SCTP 핸드오버 분석 (Linux)

<CPL-TR-05-01>

2005년 1월

고 석 주 (sjkoh@cs.knu.ac.kr)

요 약

본 문서에서는 차세대 수송계층 프로토콜인 SCTP(Stream Control Transmission Protocol) 기반 핸드오버 기법에 대하여 논의한다. 이를 위해, 리눅스 플랫폼에서 SCTP 핸드오버 실험을 위한 테스트베드를 구성하고, dual-homing 이동단말과 single-homing 이동단말에 대한 핸드오버 성능을 분석하였다. 실험결과에 따르면, dual-homing 이동단말의 경우 중첩지역에서 두 개의 IP 주소를 동시에 사용할 수 있어서 핸드오버 지연은 이동단말과 외부 단말간의 RTT(Round Trip Time) 시간에 영향을 받으며, 특히 본 실험의 경우 전체 핸드오버 과정이 1초 이내에 완료되었다. 한편, single-homing 이동단말의 경우, 한 순간에 하나의 IP 주소만을 사용함에 따라서 핸드오버 지연은 RTT보다는 하위 링크 및 네트웍 계층의 핸드오버 시간에 크게 좌우됨을 확인하였다.

목 차

1. 서론

차세대 통신망이 'All-IP' 기반의 유무선통합망으로 발전해 나감에 따라, IP 이동성(mobility) 요구사항이 증가하고 있다. 기존의 WLAN, 3G 셀룰러시스템과 함께 IEEE 802.16 기반의 'WiBro(Wireless Broadband)' 서비스가 활성화될수록 IP 이동성에 대한 필요성은 더욱 증가할 것이다. 특히, 차세대 통신망에서의 IP 이동성 기술은 '동일망내'에서의 이동 뿐만 아니라, '이종망간'(예: 3G-WLAN, 3G-WiBro) 이동에 대해서도 Seamless 서비스를 제공해야 한다. [1]

IP 이동성 기술은 크게 위치관리(Location Management) 기술과 핸드오버(Handover Management) 기술로 구분된다. 위치관리 기술은 이동단말(Mobile Terminal, MT)이 현재 사용중인 IP 주소 정보를 위치관리 서버에 등록하는 기술이며, 궁극적으로 외부 단말이 MT에 연결설정을 요구할 때에 적용된다. 반면에, 핸드오버 기술은 MT의 이동 중에 IP 주소가 바뀌어도, 진행 중인 세션이 끊기지 않도록 세션의 연속성을 제공하는 기술이다. 핸드오버 기술의 주요 목표는 핸드오버 동안에 발생할 수 있는 데이터 손실 및 핸드오버 지연시간을 최소화 하는 것이다.

현재까지 주로 연구되어온 IP 이동성 기술로는 MIP(Mobile IP)가 있다. MIP에서 MT는 HA(Home Agent)에 현재의 위치를 등록한다. 한편, 핸드오버 지원을 위해 MIP를 확장한 FMIP(Fast Handover for MIP) 및 HMIP(Hierarchical MIP) 방식이 제안되어 왔다. MIP는 네트웍 계층의 이동성 지원 기술로써, 특히 핸드오버 지원을 위해서는 라우터간 핸드오버 터널을 설정해야 하며, 이는 기본적으로 망에 투자를 요구한다.

본 문서에서는 수송계층(Transport Layer)에서의 이동성 기술인 SCTP(Stream Control Transmission Protocol)[2, 3] 이동성 기법에 대하여 논의한다. SCTP는 TCP/UDP에 이은 3번째 수송계층 프로토콜 표준으로써 향후에 이동단말에 널리 사용될 것으로 전망된다. 본 문서는 이동단말에 SCTP가 탑재되어 핸드오버를 지원하는 'SCTP 핸드오버' 기법에 대한 성능분석 결과를 기술한다. SCTP는 멀티홈잉(multi-homing) 특성을 제공하며, 하나의 세션에 두 개 이상의 IP 주소를 바인딩(binding) 할 수 있다[4]. 이를 활용하여 SCTP는 IP 주소가 바뀌어도 새로운 주소를 세션에 추가함으로써 핸드오버 기능을 수행할 수 있다.

기존 연구에서는[5, 6, 7] SCTP 핸드오버에 대한 성능 분석을 위해 ns-2(Network Simulator)를 사용하였다[8,9]. 특히, 하위 무선 링크계층의 신호세기를 토대로, 언제 새로운 IP 주소를 추가하고 기존 IP 주소를 삭제할 것인지에 대한 실험 분석결과를 제공하고 있다.

본 문서에서는 리눅스(Linux) 플랫폼 기반 테스트베드에서 SCTP 핸드오버 성능분석 결과에 대하여 기술한다. 리눅스 커널 2.6.0 이상에서 제공되는 SCTP API(Application Programming Interfaces)를 토대로 [10, 11] 핸드오버 성능분석을 위한 테스트베드를 구축하고, 관련 실험을 통해 소요되는 핸드오버 지연(latency) 시간을 분석하였다. 특히, 동일망간 Horizontal 핸드오버와 이종망간 Vertical 핸드오버 환경에 따라 실험 시나리오를 구성하였으며, 각 시나리오별 SCTP 핸드오버 성능을 비교, 측정 및 분석하였다.

본 문서는 다음과 같이 구성된다. 먼저, 2절에서는 SCTP 수송계층 이동성 지원 모델에 대하여 간략히 언급하고, 기존 MIP 방식과의 차이점을 기술한다. 3절에서는 SCTP 핸드오버 알고리즘 및 이를 리눅스 플랫폼에서 구현하기 위한 SCTP API 함수의 사용 방법에 대하여 기술한다. 4절에서는 동일망간 및 이종망간 핸드오버 시나리오에 대한 테스트베드 실험결과를 기술하고, 각 시나리오에 대한 핸드오버 성능 분석결과를 기술한다. 마지막으로 5절에서 향후 연구이슈와 함께 결론을 기술한다.

2. SCTP 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, 멀티홈잉 특성을 통해 여러 개의 IP 주소를 SCTP 세션에 바인딩 할 수 있다. [2]

그림 1은 SCTP 멀티홈잉 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. [3]

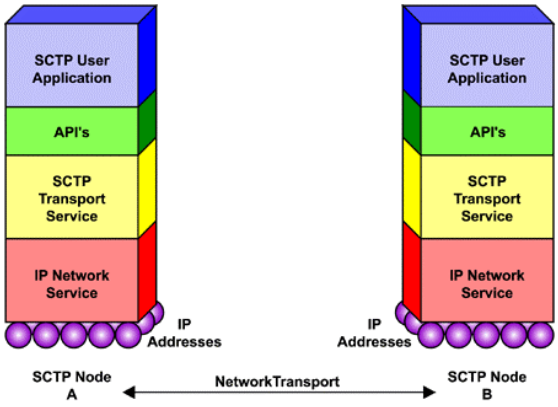


그림 1. SCTP 멀티홈잉

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

2.1 SCTP 이동성 기법

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

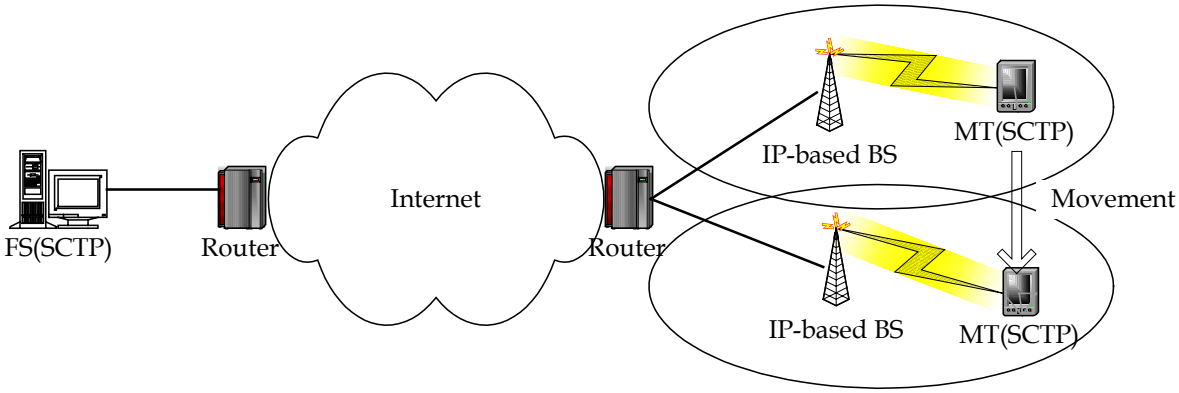


그림 2. SCTP 핸드오버 모델

그림에서 SCTP가 탑재된 MT에서 인터넷 망에 있는 FS(Fixed Server)에 연결을 설정한다. FS도 역시 SCTP를 사용한다고 가정한다. 연결설정 후에 MT와 FS간에 데이터 송수신을 수행되고, 일정 시점에 MT는 새로운 네트웍으로 진입하여 IP 주소를 할당 받는다. MT는 새로운 할당 받은 IP 주소를 SCTP 세션에 바인딩 함으로써 핸드오버를 수행할 수 있다.

만약, SCTP 연결설정이 MT가 아닌 FS에서 시작되는 경우, FS는 MT의 현재 위치(IP 주소)에 대한 정보를 필요로 하고, 이 경우 MIP와 같은 별도의 위치관리 기법이 요구된다. MIP와 함께 사용되는 경우, MIP HA는 FS에서 MT로 향하는 첫번째 SCTP INIT 패킷을 MT로 전달해주기만 하면 된다. 그 이후의 핸드오버 기능은 SCTP 핸드오버 기법에 의해서 지원될 수 있다. [5, 6, 7]

한편, 현재까지 SCTP 프로토콜은 리눅스 및 Solaris 플랫폼에서 개발되고 있으며[12, 13, 14], 사용자 레벨의 라이브러리 형태로 구현된 SCTPLIB 구현코드도 공개되어 있다. [15]

2.2 MIP와 SCTP 특성 비교

기존 MIP와는 달리 SCTP는 수송계층에서의 이동성 지원기술이다. 또한, MIP는 본래 '위치관리' 기능을 제공하는 반면에, SCTP는 '핸드오버' 지원을 위한 기술이다.

표 1은 MIP와 SCTP의 특징을 이동성 지원 관점에서 비교한다.

표 1. MIP와 SCTP의 이동성 지원 특성 비교

분류	SCTP	Mobile IP
해당 계층	수송계층(Transport)	네트웍 계층(Network)
수송계층 서비스	SCTP	TCP/UDP
위치관리	MIP 등의 별도 위치관리 기법이 요구됨	제공
핸드오버	제공	핸드오버 지원을 위해 MIP 확장 필요 (FMIP/HMIP)
경로 최적화	본질적으로 제공됨 (종단 단말간 통신)	추가 확장 기능 사용 (예: Route Optimization)
라우터 지원 (핸드오버)	필요 없음	요구됨 (라우터간 터널링)
요구사항 (보급 측면)	단말에 SCTP 탑재	라우터 및 단말에서 MIP 탑재

MIP의 경우, 핸드오버 지원을 위해 FMIP/HMIP 등의 프로토콜 확장이 요구되며, 현재까지 다양한 세부 기법들이 제안되어 왔으나 아직 표준화 및 상용화를 위해서는 좀 더 연구가 필요해 보인다. 반면에, SCTP는 IP 핸드오버를 위해 사용될 수 있으며, 위치관리가 필요한 경우 (FS에서 MT로 연결설정이 요구되는 경우), MIP 혹은 다른 위치관리 기법과 함께 사용될 수 있다.

경로 최적화(Route Optimization) 관점에서, SCTP는 본질적으로 종단간 통신 프로토콜이기때문에 최적화된 경로로 데이터가 전송되는 반면에, MIP의 경우 '경로최적화' 옵션이 요구된다. 특히, 프로토콜 보급 및 적용 관점에서, SCTP는 라우터의 별도 도움이 요구되지 않고 이동 단말에 SCTP 프로토콜만 탑재하면 되는 반면에, MIP의 경우 핸드오버를 위해 라우터간 핸드오버 터널 설정이 요구되며 이를 위한 망투자 비용이 소요된다.

종합해보면, MIP와 SCTP는 경쟁관계에 있는 기술이라기보다는, 네트웍 및 서비스 보급 상황에 따라 함께 사용될 수 있는 보완관계 기술로 볼 수 있다.

3. SCTP 핸드오버 기법

3.1 핸드오버 알고리즘 및 시나리오

SCTP를 탑재한 이동단말 MT가 외부 서버 FS와의 세션 도중에, 단말의 이동으로 인해 IP주소를 바꾸게 되는 경우 SCTP 핸드오버는 다음과 같은 절차에 따라 수행된다. [5, 6]

① MT는 L2/L3에서 새로운 영역의 Link-Up 신호를 감지하고, 새로운 IP 주소를 설정한다.

② SCTP 세션에 새로운 IP 주소를 추가(Add-IP)하고, ASCONF chunk를 FS에게 보내면, FS가 ASCONF-ACK chunk로 응답한다. 이후, MT는 기존 및 새로운 주소로 DATA를 받을 수 있다.

③ 새로운 영역의 신호 세기가 좀 더 강해지면, MT는 '주요 IP 주소변경(Primary-Change)을 위한 ASCONF chunk를 보내고, FS는 ASCONF-ACK로 응답한다. 이후, FS는 DATA chunks를 새로운 IP 주소로 보낸다.

④ MT는 기존 영역을 완전히 벗어나, Link-Down 신호를 감지한다.

⑤ MT는 기존 IP 주소삭제(Delete-IP)를 위해, ASCONF chunk를 FS에게 보내고, FS는 ASCONF-ACK chunk로 응답한다. 이후, MT는 새로운 주소만 사용한다.

한편, 상기 핸드오버 알고리즘은 네트웍 환경에 따라 다음 두 가지 시나리오에 적용될 수 있다.

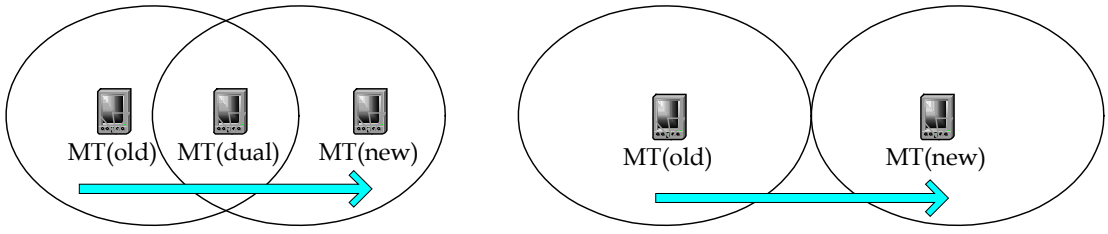
A. 시나리오 1 (그림 3(a) 참조): Dual-homing MT

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

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

3G 혹은 WiBro 등의 동일망에서의 '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.



(a) Dual-homing MT (b) Single-homing MT

그림 3. SCTP 핸드오버 적용 시나리오

3.2 핸드오버 구현을 위한 SCTP APIs

본 문서에서는 Linux 커널 2.6.8에 탑재된 SCTP 프로토콜 기능을 사용하여 SCTP 핸드오버 실험을 수행하였다[12, 13]. 리눅스 환경에서 SCTP 핸드오버를 지원하기 위해서는 다음과 같은 SCTP API 함수의 사용이 요구된다. [10, 11]

- sctp_bindx(ADD): 새로운 IP 주소를 SCTP 세션에 추가
- sctp_bindx(REMOVE): 기존 IP 주소를 SCTP 세션에 추가
- setsockopt(PRIMARY_PEER_ADDR): Primary IP 주소 변경을 상대방 SCTP에 통보

위에서 Add-IP 함수는 하위 네트웍 인터페이스 및 IP 주소 설정이 완료된 후에 수행될 수 있다. 아울러, SCTP 응용에서 하위 SCTP에서 발생하는 IP 주소 변경사항을 검사하기 위해서는, 다음과 같은 SCTP 구조체 변수 및 setsockopt() 함수를 사용해야 한다. [11]

- struct sctp_event_subscribe event;
- setsockopt(sock, IPPROTO_SCTP, SCTP_EVENTS, &event, sizeof(event));

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

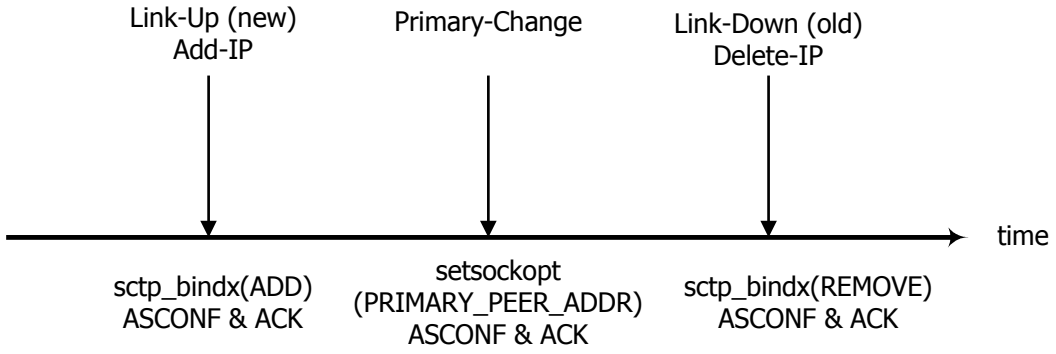


그림 4. 핸드오버를 위한 SCTP API 호출 순서

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 setsockopt() 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. 실험 및 성능 분석

본 절에서는 Linux 테스트베드 환경에서 수행한 SCTP 핸드오버 성능분석 실험결과를 기술한다.

4.1 실험 환경 및 시나리오

먼저 SCTP 핸드오버 실험을 위해 1대의 라우터와 2대의 단말기로 (각각 FS 및 MT) 구성되는 테스트베드를 구축하였다. 각 단말기에는 Linux Kernel 2.6.8 [12] 및 LK-SCTP 도구[13]를 설치하고, 이동단말 MT에는 두 개의 NIC를 탑재하였다.

실험 환경은 MT에 장착된 NIC의 활성화 여부에 따라 크게 다음 두 가지로 구분된다.

A. 실험환경 1: Dual-homing MT

MT에 두개의 NIC가 탑재되고, 동시에 두개의 NIC가 활성화된다. MT는 dual-homing 상태에서 패킷 송수신을 수행한다. 이종망간 'Vertical Handover' 상황을 연출하기 위한 것으로, 3G-WLAN 연동처럼 MT는 두 종류의 NIC를 탑재하는 것으로 가정한다.

B. 실험환경 2: Single-homing MT

MT에 두개의 NIC가 탑재되어있지만, 한 순간에 하나의 NIC만 활성화된다. 즉, MT는 single-homing 상태에서 패킷 송수신을 수행한다. 이는 동일망간 'Horizontal Handover' 상황을 연출하기 위한 것으로, L2/L3 하위계층에서 Hard Handoff가 발생하는 것으로 가정한다.

다음 그림은 실험에 사용된 테스트베드 구성도이다. 본 문서에서는 MT의 핸드오버 상황을 제한된 실험실 환경에서 구현하기 위해 매우 단순화된 네트웍 모형으로 구성하였다. FS는 192.168.0.100 주소를 사용하고, MT는 NIC별로 192.168.0.101 및 192.168.0.102 주소를 사용한다.

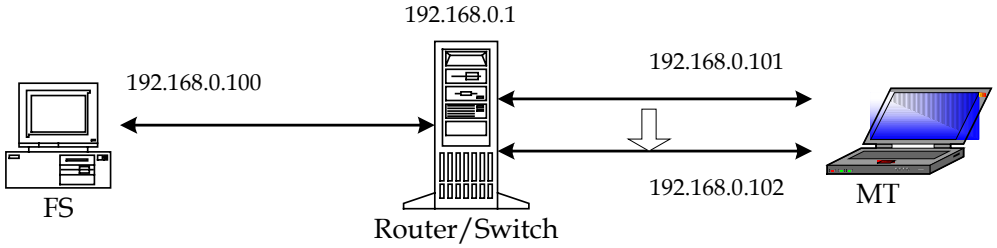


그림 5. 테스트베드 구성도

'실험환경 1(dual-homing)'에서는 중첩(overlapping)영역에서 일시적으로 두 개의 NIC가 동시에 활성화되도록 하였다. 반면에 '실험환경 2(single-homing)'에서는 비록 NIC는 두 개가 있지만, 한 순간에 하나의 NIC만 활성화되도록 하였다.

한편, 모든 실험에 공통적으로 적용된 SCTP 핸드오버 시나리오는 다음과 같다.

① 세션 초기화

먼저, FS는 고정서버로써 동작하며, '192.168.0.100' 주소를 이용하여 bind() 함수를 수행한 후에 MT의 접속을 기다린다. 처음에 MT는 '192.168.0.101' 주소를 사용하여 FS 서버에 접속한다. 두 단말간에 SCTP 세션이 설정된 후에 FS와 MT는 데이터를 교환한다.

② Add-IP

MT는 'sctp_bindx(ADD)' 함수를 사용하여 새로운 IP 주소 '192.168.0.102'를 '추가(add)'한다.

③ Primary-Change

MT는 'setsockopt(PEER_PRIMARY_ADDR)' 함수를 사용하여, FS에게 'Primary-Change'를 요청한다. 이 후에, FS는 데이터를 새로운 IP 주소 '192.168.0.102'로 전송한다.

④ Delete-IP

MT는 'sctp_bindx(REMOVE)' 함수를 사용하여 기존 IP 주소 '192.168.0.101'를 SCTP 세션에서 '삭제(delete)'한다. 이는 MT가 기존 IP 주소영역을 이탈하여 해당 NIC 인터페이스의 동작이 멈추었음을 의미한다.

⑤ 세션 종료

단말간 데이터 송수신을 완료한 후, SCTP 세션을 종료한다.

위와 같은 핸드오버 시나리오에 따라 FS와 MT 간에 SCTP 세션이 진행되며, FS는 1,000 바이트 크기의 DATA 패킷을 지속적으로 MT에 전송하고, MT 또한 소량의 데이터를 FS에 전송하도록 하였다. 기타 다른 조건은 통상적인 네트웍 프로그래밍 관례에 따라 수행되었다.

SCTP 핸드오버 성능분석을 위해 먼저 각 실험환경 및 시나리오에 대한 SCTP 패킷 교환 과정을 분석하였다. 이를 위해, 'ethereal'[16]을 사용하여 SCTP 세션에서 송수신되는 패킷들을 capturing 하였다. 아울러, 핸드오버 성능변수로써 '핸드오버 지연(handover latency)'를 측정하였으며, 본 실험에서 '핸드오버 지연시간'은 '이전 IP에서 마지막으로 데이터를 수신한 시각'과 '새로운 IP에서 처음으로 데이터를 수신한 시각'과의 차이로 정의한다[6]. 즉,

Handover Latency = '새로운 IP에서의 데이터 수신 시각' – '기존 IP에서의 데이터 수신시각'.

4.2 Dual-homing MT의 핸드오버 실험 결과

본 절에서는 MT에 두 개의 네트웍 인터페이스를 동시에 사용하는 dual-homing MT의 핸드오버 실험결과를 분석한다. 핸드오버 실험을 위해 본 문서에서 다음 사항을 가정한다.

- MT는 기존 NIC를 사용하여 SCTP 세션을 수행하면서 새로운 망에 진입한다.
- 이때, 기존 SCTP 세션을 유지하면서 '별도의 독립적인 프로세스'를 통해 새로운 망에 대한 NIC 활성화(Link-Up) 작업 및 신규 IP 주소설정 작업을 수행한다. 즉, 신규 IP 주소에 대한 설정작업이 진행중인 SCTP 세션의 처리시간에 영향을 주지 않는다.
- MT는 일정시간동안 기존 IP 주소와 신규 IP 주소를 동시에 사용하는 dual-homing 상태에서 SCTP 세션을 수행한다.

상기와 같은 실험환경에 대하여, 기존 네트웍 인터페이스의 'Link-Down' 시점에 따라, 다음 두 가지의 세부 실험 시나리오를 수행하였다.

① 시나리오 1: Link-Down이 Delete-IP 이후에 발생 (그림 6 참조)

이 시나리오는 MT의 이동속도가 느린 경우에 해당한다. MT의 이동에 따라, Add-IP, Primary-Change, Delete-IP가 수행된 후에 기존 인터페이스의 Link-Down이 발생하는 경우이다.

② 시나리오 2: Link-Down이 Add-IP 이후에 발생 (그림 7 참조)

이 시나리오는 MT의 이동속도가 매우 빠른 경우에 해당한다. MT가 Add-IP 직후에 바로 Link-Down이 발생한다. 기존 인터페이스의 Link-Down 이후, 새로운 NIC만 사용되며, Primary-Change, Delete-IP 절차는 모두 새로운 인터페이스의 IP 주소로 전송된다.

그림 6은 시나리오 1(기존 인터페이스의 Link-Down 사건이 'Delete-IP' 이후에 발생)을 보여준다. 그림에서 새로운 인터페이스의 Link-Up 직후에 Add-IP가 수행되고, 기존 인터페이스의 Link-Down 직전에 Delete-IP가 수행된다.

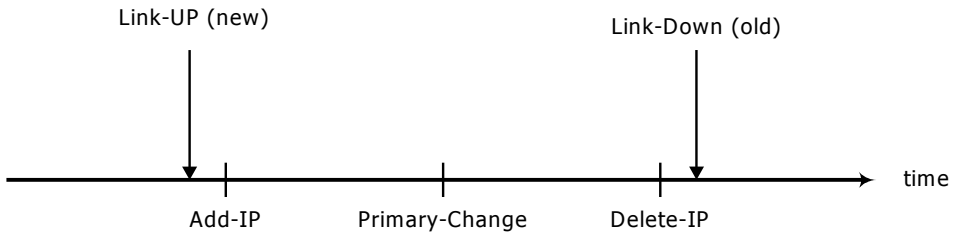


그림 6. 핸드오버 시나리오 1 (Delete-IP < Link-Down)

그림 7은 시나리오 2(MT의 빠른 이동으로 인해 Link-Up과 Link-Down이 거의 동시에 발생하는 경우)를 보여준다. Link-Up 직후에 Add-IP가 수행되고, 곧 바로 Link-Down 사건이 발생한다.

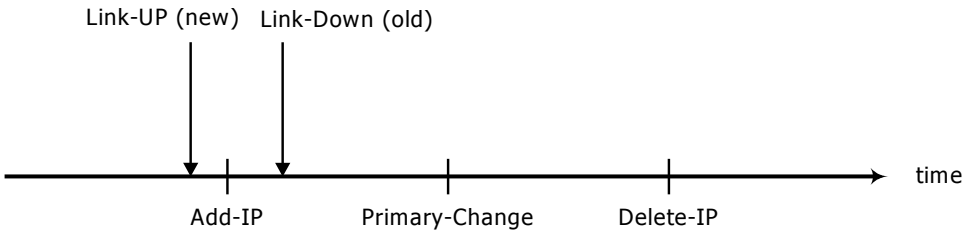


그림 7. 핸드오버 시나리오 2 (Add-IP < Link-Down < Primary-Change)

A. 시나리오 1에 대한 실험결과

그림 8은 dual-homing MT의 시나리오 1에 대한 패킷 capturing 결과이다.

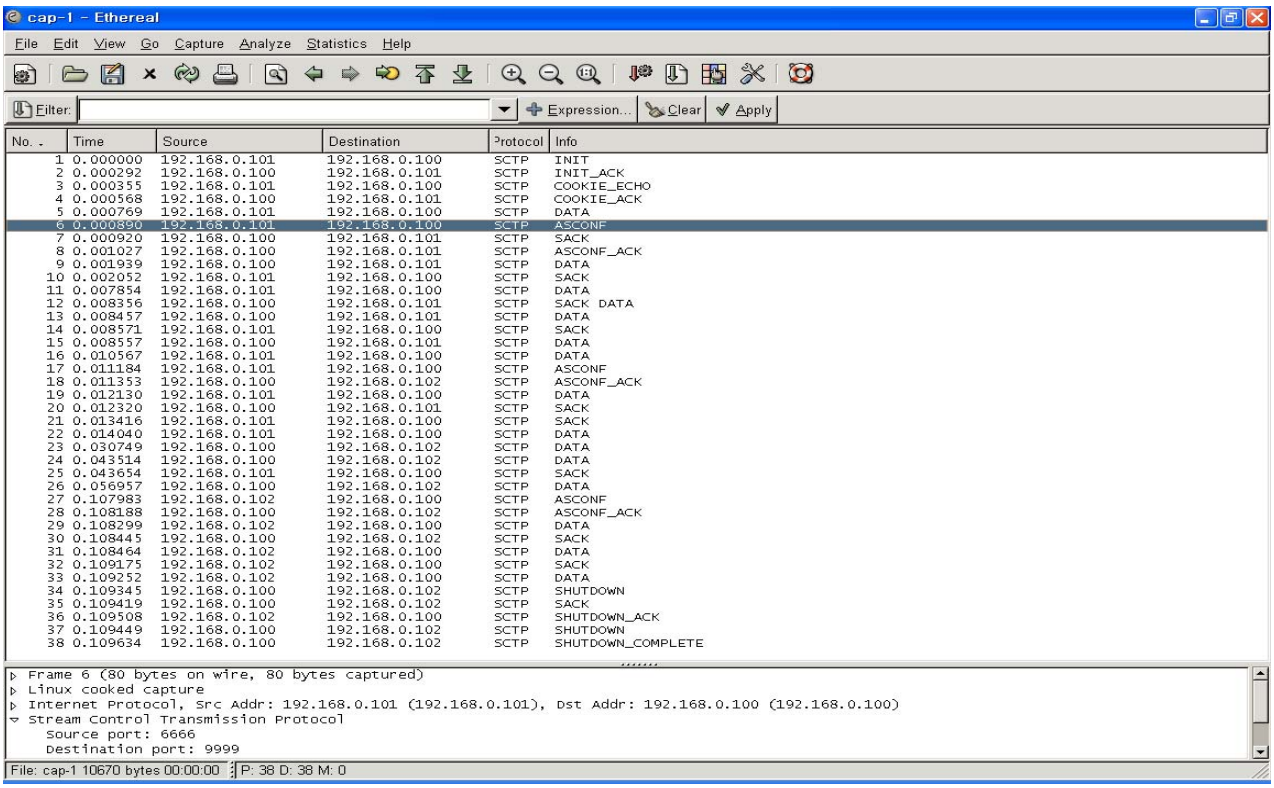


그림 8. Dual-homing MT의 핸드오버 시나리오 1 실험결과

그림에서 알 수 있듯이, FS(192.168.0.100)와 MT(192.168.0.101)은 1 ~ 4번 패킷을 통해 SCTP 세션초기화 과정을 마치고 DATA 패킷 교환을 시작한다.

6번 패킷에서 MT는 FS에게 'ADD-IP' ASCONF 패킷을 전송하여, '192.168.0.102' 주소를 추가한다. (dual-homing MT 이므로, 그 전에 두 번째 NIC가 활성화되어 있다). 8번 패킷에서 FS는 ASCONF-ACK 패킷으로 응답한다.

17번 패킷에서 MT는 FS에게 'Primary-Change' ASCONF 패킷을 전송한다. 이에 응답하여 18번 패킷에서 FS는 MT에게 ASCONF-ACK 패킷을 '새로운 주소(192.168.0.102)'로 전송한다. Primary IP 주소가 바뀌었으므로, 이 후에 FS는 MT의 새로운 주소로 DATA 패킷을 전송한다. (23번 패킷 참조)

기존 NIC의 Link-Down 직전에, 27번 패킷에서 MT는 FS에게 'Delete-IP' ASCONF 패킷을 전송하여, '192.168.0.101' 주소를 삭제한다. 28번 패킷에서 FS는 MT에게 ASCONF-ACK 패킷을 전송한다. 이후, 34 ~ 38번 패킷에서 SCTP 세션을 종료한다(37번 패킷은 35번 SACK 수신으로 인해 FS가 재전송한 패킷이다).

상기 실험결과에서 다음과 같은 흥미로운 점을 관측할 수 있었다.

a) Add-IP chunk와 Primary-Change chunk가 동일한 패킷에 piggybacking 되지 않고, 별도의 패킷으로 전송된다. 즉, sctp_bindx(ADD) 함수 호출 후에 Add-IP 패킷이 전송되고, setsockopt(PEER_PRIMARY_ADDR) 함수 호출 후에 Primary-Change 패킷이 전송된다 (6, 17번 패킷 참조). 이 점은 현재 Linux 기반 SCTP 구현 방식에 따른 문제점으로 보이며, 추후 두 가지 chunks를 piggybacking 하는 방법에 대한 연구가 필요해 보인다.

b) Primary-Change 이후에, FS는 MT에게 새로운 IP 주소로 (192.168.0.102) DATA 패킷을 전송하지만, MT는 여전히 기존 IP 주소를 Source IP 주소로 하여 FS에게 DATA를 전송한다. (19번 및 22번 패킷 참조). 사실상, SCTP의 Primary-Change 요구는 'MT'가 'FS'에게 이후 DATA를 새로운 Primary 주소로 보내달라는 의미이며, 자신의 (MT의) DATA 전송은 Kernel에 있는 인터페이스의 포워딩 테이블 설정에 의존한다.

c) Delete-IP 이후에, MT는 새로운 IP를 Source로 하여 DATA를 전송한다. (29번, 31번 패킷 참조)

그림 8의 실험결과에서 handover Latency는 15번 패킷과 23번 패킷의 시간 차이인 "0.030 – 0.008 = 0.022 (초) = 22 (ms) "로 관측되었다. 그림 8의 15 ~ 23번 사이에서 관측된 패킷 정보를 토대로 분석하면, 전체 핸드오버 지연에 소요되는 시간은 대략적으로 다음과 같이 나누어 볼 수 있다.

① Primary-Change 수행에 필요한 ASCONF/ASCONF-ACK 교환 시간 (1 RTT)

- 17, 18번 패킷

② 상대방 단말(FS)에서 바뀐 Primary IP 주소를 인식하고, DATA 전송에 바뀐 주소를 사용하도록 설정하는 시간 (SACK 처리 및 주소설정을 포함한 커널에서의 처리시간)

- 20 ~ 23번 패킷

그림 8에서는 두 번째 요소인 '커널처리시간'이 가장 큰 값으로 관측되었으나, 커널에서의 처리시간은 RTT에 관계없이 일정한 값이며, 일반적인 네트웍 상황에서 RTT 값에 비해 상대적으로 매우 작은 시간으로 볼 수 있다.

따라서, dual-homing MT의 핸드오버 지연은 "RTT + 기타 커널처리 시간"으로 볼 수 있다. 특히, RTT (Round Trip Time)은 두 단말간의 거리에 비례하여 증가하고, 일반적으로 커널처리시간보다 매우 큰 값이므로, 대략적인 핸드오버 지연시간은 Primary-Change 수행을 위한 "RTT" 값에 의존한다고 볼 수 있다.

B. 시나리오 2에 대한 실험결과

한편, 그림 9는 dual-homing MT의 핸드오버 시나리오 2(새로운 IP 주소에 대한 Add-IP 직후에 기존 NIC에 대한 Link-Down이 발생하는 경우)에 대한 패킷 capturing 결과이다.

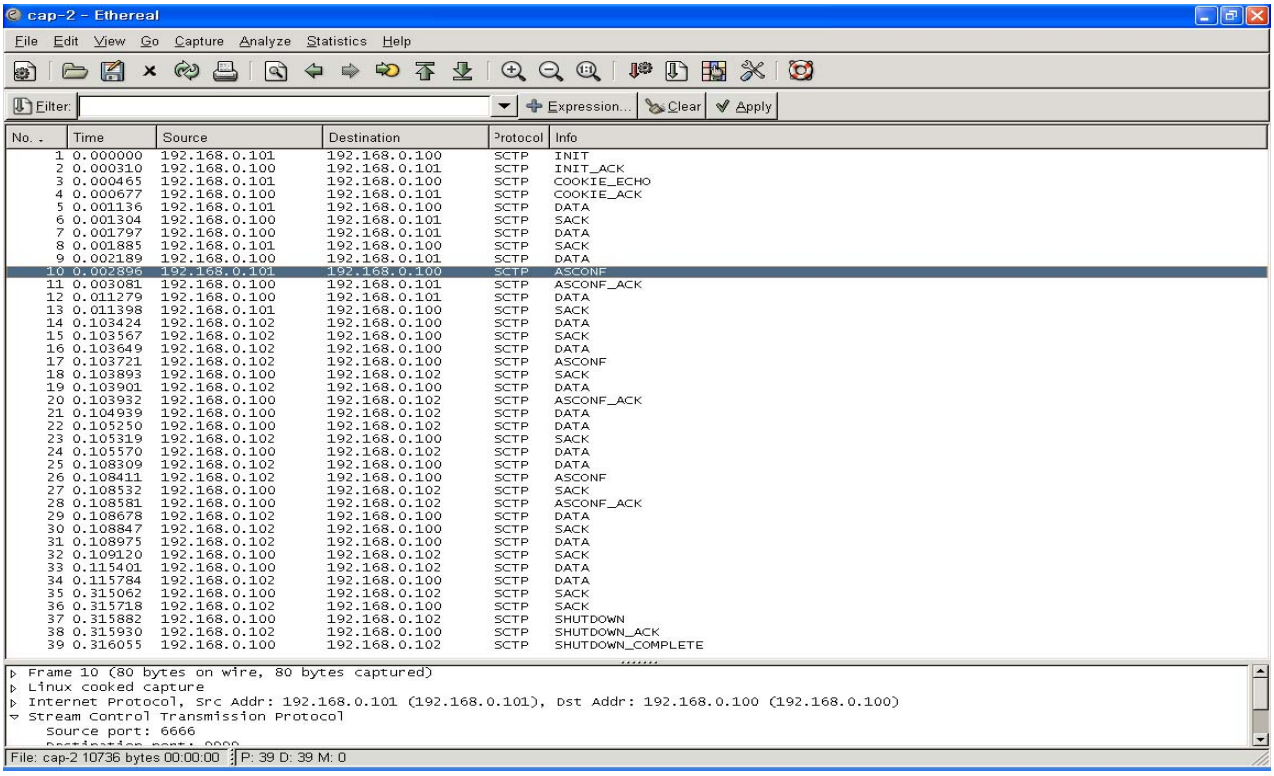


그림 9. Dual-homing MT의 핸드오버 시나리오 2 실험결과

그림 9는 그림 8과 유사한 결과를 보여주고 있으나 다음과 같은 차이점을 보인다.

a) 시나리오 1과는 달리, MT는 ADD-IP 직후에(10, 11번 패킷 참조) 기존 NIC의 Link-Down이 발생하여, FS에게 새로운 IP 주소로 DATA를 전송한다(14번 패킷 참조). 그림 8에서는 Primary-Change 이후에도 MT는 FS에게 보내는 DATA에 대하여 기존 IP를 source IP로 사용하였다.

b) 17번 패킷에서 Primary-Change를 수행한 이후에, FS와 MT간에 주고 받는 모든 패킷은 MT의 새로운 IP 주소를 사용하여 이루어진다.

한편, 그림 9의 실험결과에서 handover Latency는 12번 패킷과 21번 패킷의 시간 차이인 "0.104 – 0.011 = 0.093 (초) = 93 (ms) "로 관측되었다. 그림 8의 시나리오 1에 비해 핸드오버 지연시간이 '70 ms 정도' 증가한 이유는, 13 ~ 14번 패킷 과정에서 MT가 '기존 NIC의 Link-Down 사건을 인식하고 처리하는 데에 소요된 시간' 때문이다. 즉, MT의 Add-IP 직후에 Link-Down 사건이 발생하였고, 이를 상위의 SCTP에서 인식하고 처리하는 데에 소요된 시간이다. 만약, Link-Down 처리시간을 무시할 만한 고정된 시간으로 본다면, 시나리오 1에서처럼 핸드오버 지연은 'Primary-Change' ASCONF 패킷을 처리하기 위한 'RTT' 시간에 의존한다고 볼 수 있다.

그림 9에서 한가지 흥미로운 점은, FS가 12번 DATA 패킷을 전송한 이후에, 17번 Primary-Change ASCONF 패킷을 받기 까지 MT에게 DATA를 전송하지 못하고 있다는 점이다. MT의 Link-Down으로 인해 기존 IP로 DATA 패킷을 보낼 수 없기 때문이다. 따라서, 시나리오 2와 같이 MT의 빠른 이동속도로 인해 기존 네트웍 인터페이스의 Link-Down이 비교적 빨리 발생하는 경우에는, 되도록 Primary-Change 절차를 빨리 수행하는 것이 핸드오버 지연을 감소시키는 데에 도움이 된다는 것을 알 수 있다.

4.3 Single-homing MT의 핸드오버 실험 결과

본 절에서는 MT가 한 순간에 하나의 네트웍 인터페이스만 사용하는 single-homing 경우에 대하여 실험결과를 분석한다. 동일망에서의 핸드오버 분석을 위해 다음 사항을 가정한다.

- MT는 하나의 네트웍 인터페이스를 가지며, 한 순간에 하나의 IP 주소만 사용하는 single-homing 상태에서 SCTP 세션을 수행한다.
- MT가 SCTP 세션 도중에 새로운 망에 진입하는 경우, 하위계층(L2/L3)의 Link-Up 및 Link-Down 등을 거친 후에, DHCP 등을 사용하여 새로운 IP 주소를 얻는다. 즉, L2/L3 핸드오버 지연이 전체적인 SCTP 세션에 대한 핸드오버 지연에 영향을 준다. 위와 같은 시나리오는 3G, WLAN 등 동일망에서 이동하는 단말에 적용될 수 있다.

위와 같은 시나리오를 실험하기위해, MT의 핸드오버 시점에 Link-Up(new), Add-IP, Link-Down(old) 사건이 거의 같은 시점에 일어나도록 설정하였다. 즉, SCTP 세션 도중에 핸드오버로 인한 Link-Up 및 Link-Down 처리시간이 핸드오버 지연에 영향을 준다.

그림 10은 상기와 같은 single-homing MT의 핸드오버에 대한 패킷 capturing 결과이다.

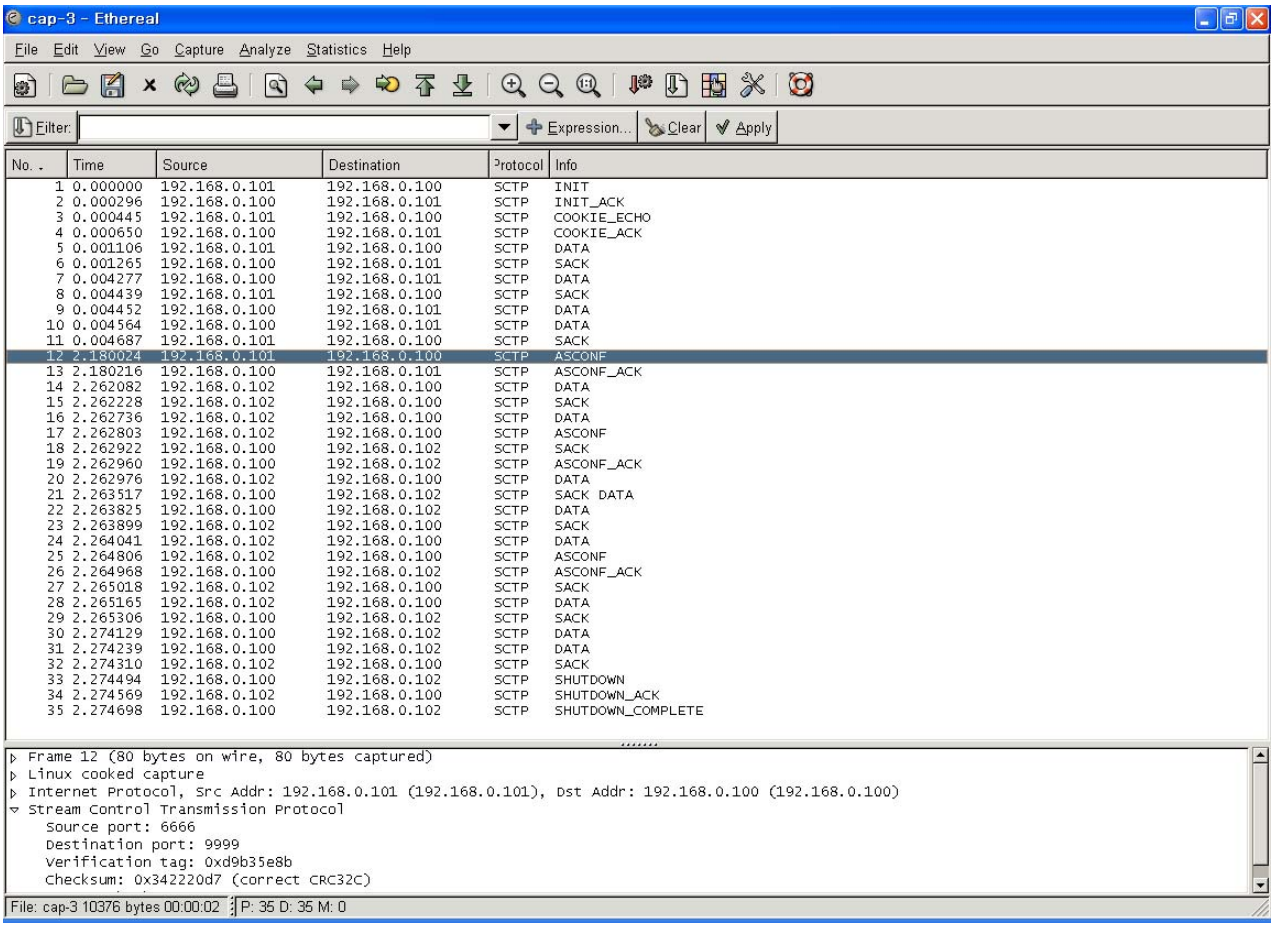


그림 10. Single-homing MT의 핸드오버 실험 결과

그림 10의 single-homing MT 실험결과는 이전에 수행한 dual-homing MT 실험결과와 대체적으로 유사하지만, 다음과 같은 특징을 보여준다.

a) 본 실험에서는 Add-IP 패킷 전송 시점에서 '새로운 NIC의 활성화' 및 '기존 NIC의 비활성화' 작업이 함께 수행되고 있다. 11번 패킷과 12번 패킷간의 시간간격에서 알 수 있듯이, 본 실험의 경우 Linux 환경에서 NIC 활성화/비활성화에 대략 2.2초의 시간이 소요됨을 알 수 있다.

b) 사실상, 17, 19번의 Primary-Change를 위한 ASCONF 패킷들과, 25, 26번의 Delete-IP를 위한 ASCONF 패킷들은 별 의미가 없다. 즉, 그와 관계없이 FS와 MT는 새로운 IP만을 사용하여 SCTP 데이터 통신을 수행한다.

c) 핸드오버 지연시간은 10번 패킷과 21번 패킷의 시간 차이인 "2.263 – 0.004 = 2.259(초)"이다. 거의 대부분의 시간이 상기한 NIC 활성화/비활성화 시간이다. 즉, Primary-Change를 위한 ASCONF 패킷 교환시간(RTT)은 상대적으로 매우 작다.

d) 13번 패킷 이후에 MT의 기존 IP 주소는 사용할 수 없으므로, FS는 MT로부터 Primary-Change ASCONF 패킷이 올 때까지 DATA 패킷 전송을 하지 않는다. 즉, FS는 Primary-Change 패킷 혹은 Delete-IP 패킷이 도착할 때 까지, DATA 전송을 보류하고 있다. 따라서, single-homing MT의 경우, Primary-Change를 빨리 수행하는 것이 핸드오버 지연 단축에 도움이 된다.

Single-homing MT의 핸드오버에 대한 위와 같은 실험결과로부터, SCTP 핸드오버 지연은 하부의 네트웍 인터페이스의 활성화 및 비활성화 시간에 큰 영향을 받음을 알 수 있다. NIC의 활성화 과정에는 링크계층의 Link-Up/Link-Down 뿐만 아니라, IP 계층의 주소설정 (DHCP 등) 및 AAA 인증 과정도 포함된다.

실제로, '하부 계층의 핸드오버 지연이 전체적인 SCTP 핸드오버 지연에 미치는 영향'을 분석하기 위해, 본 문서에서는 '네트웍 인터페이스 Up/Down 시간'을 인위적으로 1초씩 증가시키며 실험을 수행하였다. 실험결과, 상기한 예측대로 SCTP 핸드오버 지연이 '하위 계층 인터페이스 활성화' 시간에 비례하여 증가함을 확인할 수 있었다.

그림 11은 그 중 한 예로써, 'L2/L3 핸드오버' 시간을 대략 30초로 설정한 경우에 대한 결과이다.

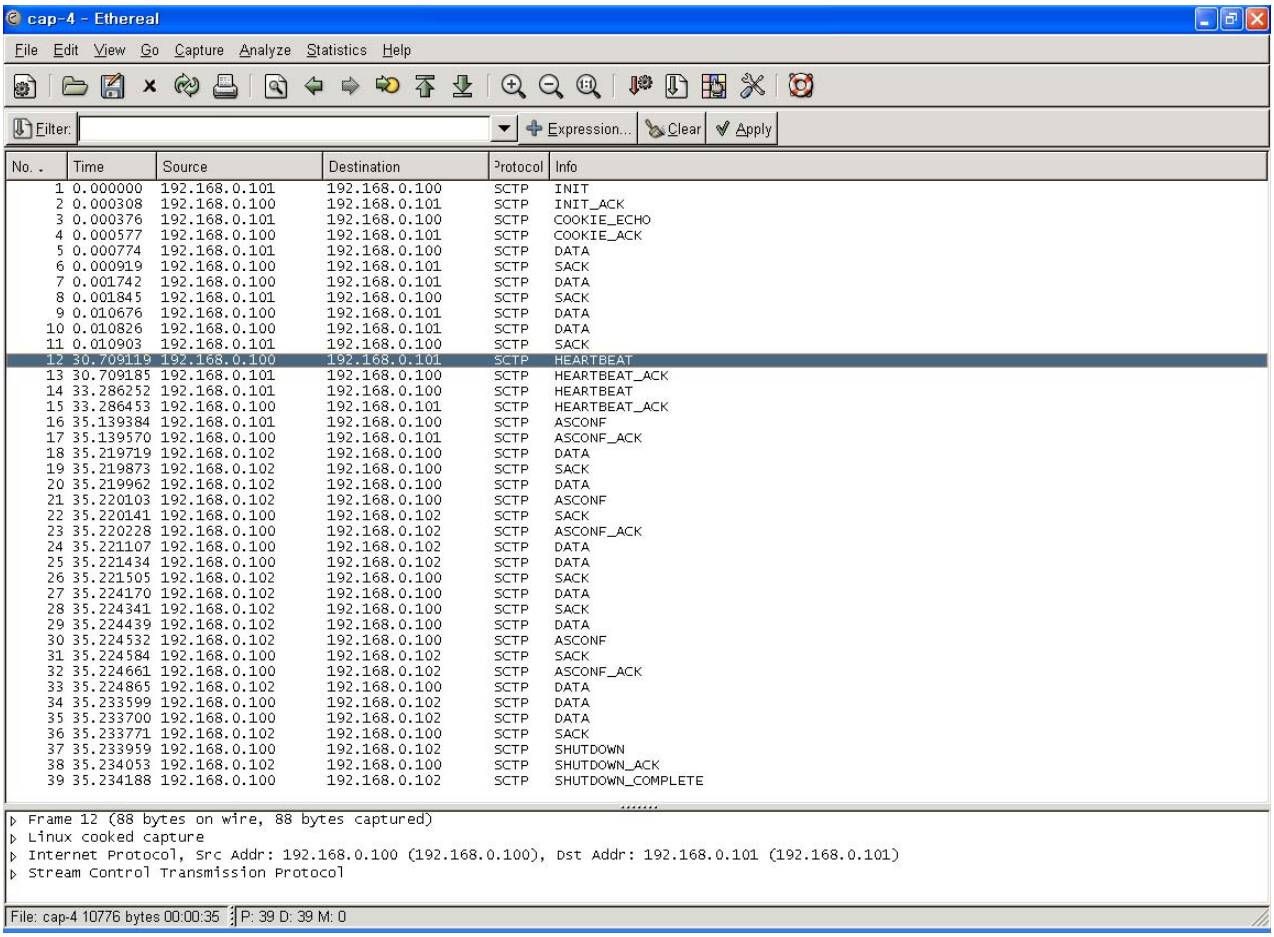


그림 11. Single-homing 핸드오버 (L2/L3 핸드오버 = 30초)

그림에서 볼 수 있듯이, 11번 패킷 이후에 L2/L3 핸드오버가 30초 정도 진행되다가, 35번 패킷에서 새로운 주소가 FS에게 전달되고 있다. 한편, 각 IP 주소에 대한 유지관리를 위해서 SCTP 프로토콜에 따라 HEARTBEAT 패킷이 30초 주기로 FS와 MT간에 교환되고 있다. 한편, 24번 패킷에서 MT는 새로운 주소로 DATA를 전송하며, 핸드오버 지연은 10번 패킷과의 시간 간격인 35초 정도가 소요된다.

한 가지 주목할 만한 점은, Single-homing으로 인한 네트웍 인터페이스 활성화/비활성화 지연이 SCTP 세션 자체의 중단을 초래하지는 않는다. 즉, SCTP 단말간의 통신은 종단간 SCTP 프로토콜 메커니즘에 의해 유지된다.

한편, 그림 12는 그림 10에 대한 실험에서 Primary-Change 발생 시간을 인위적으로 지연시킨 실험에 대한 결과이다. 그림 10에서는 17번 Primary-Change 패킷이 2.2초 경에 발생되는 데 비해, 그림 12에서는 25번 Primary-Change 패킷이 3.2초 경에 발생되고 있다.

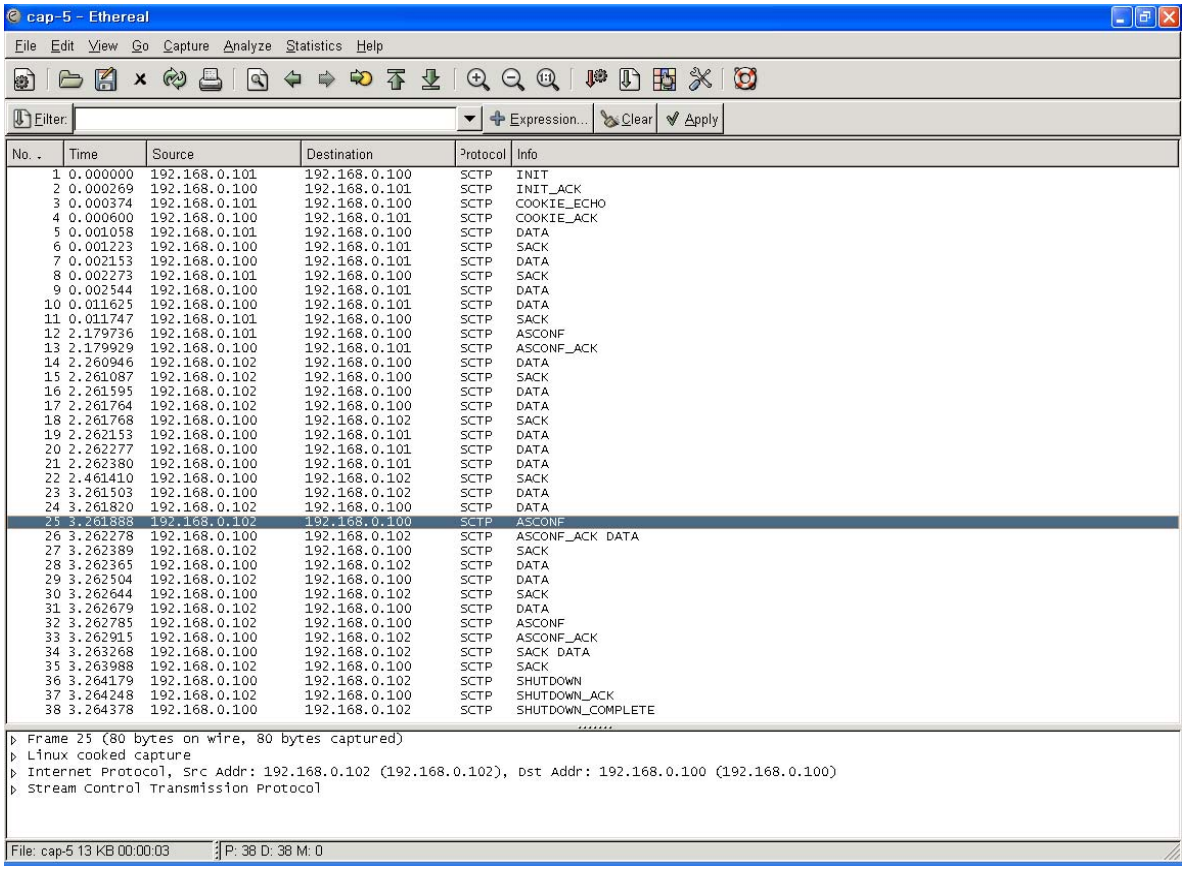


그림 12. Primary-Change 지연에 따른 SCTP 핸드오버

그림에서, 이전 주소에 대한 Link-Down이 발생했음에도 불구하고, Add-IP 이후에 Primary-Change가 완료되기까지 FS는 이전 주소(192.168.0.101)로 DATA로 전송하고 있다. (19번 패킷 참조). 실제 19번 DATA 패킷은 손실되었으며, Primary-Change 시점 이후인 28번 패킷에서 DATA가 재전송됨을 확인하였다. (20, 21번 패킷은 동일한 방식으로 복구되나, 본 실험에서는 그 전에 세션을 종료하였음).

지금까지의 single-homing MT의 핸드오버 실험 결과, 핸드오버 지연은 하위 계층의 Link-Up 및 Link-Down 소요 시간에 크게 영향을 받음을 알 수 있었고, 상대적으로 ASCONF 패킷 교환에 소요되는 RTT 시간은 무시할 만 한다. 또한, Single-homing MT의 경우, Add-IP 직후에 바로 Primary-Change 절차를 수행하는 것이 핸드오버 지연 감소에 바람직해 보인다. 한편, 하위 L2/L3 계층의 핸드오버 시간이 매우 긴 경우에도, SCTP 세션의 종료에는 영향을 주지 않고 SCTP 프로토콜 메커니즘에 의해 Soft Handover가 지원된다.

5. 결론 및 향후 연구

지금까지 리눅스 플랫폼에서 SCTP 기반 핸드오버 기법에 대한 성능분석에 대하여 기술하였다. 소규모 실험실 규모의 테스트베드를 구축하여, 이종망간 핸드오버 시나리오 및 동일망간 핸드오버 시나리오에 대한 SCTP 핸드오버 동작 패턴을 분석하고 핸드오버 지연 관점에서 성능을 비교하였다.

실험결과, 이종망간 dual-homing SCTP 핸드오버의 경우, 핸드오버 지연은 이동단말과 외부 단말간의 RTT 시간에 영향을 받는다. 본 실험의 경우, RTT 값이 매우 작아 전체 핸드오버 과정에 1초 이내에 완료되었다. 한편, 동일망간 single-homing 핸드오버의 경우, 핸드오버 지연시간은 RTT 보다는 하위 링크/네트웍 계층의 활성화 시간에 크게 영향을 받는 것을 확인하였다. 특히, 이 경우에는 Add-IP 직후에 바로 Primary-Change를 수행하는 것이 핸드오버 지연 감소에 도움을 준다. 주목할 만한 점은 하위 계층의 활성화 시간에 관계없이 SCTP 세션은 SCTP 프로토콜 동작방식에 따라 연속성이 유지된다는 점이다.

향후 연구로써, 본 연구에서 수행한 실험을 좀 대규모의 테스트베드 환경 혹은 실제 인터넷 망에서 수행할 필요가 있으며, 기존 MIP와의 구체적인 연계방안에 대한 실험, 분석 및 검증작업도 고려해 볼 만하다.

참고 문헌

[1] ITU-T Supplement to Recommendations Q.sup52, “Technical Report on Mobility Management Requirements for Systems Beyond IMT-2000”, 2005

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

[3] Stewart, R., et al., "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", IETF Internet Draft, draft-ietf-tsvwg-addip-sctp-08.txt, June 2004

[4] 송정화 외, "SCTP의 멀티호밍 특성에 대한 성능 평가,” 정보처리학회문서지(C), 제11-C권 제2호, pp. 245 ~ 252, 2004년 4월

[5] 장문정 외, "mSCTP를 이용한 종단간 이동성 지원 방안," 정보과학회문서지:정보통신, 제 31권 제 4호, pp. 393 ~ 404, 2004년 8월

[6] Chang M., et al., "Transport Layer Mobility Support Utilizing Link Signal Strength Information", IEICE Transations on Communications, Vol. E87-B, No. 9, pp. 2548-2556, September 2004.

[7] Chang M., et al., "A Transport Layer Mobility Support Mechanism", LNCS 3090, Vol. 3090, pp. 287 - 296, May 2004.

[8] Network Simulator 2, Available from http://www.isi.edu/nsnam/ns/

[9] ns-2 modules for SCTP, Available from http://www.cis.udel.edu/~iyengar/research/

[10] Stewart, R., et al., “Sockets API Extensions for Stream Control Transmission Protocol”, IETF Internet Draft, draft-ietf-tsvwg-sctpsocket-09.txt, September 2004

[11] Stevens R. et al., Unix Network Programming: The Sockets Networking API, Vol. 1, 3rd ed., 2004

[12] Linux Kernel Archives, Available from http://www.kernel.org/

[13] Linux Kernel SCTP Project, Available from http://lksctp.sourceforge.net/

[14] Solaris SCTP, Available from http://playground.sun.com/sctp/

[15] SCTPLIB, Available from http://www.sctp.de/sctp-download.html

[16] Ethereal, Available from http://www.ethereal.com/