS-1.0.1
LKSCTP-TOOLS-1.0.1은 SOURCEFORGE의 OPEN PROJECT로써 SCTP API들을 응용 레벨에서 쉽게 사용할 수 있도록 하는 UTILITY이다. 따라서, SCTP 핸드오버를 지원하는 본 프로그램에서는 꼭 설치가 되어 있어야 한다. 참고로, LKSCTP TOOL은 현재까지 1.0.6 버전까지 출시되어 있다. 그러나, 가능하면 LKSCTP-TOOLS-1.0.1로 설치하여 본 프로그램과 호환성이 있도록 하는 것이 좋을 듯 하다.

UNP 코드
UNP 소스코드는 “Unix Network Programming” 책에서 제공하는 예제 코드로써 MSCTP-IPTV 프로그램에서 Primary Address를 화면에 출력시키기 위해 사용되고 있다.

WIRELESS_TOOLS
WIRELESS_TOOLS.28 는 하부레벨에서 AP와의 연결성에 관한 이벤트를 제공하기 위해 사용되는 UTILITY이다. MT가 새로운 영역으로 이동하여 새로운 AP와 연결이 설정되면 WIRLEESS_TOOLS에서 새로운 링크 업에 대한 이벤트를 상위 응용 프로그램을 송신한다. 본 프로그램에서는 이를 수신하여 SCTP 핸드오버를 수행하게 되도록 설계 및 개발되어 있다.
본 프로그램의 정확한 명칭은 “wireless-tools_28.orig.tar.gz” 이다.

QT-X11-OPENSOURCE-SRC-4.1.4
QT-X11-OPENSOURCE-SRC-4.1.4는 MT가 IPTV서비스를 제공받기 위해 설치하는 프로그램의 UI를 지원하기 위한 응용 프로그램이다. 참고로, 본 프로그램의 UI는 QT4를 이용하여 개발되었다.

MPLAYER-1.0PRE8

MPLAYER-1.0PRE8은 IPTV서비스에서 멀티미디어 서비스를 지원하기 위한 응용 프로그램으로써 각종 멀티미디어 CODEC을 지원하고 있다.

4.1.2 Fixed Server (FS)

FS에는 특별한 UI가 제공되지 않고 텍스트 기반으로 서비스를 제공한다. 따라서, 다음과 같이 3개의 프로그램만 설치되어 있다면 간단히 동작하게 된다.

Fedora 5 with Kernel 2.6.16 (kernel 2.6.16으로 kernel compile)

LKSCTP-TOOLS-1.0.1

UNP 코드

4.2 Mobile SCTP – IPTV 프로그램 설치

2장에서 언급한 응용프로그램 및 OS가 올바르게 설치되어 있다면, Mobile SCTP – IPTV 프로그램을 설치해야 한다. 먼저, MT용 Mobile SCTP –IPTV 프로그램을 설치해보자. MT용 프로그램 명은 s-tv-0.4.tar.gz이고, /usr/local/src/에 해당 소스코드를 설치하도록 한다.

먼저 s-tv-0.4.tar.gz의 압축파일을 풀고, 다음과 같은 절차로 컴파일 할 수 있다.

```
$ tar zxvf s-tv-0.4.tar.gz
$ cd s-tv-0.4
$ ls
ChangeLog    Makefile Makefile.vars          READEME          gui
Modules      msctp          ref      temvod  test     trace.dat

$ cd msctp
$ make lib
```

MT용 프로그램인 /usr/local/src/s-tv-0.4/msctp 디렉토리 안에서 위와 같이 컴파일 하는 도중 RTNL 관련 컴파일 에러가 발생한다면 다음과 같이 수정해보자.

RTNL 관련 에러 수정

---

cd s-tv-0.4

mv /usr/include/linux/rtnetlink.h /usr/include/linux/rtnetlink.h.back

cp /usr/local/src/linux-2.6.16/include/linux/rtnetlink.h /usr/include/linux/

또한, NETLINK_ADD_MEMEBERSHIP 관련 컴파일 에러가 발생한다면 다음과 같이 수정해 보자.

NETLINK_ADD_MEMEBERSHIP 관련 에러 수정

---

mv /usr/include/linux/netlink.h /usr/include/linux/netlink.h.back

cp /usr/local/src/linux-2.6.16/include/linux/netlink.h /usr/include/linux/

만일, 위와 다른 컴파일 에러가 발생한다면 dpkim@cs.knu.ac.kr로 에러를 보고해주기 바란다.

msctp 폴더 안에서 정상적으로 컴파일이 수행되었다면 msctp 폴더 안에 “libmsctp.a” 라는 라이브러리가 생성되었을 것이다.

그 후, 다음과 같이 msctp 폴더를 나와 gui폴더 안으로 들어가서 compile을 수행하도록 한다.

```
$ cd ..
$ cd gui
$ make
```

gui 폴더 안에서 컴파일 에러가 발생되었다면 Makefile에 명시되어 있는 경로에 해당 라이브러리와 실행파일들이 제대로 있는지 확인해봐야 한다.

만약, gui 폴더 안에서 컴파일이 올바르게 수행되었다면, /usr/local/src/s-tv-0.4 폴더 안에 “s-tv” 라는 실행파일이 생성되었을 것이다.

다음으로 FS에서 MSCTP-IPTV 프로그램을 설치해보자. FS에서 본 프로그램은 별 문제없이 설치가 될 것이다. FS용 프로그램은 /usr/local/src 에 있으며 다음과 같은 파일로 구성된다.

```
$ tar zxvf vod-etri-server.tar.gz
$ cd vod-etri-server
$ ls
Makefile libunp.a print.c vod_24s1e1.avi vodserv6 vodserv6_print.c
client.h net.sh server.h vod_losts1e1.avi vodserv6.c

$ make
```

위와 같이 컴파일을 수행하면 “vodserv6” 라는 실행파일이 생성된다. 만일 컴파일 에러가 발생한다면 vodserv6.c 파일의 137 라인에서 .avi 파일의 경로를 수정해주기 바란다.

4.3 mobile SCTP – IPTV 프로그램 동작 방법

3장에서 언급한대로 MT와 FS에 Mobile SCTP – IPTV 프로그램이 정상적으로 설치가 되었다면, 이제 실행시켜보자.

두 MT와 FS에서 프로그램을 실행시키기 전에 소스코드에서 주소를 바로 잡도록 하자. 본 프로그램은 SCTP 초기화를 위해 기본적으로 MT에 2001:220:0:1:20d:28ff:fe46:4c3b 가 BINDING 되도록 설정되어 있고, MT에서 FS로 연결하기 위해 MT는 서버용 주소 2001:220:10::155를 알고 있다고 가정한다.
만일, MT에서 자신의 주소를 변경하고자 할 때는 gui 폴더 안의 vodclnt6.c에서 203 라인에 새로운 주소를 설정하면 된다. 또한, MT에서 연결하고자 하는 FS의 주소를 변경하고자 할 경우에는 역시 MT에서 gui 폴더 안의 main_gui.cpp 파일에서 66 라인의 주소를 변경하면 된다.
주소를 변경하였다면 다시 3장에서 언급한대로 ‘s-tv-0.4’(MT용 프로그램)
과 ’vod-server’ (서버용) 프로그램들을 재 컴파일하고 프로그램을 실행시켜보자.

MT와 FS에서 프로그램을 실행시키기 위해서는 다음과 같은 절차로 수행하면 된다. 단, FS가 MT보다 먼저 프로그램이 실행되어 있어야 한다. 이는 FS가 서버로써 SCTP 세션 초기화를 기다리고 있고, MT는 SCTP 세션을 요청하기 때문이다.

4.3.1 Fixed Server (FS)

```
$ cd vod-etri-server      / / Mobile SCTP – IPTV 의 FS용 프로그램 이름
$ ./vodserv6 9998         / / 9998는 포트 번호이다.
```

4.3.2 Mobile Terminal (MT)

```
$ cd s-tv-0.4                    / / Mobile SCTP – IPTV의 MT용 프로그램 이름
$ ./s-tv
```

MT에서 프로그램이 동작하여 GUI 화면이 디스플레이 되면 Home (집모양)을 클릭하고 Server 주소를 선택한다. 본 프로그램에서는 CINEMA Server라고 명시되어 있는 것을 클릭하고 Connect 버튼을 누르면 FS와의 연결이 진행된다. 정상적으로 연결이 되면 서비스 제공 목록이 서버로부터 전송되고, MT에서는 서비스 번호 1 번을 입력하고 Request Contents를 클릭한다. 그런 후, 화면표시를 클릭하고 play버튼을 누르면 IPTV 서비스가 시작된다. 이후, 새로운 AP가 나타나면 자동적으로 SCTP 핸드오버가 수행되게 된다.

4.4 테스트베드 구성 및 실행

다음 그림은 실험에 사용된 테스트베드 구성 도이다.

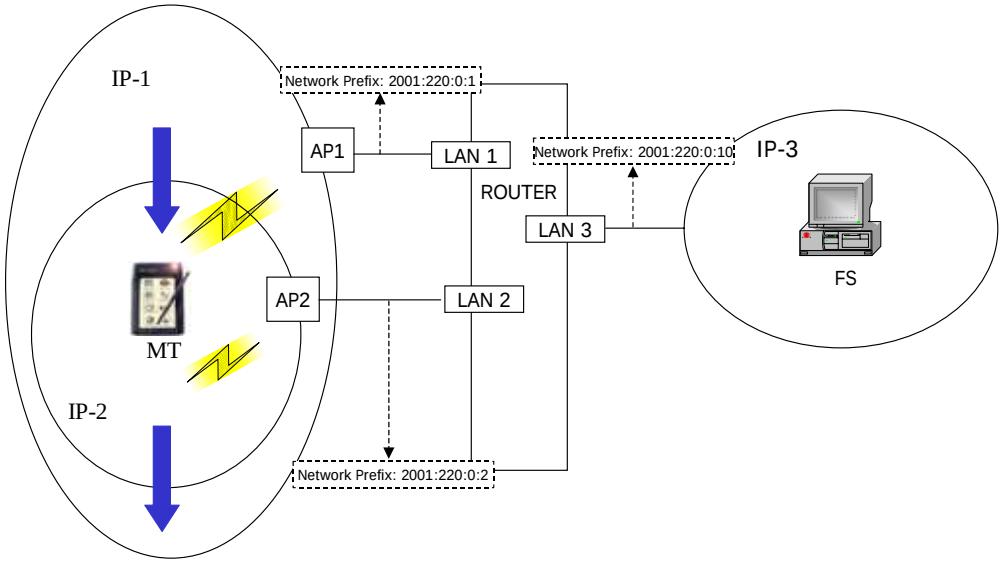


그림 10 실험을 위한 테스트베드 구성도

위의 그림에서와 같이, 테스트 베드는 3개의 다른 IP영영으로 구성된다. 서버 (FS)는 IP-3영역에서 IPTV 서비스를 제공하고, 이동 단말은 IP-1영역에서 IP-2 영역으로 이동하게 된다. 제안 시스템을 평가하기 위해 적용된 실험 시나리오는 다음과 같다.

가. 세션 초기화

먼저 서버는 2001:220:0:3::/64의 Network Prefix를 사용하며, AP1영역에서 2001:220:0:1::/64의 Network Prefix를 가지는 이동 단말 클라이언트와 SCTP세션을 초기화한다. 이후, 클라이언트는 서버로부터 IPTV 서비스를 제공받는다.

나. ADD-IP

단말이 2001:220:0:2::/64의 Network Prefix를 사용하는 AP2로부터 새로운 신호를 수신하면, SCTP 세션에 동적으로 새로이 생성된 IPv6 주소를 추가(Binding)한다.

다. Primary Change (AP1->AP2)

단말이 AP1와 AP2사이에 있고, AP2 영역의 링크 정보가 AP1보다 좋을 때, 새로이 추가된 IPv6주소를 주요 데이터 전송 경로로 변경하고자 서버에게 요청하고, 이에 대한 응답을 받는다.

라. Primary Change (AP2->AP1)

단말이 AP2와 AP1사이에 있고, AP1 영역의 링크 정보가 AP2보다 좋을 때, 새로이 추가된 IPv6 주소를 주요 데이터 전송 경로로 변경하고자 서버에게 요청하고, 이에 대한 응답을 받는다.

마. Delete-IP

이동 단말이 HOTSPOT 영역을 완전히 벗어나 해당 AP로부터 더 이상 신호를 받지 않을 시점에, 이동 단말은 해당 주소를 진행 중인 SCTP 세션에서 삭제하고, IPTV 서비스를 지속적으로 진행한다.

본 실험을 통해, 제안된 설계를 통해 개발된 IPTV 시스템이 잘 동작 하고 있음을 알 수 있다.
그림 11은 본 시스템에 의해 동작되는 화면을 보여주고 있다.

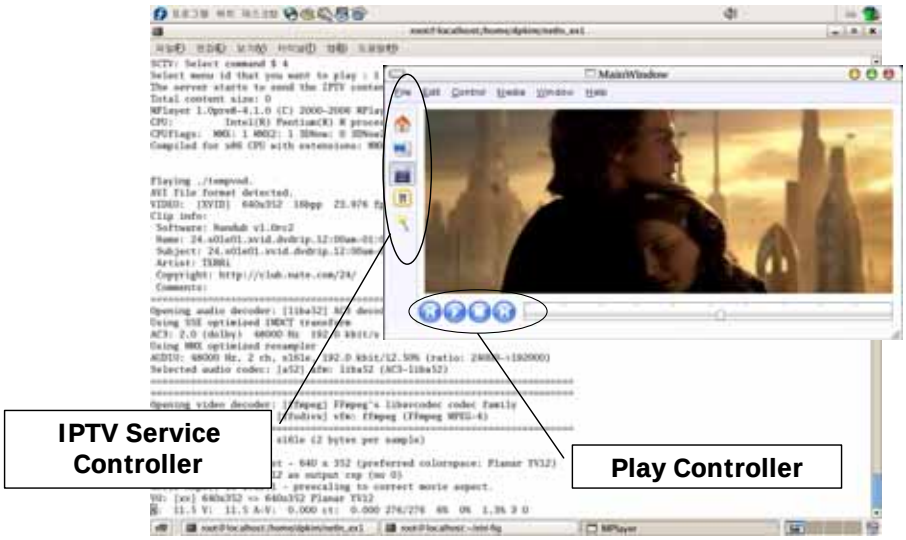


그림 11 실험 결과 화면

그림 11과 같이, 개발된 시스템은 클라이언트의 GUI를 통해 서버로부터 채널 목록을 수신하고, 사용자가 원하는 채널을 선택하면, 해당 멀티미디어 컨텐츠를 플레이하여 볼 수 있도록 되어 있다. 더욱이, SCTP 핸드오버 시스템은 이동 단말의 하위 계층 (L2 및 L3)정보를 분석하여 응용 레벨에서 동작하는 시스템과 독립적으로 수행하도록 개발되었다. 다시 말해, 이동 단말이 새로운 IP 영역으로 이동하여 SCTP핸드오버가 수행되더라도 응용 레벨의 IPTV 시스템에는 어떠한 영향도 미치지 않는다.

4.5 참고사항

본 프로그램에서 SCTP 핸드오버를 수행시키기 위해서는 CISCO AIRONET 350 시리즈의 WLAN카드와 INTEL 2100 BG나 INTEL 2200 BG WLAN 카드가 구비되어 있어야 한다. 더욱이, CISCO WLAN 카드가 eth1이고 INTEL WLAN 카드가 eth2로 설정되어 있어야 정상적으로 동작하게 된다.

5. 결론

본 문서에서는 이동 단말이 새로운 망으로 이동하였음을 탐지하고, 새로운 망으로 mSCTP 핸드오버를 자동적으로 수행하는 시스템에 관한 설계 및 개발에 대해 상세히 설명하였다. 현재 개발된 시스템은 WLAN 간 IP 이동성을 지원하는 시스템으로써 노트북과 같은 이동 단말에 두 개의 WLAN 카드를 장착하고 있어야 하며, 단말의 이동성을 연출하기 위해 AP와의 링크를 빼고 꼽는 방식으로 mSCTP 핸드오버를 실행시킬 수 있도록 하였다.
향후연구로써 본 시스템을 WiBro나 3G와 같은 이 기종 무선 인터페이스를 장착하여 실제 이 기종 망에서 mSCTP 핸드오버를 지원할 수 있는 시스템으로 확장할 계획이다.