SHIM6: 인터넷 주소의 Identifier/Locator 분리 기술

2008년 1월

경북대학교 통신프로토콜연구실

김지인 (jiin16@gmail.com)

요 약

현재의 인터넷은 이동성 지원, 멀티호밍 환경 등으로 인해 그 아키텍처 변경이 불가피 하다. 그 변화 중 하나로 인터넷 주소의 Identifier/Locator 분리 기술이 있다. 이 기술에는 VIP, LINA, HIP등이 있다. 하지만 이들은 기존 단말간의 호환성을 보장하지 않는다. 이 단점을 보완한 Shim6를 소개하고 test를 수행해 보았다.

목 차

1. 서론

현재 인터넷은 1969년 ARPANet을 기반으로 현재의 OSI 7 계층 모델에서 TCP/IP를 사용하는 형태로 발전해 왔다. 하지만 2000년에 들면서 빠른 이동 단말의 지원, 다중 인터페이스 단말의 등장, 다양한 무선구간의 확장, 센서와 같은 다량의 단말 접속, 안전한 전자거래, 네트워크에서의 서비스 품질 보장, 비즈니스 측면 보완 등 현재의 인터넷 기술에 대한 문제 제기 및 이를 보완, 대체하기 위한 다양한 연구들이 진행되어 왔다. 이러한 현재의 인터넷의 문제를 보완, 해결하기 위해 현재의 인터넷의 철학을 뛰어넘는 보다 새로운 인터넷에 대한 연구가 필요한 실정이다.

이에 대한 현재 진행되고 있는 대표적인 연구들을 살펴보면, 최근 인터넷 상의 가장 큰 문제인 routing과 addressing의 확장성을 높이는 연구가 진행되고 있다. 이동성, 멀티호밍, PI(Provider Independent) routing 등의 증가로 인한 routing 테이블 엔트리의 증가는 현재 인터넷에서 가장 시급히 개선되어야 할 문제 중 하나로 인식되고 있다. 이러한 문제는 현재 사용되는 IP 어드레스에 identifier와 locator 기능이 함께 사용되기 때문이며, 이를 해결하기 위해서는 현재 어드레스 개념에서 identifier와 locator를 분리하거나, 혹은 identifier를 근본적으로 다시 설계하려는 연구가 진행 중에 있다. Identifier/Locator 분리 작업에 대한 연구 및 표준화는 IETF RoAP BoF 등을 통해 추진 중에 있다.

2. Identifier/Locator 분리

현재 인터넷에서의 핵심 요소인 IP 주소는 네트워크에 연결되어 있는 어떤 device의 interface에 대한 유일한 식별자(identifier) 역할을 하며, 동시에 네트워크 상의 device의 위치에 대한 식별자(locator) 역할을 모두 하고 있다. 하지만 현재의 네트워크 환경은 이동성을 지원함으로써 단말의 위치와 상관없이 단말을 식별할 수 있어야 한다. 하지만 현재 사용하는 IP 주소의 경우 단말을 인식하는 identifier와 위치를 인식하는 locator의 의미가 함께 사용되기 때문에 단말이 이동할 경우 IP 주소가 바뀌면서 단말의 identifier도 바뀌면서 통신 중 세션이 끊어져 연속적인 서비스(Seamless service)를 제공할 수 없다.

그리고 멀티호밍 환경(네트워크 등이 다중 IP 주소를 사용하여 동종 또는 이종 링크와 다중으로 접속을 유지하는 기술)을 지원하기 위해서는 단말의 identifier는 하나이지만 동적으로는 두 개 이상의 locator가 매핑되어야 한다. 이 경우 단말의 위치가 이동을 하더라도 identifier는 바뀌지 않고 locator만 바뀌면서 연속적인 서비스를 제공할 수 있다.

OSI 7 layer에서 IP 계층과 Transport 계층 사이에 Identity 라는 계층을 집어 넣어 여기에서 Transport 상위 계층에서 사용하는 identifier와 IP 하위 계층에서 사용하는 locator를 매핑하도록 한다. 이로 인해 단말의 이동으로 IP 주소가 바뀌어도 Identity 계층에서 변경된 IP 주소(locator)와 동적인 매핑이 이루어져 실제 상위 프로토콜의 통신 세션은 아무 변화가 없어 연속적인 서비스(Seamless service)를 제공할 수 있다.

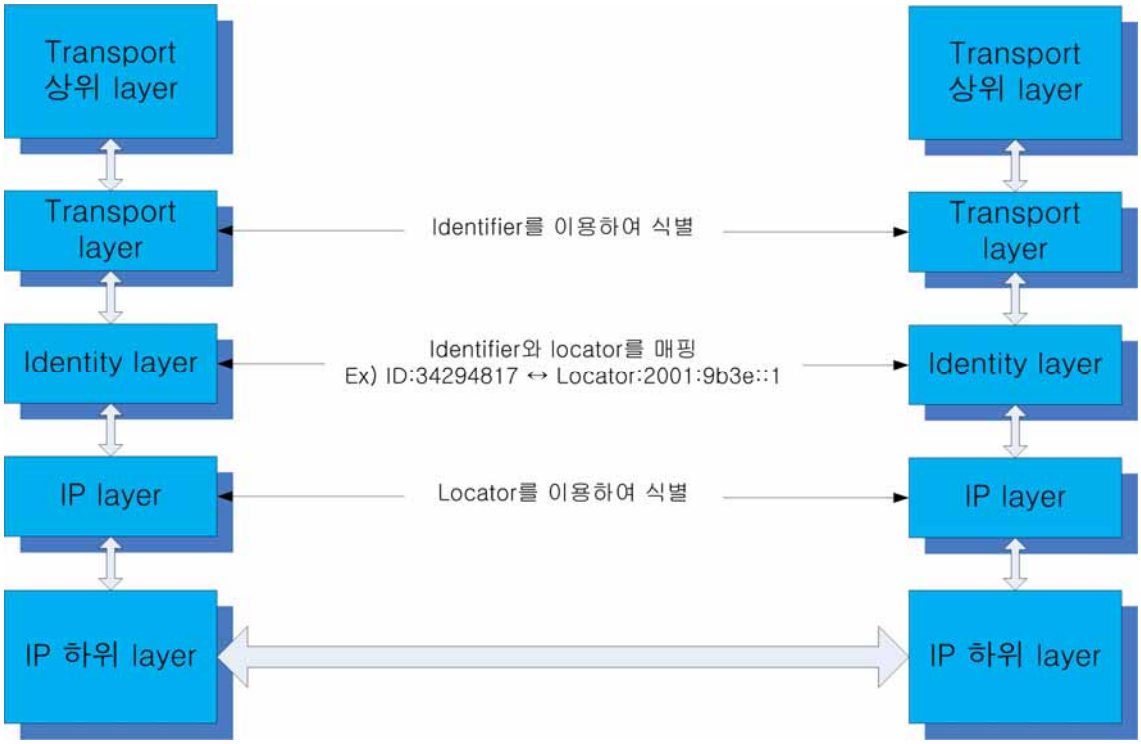

그림 1. Identifier와 locator 분리

2.1 VIP(Virtual Internet Protocol)

VIP는 1994년 차세대 IP인 IPv6가 한창 논의되고 있을 당시 IPv4에서 host의 이동성을 고려한 프로토콜로 처음 제안됐다. VIP는 Virtual Network라는 개념을 도입한다. host는 Transport 계층에서 Virtual Network를 새로이 정의한다. 이 Virtual Network에서도 실제 Network와 마찬가지로 IP 주소를 지정하는데 이를 VIP address라 한다. host는 실제 Network에서는 이동을 하지만 이 Virtual Network에서는 이동을 하지 않으며 항상 같은 Virtual Network에 접속해 있다. 따라서 host의 실제 Network주소는 host의 위치를 반영하여 변화하지만 VIP address는 변화하지 않는다. Transport 계층 상위 계층에서는 이 변화하지 않는 VIP address를 이용하여 서비스를 하므로 연속적인 서비스(Seamless Service)를 제공한다.

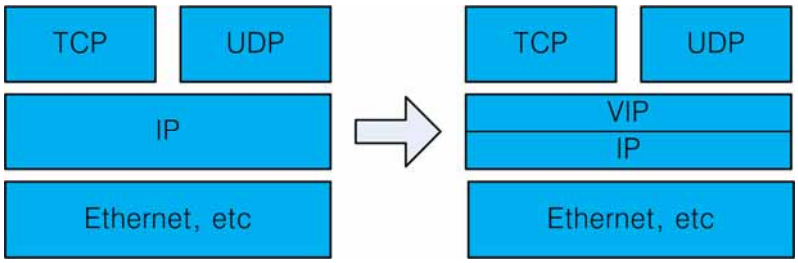

그림 2. VIP의 프로토콜 계층

하지만 실제로는 그림 2와 같이 Transport 계층과 IP 계층 사이에 VIP 계층을 두어 VIP address와 실제 IP address를 매핑하도록 정의하였다. VIP의 가장 큰 단점은 VIP를 지원하지 않는 host들과의 호환성이 없으며 현재 인터넷 아키텍처의 수정이 불가피하다는 것이다.

2.2 LINA(Location Independent Network Architecture)

LINA구조는 1999년 IPv6 환경에서 멀티호밍 환경과 이동성을 지원하기 위해 처음 제안되었다. 이 구조를 위해 프로토콜 LIN6가 개발되었다. LIN6에서는 VIP와는 달리 IP 계층 위에 VIP 계층이 존재하여 identifier와 locator를 매핑하는 것이 아니라 IP 계층을 identification 서브계층과 delivery 서브계층으로 나누어 상위의 identifier와 하위의 locator의 매핑을 담당하게 한다.

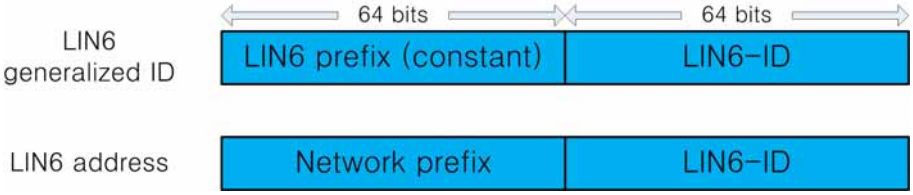

그림 3. LIN6 generalized ID와 LIN6 address

LIN6에서의 주소는 그림 3과 같이 LIN6는 각각의 node마다 interface id가 아닌 node id를 부여한다. 각각의 node는 interface의 수에 관계없이 유일한 하나의 node id에 의해 식별된다. IPv6 계층에서, 이 node id는 LIN6 ID라 불리며 IPv6에서 주소 길이는 128 bits이므로, 앞에 64bits LIN6 prefix를 붙여 128bits로 만들었으며, 이는 Transport 계층 위에서 identifier로 LIN6 generalized ID로 사용된다. node의 interface마다 LIN6 address가 부여되며, LIN6 address는 IPv6의 network prefix 64bits와 LIN6 ID 64bits를 합쳐 128bits IPv6 네트워크 주소인 locator로 IP 계층 아래에서 사용된다.

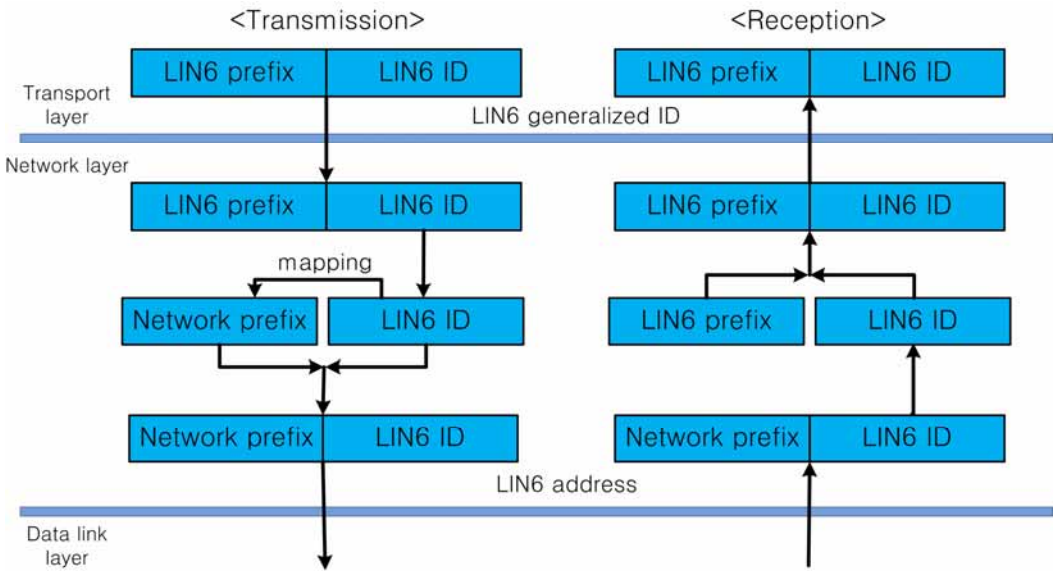

그림 4. LIN6 Address processing

LIN은 그림 4에서처럼 기존의 아키텍처의 변화가 없으며, 현재 WIDE 프로젝트의 일환으로 지속적인 연구가 진행되고 있고, 이미 FreeBSD에서는 구현되어 있는 상황이다.

2.3 HIP(Host Identity Protocol)

HIP은 IP layer에서 identifier와 locator 분리뿐만 아니라 보안성까지 지원하는 프로토콜이다. HIP에서 identifier는 public-private key 쌍이다. host가 private key를 처리할 때, public key가 나타나면서 식별된다. public key인 Host Identity(HI)는 128 bits 값으로 이를 hash함으로써 Host Identity Tag(HIT)가 생성된다. HIT가 128 bits이기 때문에 IPv6 주소 길이와 같다. 따라서 transport 상위 계층에서 주소 개념으로 사용된다. public key와 hash를 사용하여 보안성은 우수하나 HIP은 기존의 아키텍처가 아닌 새로운 아키텍처의 도입을 제안하고 있다.

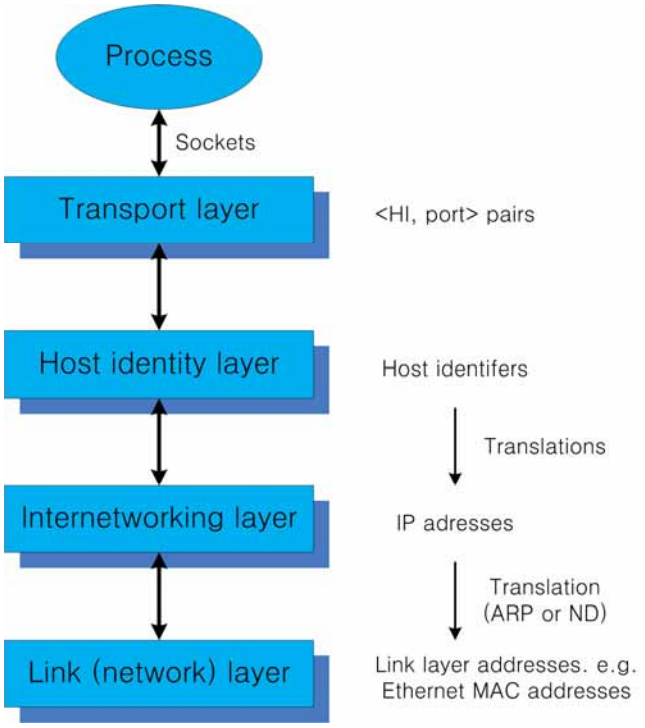

그림 5. HIP new architecture

그림 5는 HIP에서 제안하는 새로운 아키텍처이다. 기존의 IP 계층을 internetworing 계층으로 바꾸고 Host identity 계층을 추가한 형태이다. Host identity 계층에서는 identifier인 HIT와 loactor를 매핑하는 역할을 담당한다. 하지만 HIP은 기존의 단말과 호환이 되지 않으며, DNS의 확정과 새로운 기능을 하는 Rendezvous Server(RVS) 등의 에이전트가 추가되어야 하는 단점이 존재한다.

3. L3Shim(Network Layer Shim)

L3Shim은 identifier와 locator를 분리하는 기술로 기본적인 개념은 HIP와 동일하다. 하지만 별도의 identifier와 locator를 사용하지 않고, 그냥 기존의 IP address를 그대로 사용한다. 따라서 VIP, LINA, HIP의 단점인 기존 단말과의 호환성 문제를 해결한다. 이를 위해서 우선 shim은 처음 통신이 설정되는 순간 host의 identifier와 locator를 동일한 IP address로 설정한 후 서로 Shim을 지원하는지 context를 교환을 실시한다. context를 교환 시에는 상대방 host의 locator 정보를 가져온다. Shim을 지원하는 경우 통신 경로에 문제가 생길 경우, 다른 locator로 통신을 변경한다. 이 과정에서 그림 6과 같이 shim은 IP 계층에 shim6 계층을 두고 identifier와 locator를 매핑하는 역할을 수행한다.

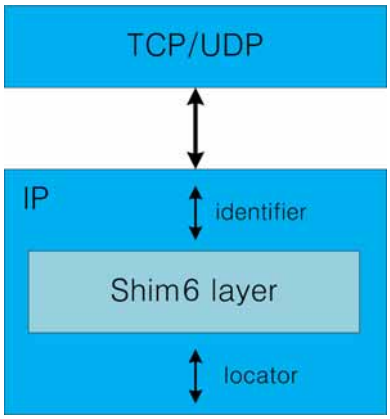

그림 6. shim6 architecture

shim은 현재 UCL(Universite Catholique de Louvain), ETRI(한국 전자통신 연구원) & SNU(Seoul National University)에서 구현 중이다. ETRI shim6 구현의 경우 phase 2까지 구현이 되었다. phase 1은 netfilter와 iptable을 이용하여 packet을 그대로 복사한 후 다른 interface로 전달하는 방법을 사용하였고, phase 2의 경우 직접 socket API를 만들어 kernel에서 packet을 복사하여 interface로 전달을 한다. 당연히 phase 1보다는 phase 2가 shim6에 더 적합하다. 아래의 test의 경우 phase 2를 이용하여 실행되었다.

4. Shim Test

4.1 Test 시나리오

A. Shim 실행 점검: iperf 이용
iperf는 보통 네트워크의 대역폭 측정을 위해 이용이 되지만 여기에서는 멀티호밍 환경에서 iperf를 이용해 임의로 TCP 연결을 만든 후 한 쪽 link를 down할 경우 다른 link를 이용해 shim이 연결을 복구하는지 알아본다.

B. Shim 연결성 test: VLC 플레이어 스트리밍 서비스를 이용
VLC는 동영상 플레이어이다. 여기에서는 멀티호밍 환경에서 스트리밍 서비스를 통해 동영상을 보던 중 link를 down할 경우 다른 link를 이용해 shim이 연결을 복구하여 서비스 제공에 무리가 알아보기 위해 사용한다. 그리고 shim 동작을 wireshark를 이용한 packet capture를 통해 알아본다.

4.2 Test 환경

모든 test는 Fedora Core 6에서 이루어졌으며, Kernel은 netfilter를 이용하기 위해 기존 2.6.18 버전이 아닌 2.6.19.7버전을 사용하였다. 그리고 shim을 test하기 위해서는 IPv6망에서 동작을 하므로 반드시 IPv6망으로 연결이 되어야 하며, 멀티호밍 환경을 구축해야 한다. Shim 설치 방법은 Appendix A를 참고하길 바란다. 이 test는 ETRI와 합동으로 수행하였으며, ETRI에서 수행한 내용은 본 문서에서 제외하였다.
Test를 수행하기 위해서는 iperf, VLC 플레이어, Wireshark가 필요하다. 이는 Fedora Core 6에서 rpm으로 제공하므로 yum을 통해 설치가 가능하다.

- VLC 플레이어: $rpm –Uvh http://rpm.livna.org/livna-release-6.rpm
  $yum install vlc
- Wireshark: $yum install wireshark
  $yum install wireshark-gnome
- iperf: $yum install iperf

Shim이 작동할 수 있도록 설정을 해야 한다. shim이 설치된 폴더에 locator_list.txt 파일을 열어 두 개 이상의 IPv6 주소를 입력한다. 그리고 iptable을 설정하고 shim을 위한 special device를 생성한다.

```
$iptables –t mangle –A INPUT –j L3SHIM_IN
$iptables –t mangle –A OUTPUT –j L3SHIM_OUT
$mknod /dev/shim6_dev c 1052 0
```

4.3 Test 수행

A. Shim 실행 점검

$./shim6

그림 7. Shim 실행화면

- 터미널 창을 열어 그림 7과 같이 shim을 작동시킨다. 다른 터미널 창을 열어 다음 명령어를 수행해 준다.

$iperf –V –s –i 1

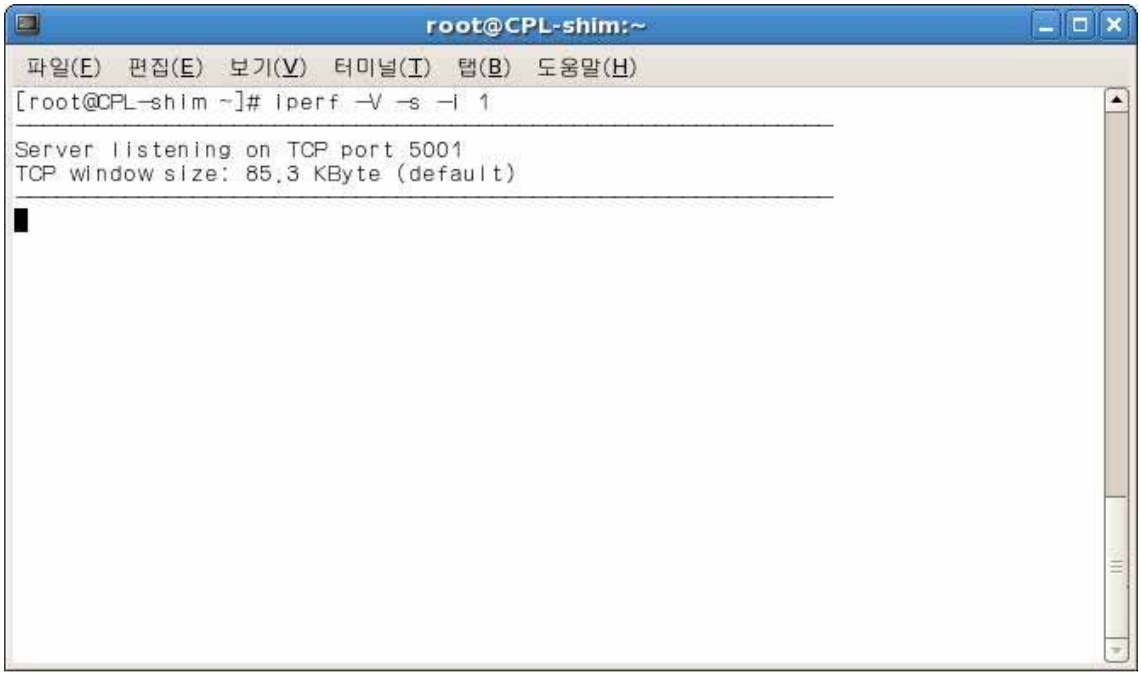

그림 8. iperf 실행화면

- 그림 8은 서버 모드로 iperf가 동작하는 화면이다. TCP 연결을 만든다. 그리고 상대방에서 iperf를 client로 접속한다. 아래 그림 9와 같이 iperf이 동작하는 것을 확인 할 수 있다.

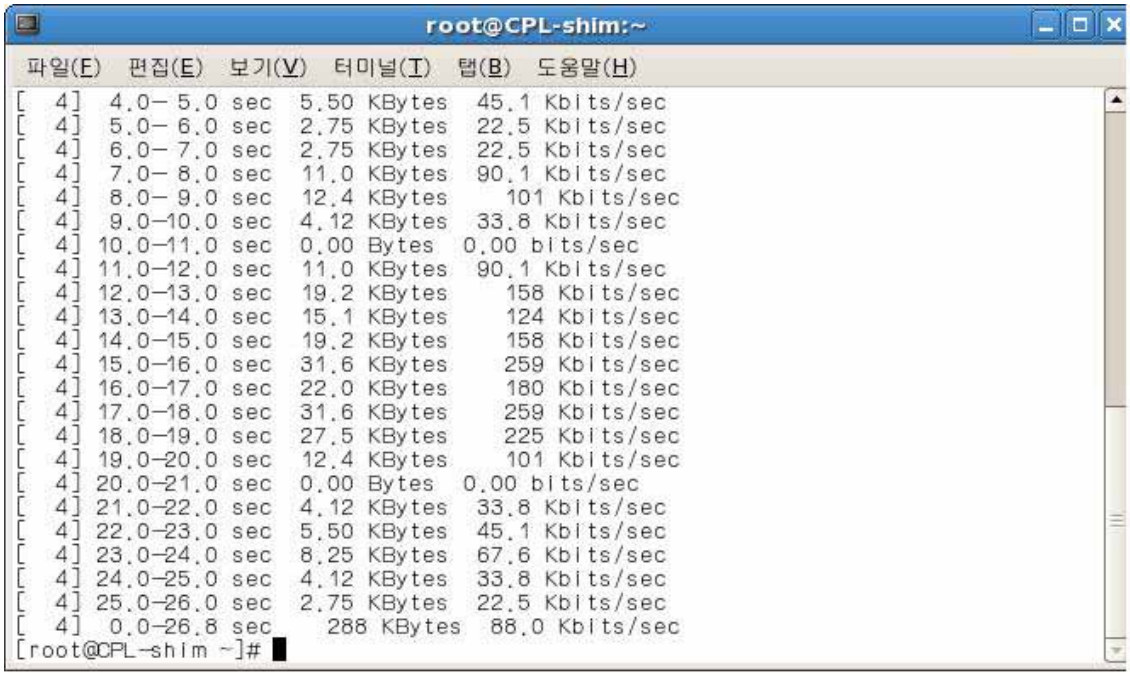

그림 9. iperf 동작화면

B. Shim 실행 점검

test를 수행하기 전 packet capture를 위해 wireshark를 실행한다. [Capture] 메뉴에서 [Interface]를 선택한 후 그 곳에서 any device를 선택해 모든 device에서 나오는 packet을 capture한다. 그리고 VLC를 실행하여 client로 네트워크 스트림을 열어 연결한다. 연결을 하면 아래 그림 10과 같이 shim이 동작한다.

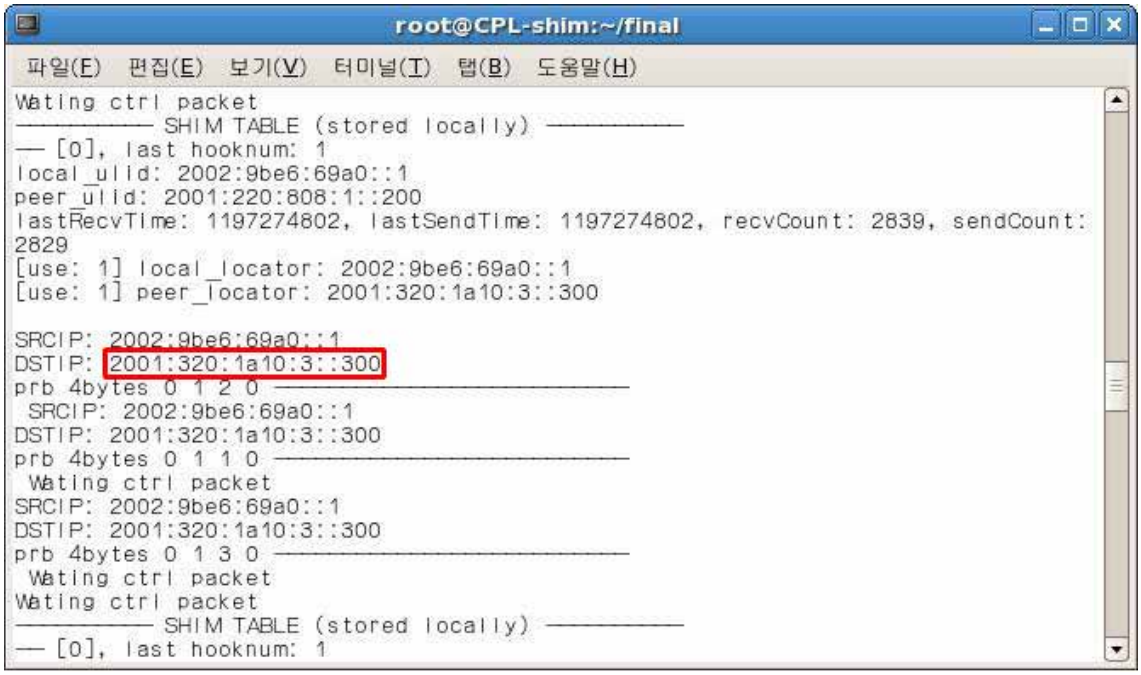

그림 10. shim 동작 전

i. server link down

server에서 link를 다운시킨다. link를 down하는 방법으로는 직접 lan선을 뽑는 방법이 있고, ip 명령어에서 link를 down하는 방법 등 다양하다. link를 down하면 client에서는 VLC 화면이 12~15초 정도 멈추었다 다시 나오는 것을 확인할 수 있다. 화면이 다시 나오면 wireshark를 저장 후 종료한다.

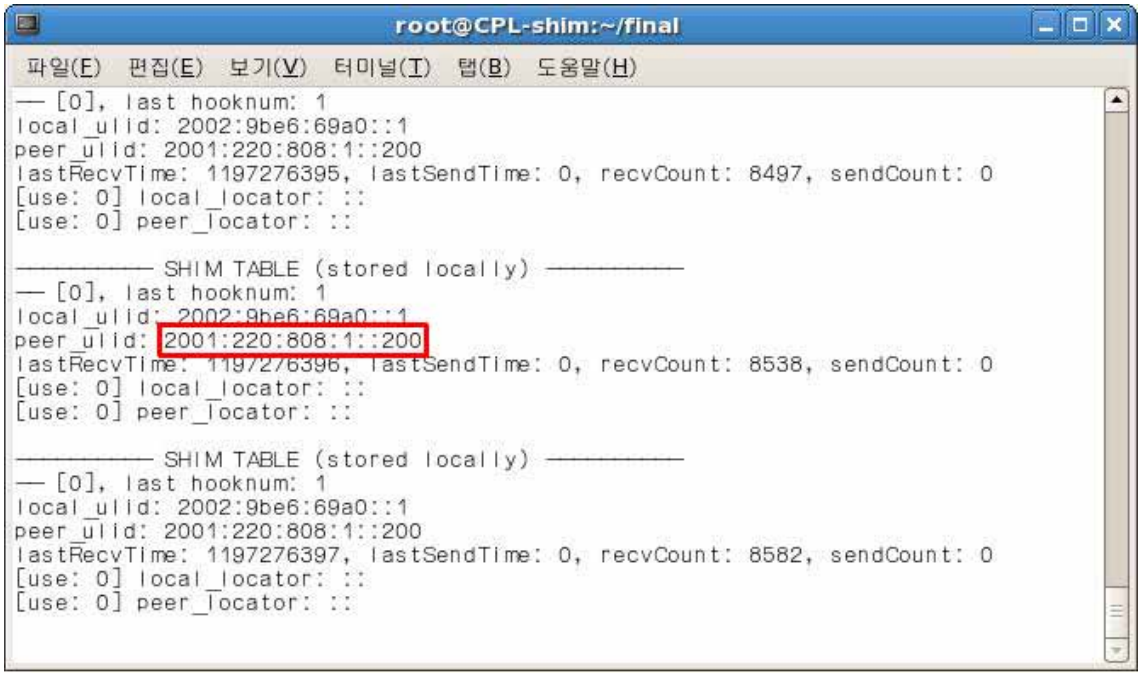

그림 11. shim 동작 후

그림 10과 그림 11을 비교해보면 shim table에서 2001:320:1a10:3::300 에서 2001:220:808:1::220으로 바뀐 것을 확인 할 수 있다.

ii. client link down

server의 link를 복구한 후 shim이 다시 연결된 것을 확인한다. 그림 12를 보면 다시 연결되어 있다는 것을 확인할 수 있다.

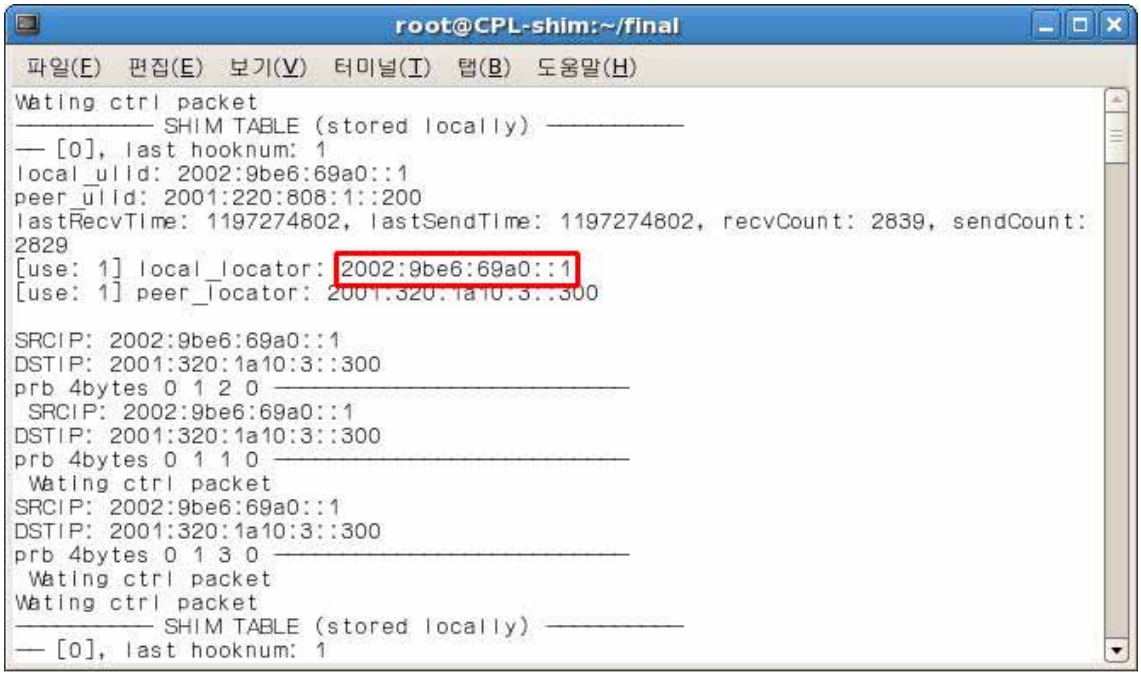

그림 12. shim 동작 전

이번엔 client의 link를 down한다.

$ip link set dev [device name] down

client에서는 VLC 화면이 12~15초 정도 멈추었다 다시 나오는 것을 확인할 수 있다. 화면이 다시 나오면 wireshark를 저장 후 종료한다.

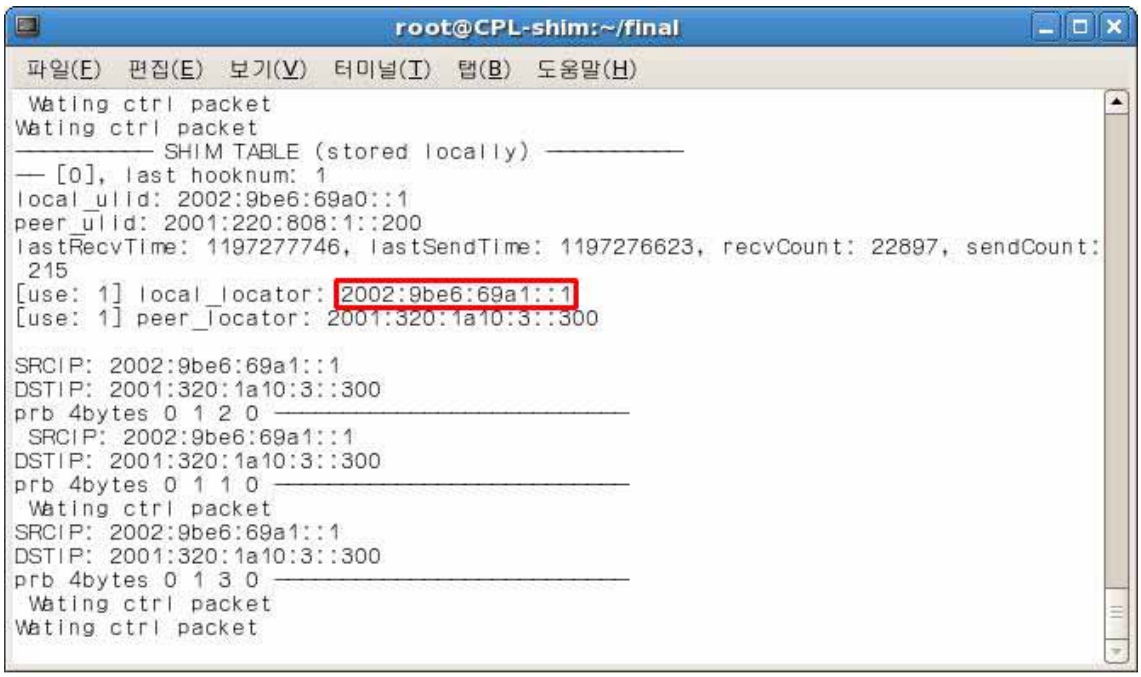

그림 13. shim 동작 후

그림 12와 그림 13을 비교해보면 local_locator부분이 바뀐 것을 알 수 있다. 이는 client link down 후 shim이 정상 작동하여 기존의 link가 아닌 다른 link에 의해 연결되고 있음을 뜻한다.

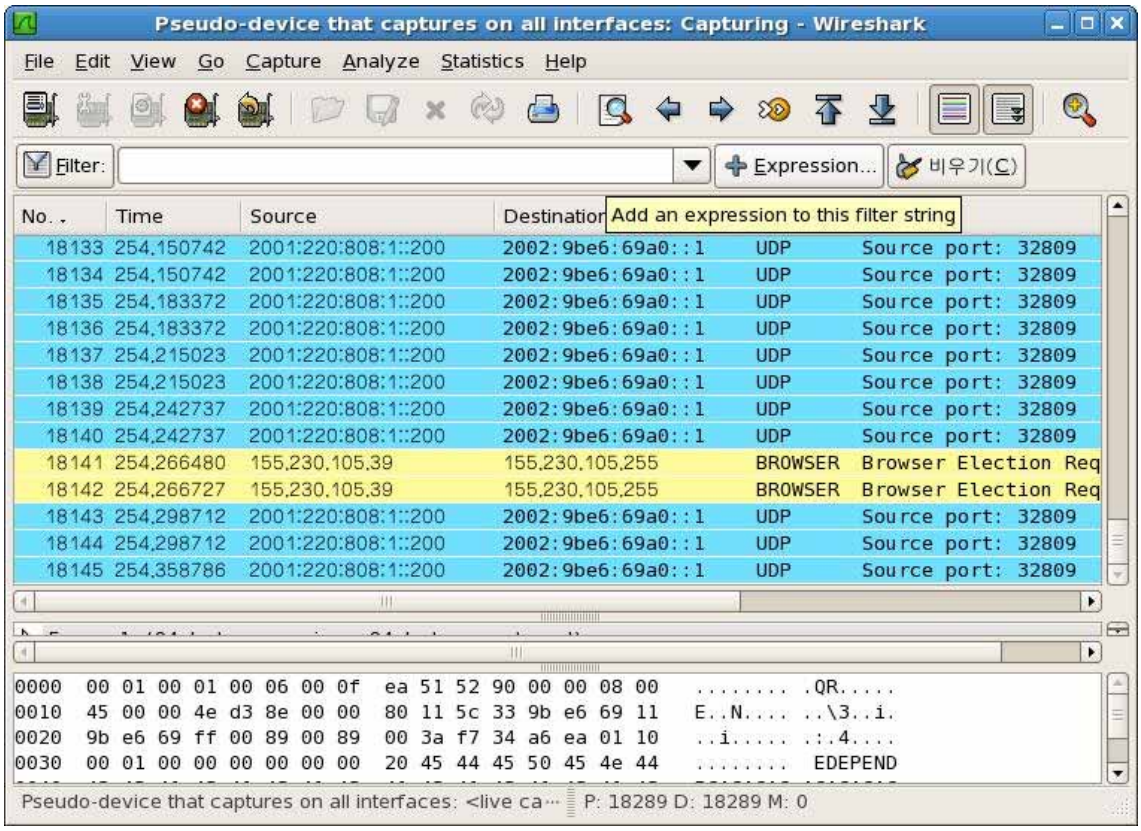

그림 14. wireshark 실행 화면

5. 결론

Test 결과 server에서 link를 down하여도 client에서 link를 down하여도 두 가지 경우 모두 정상적으로 shim이 작동하는 것을 알 수 있다. test 종료 후 wireshark를 보면 shim이 작동하고 나서 source와 destination의 IP 주소를 살펴보면 변화가 없다. shim을 실행한 터미널 화면에서는 분명 link down시 locator가 변하였는데 wireshark는 그렇지 않다. 이는 locator는 변화하였으나, identifier는 변화하지 않았기 때문이다. 즉, wireshark는 identifier를 capture한 것이지 locator를 capture한 것이 아니다.

이번 VLC를 이용한 test는 UDP기반으로 이루어졌다. UDP의 경우 세션의 개념이 명확하지 않다. 따라서 VLC가 아닌 세션의 개념이 명확한 다른 TCP 기반의 응용 프로그램에서도 추후 실험이 이루어져야 할 것으로 생각된다.

이번 test를 끝으로 더 이상의 shim에 대한 개인적인 연구는 없을 것으로 생각된다. shim은 현재까지의 인터넷 구조와는 다른 새로운 인터넷 구조를 도입하는 과정에서 생긴 identifier와 locator 분리 기술을 실제로 보여주는 역할을 담당하였고, 앞으로 멀티호밍 환경뿐만 아니라 보안성, 이동성을 고려한 새로운 인터넷 구조에 대한 다른 많은 연구가 있으리라 생각된다. 이와 같이 자연스럽게 인터넷 아키텍처의 변화가 진행이 되고, 결국 새로운 아키텍처의 적용을 고려할 시점이 다가올 것이며, 현재의 아키텍처가 아닌 새로운 아키텍처로 완전히 바뀌는 결과를 가져올 것이다.

참고 문헌

[1] E. Nordmark, “Shim6: Level 3 Multihoming Shim Protocol for IPv6”, IETF Internet Draft, draft-ietf-shim6-proto-08.txt, May 2007

[2] 유태완, “인터넷 주소의 Identifier/Locator 분리에 관한 기술 및 표준화 동향", TTA IT Standard Weekly, June 2005

[3] 서울대학교 멀티미디어 및 이동통신 연구실, 한국 전자통신 연구원 서비스 융합 표준 연구팀, shim(release 2) 실행 방법.doc, October 2007

[4] Fumio Teraoka, “LIN6: A Solution to Multihoming and Mobility in IPv6”, IETF Internet Draft, draft-teraoka-multi6-lin6-00.txt, December 2003

[5] Petri Jokela, “Host Identity Protocol – Extended Abstract”, Ericsson Research, 2004

[6] 신명기, “미래인터넷 기술 및 표준화 동향”, ETRI 전자통신동향분석 제22권 제6호, December 2007

[7] 손준영, “무선망 구현을 위한 적정 프로토콜분석에 관한 연구”, 동의대학교 대학원, February 1998

Appendix A. Shim6 설치 방법

※ 본 문서는 test를 위해 shim6를 설치하며 만든 문서이므로 device 이름(tun6to40, tun6to1)과 IP 주소는 설치할 때 변경해야 하는 부분입니다. []부분은 설치 시 반드시 자신에게 맞는 값을 넣어야 한다.

1. Kernel 컴파일

- Fedora Core 6의 경우 처음 kernel 버전이 2.6.18이다. 여기서는 netfilter와 Iptable을 재 설치 하기 위해 새로운 버전의 kernel이 필요하다. 버전은 2.6.18 이상. 2.6.22, 2.6.23.1 버전의 경우 컴파일 후 kernel panic 현상이 일어났다.

```
$cd /usr/src/
$wget http://155.230.105.153/linux-2.6.19.7.tar.gz
```

- kernel source download(여기서는 개인 웹서버 사용)

```
$tar xvfz linux-2.6.19.7.tar.gz
$cd linux-2.6.19.7
$make mrproper
$make menuconfig
<- terminal창의 크기를 일정 크기 이상하지 않을 경우 error 발생.
<- File system에서 ext3 지원 * check
$make bzImage
$make
$make modules_install
$make install
$reboot
```

- 재부팅시 새로 컴파일한 이미지로 부팅. 여기서는 2.6.19.7버전이다.

2. Iptables 설치

```
$wget http://www.netfilter.org/projects/iptables/files/iptables-1.3.5.tar.bz2
$tar xjvf iptables-1.3.5.tar.bz2
$cd iptables-1.3.5
$make
$make install
$make install-devel
$cd iptables-1.3.5
$vim Makefile
```

- Makefile 수정

```
Ifndef KERNEL_DIR
KERNEL_DIR = /usr/src/linux-2.6.19.7
endIf
```

3. Shimv2 설치

$wget http://155.230.105.153/shim2007_v2.tar
$tar xvf shim2007_v2.tar
$cd final
$cp ip6t_L3SHIM_* /usr/src/linux-2.6.19.7/net/ipv6/netfilter/
$cd /usr/src/linux-2.6.19.7/net/ipv6/netfilter/
$vim Kconfig

- Kconfig의 마지막 줄에 위치한 endmenu 윗부분에 아래 굵은 글씨 추가

config IP6_NF_TARGET_L3SHIM_IN
tristate 'L3SHIM_IN target support'
depends on IP6_NF_MANGLE
help
This option adds an L3SHIM INPUT target.

config IP6_NF_TARGET_L3SHIM_OUT
tristate 'L3SHIM OUT target support'
depends on IP6_NF_MANGLE
help
This option adds an L3SHIM OUTPUT target.
endmenu

$vim Makefile

- Makefile 마지막 부분에 추가

 # l3shim
obj-$(CONFIG_IP6_NF_TARGET_L3SHIM_IN) += ip6t_L3SHIM_IN.o
obj-$(CONFIG_IP6_NF_TARGET_L3SHIM_OUT) += ip6t_L3SHIM_OUT.o

$cd /root/final
$cp libip6t_L3SHIM_* /root/iptables-1.3.5/extensions/
$cp .L3SHIM_* /root/iptables-1.3.5/extensions/
$cd /usr/src/linux-2.6.19.7
$make && make modules_install && make install

- 선택 문항이 나오면 m선택(2번)

$cd /root/iptables-1.3.5
$make && make install
$vim Makefile

- PREFIX 경로 수정

#PREFIX:=/usr/local
PREFIX:=

$make && make install

4. IPv6 tunneling

- Create interface

$/sbin/ip -6 tunnel add tun6to40 mode sit ttl 64 remote any local [IP address1]

$/sbin/ip -6 tunnel add tun6to41 mode sit ttl 64 remote any local [IP address2]

- Activate interface

$/sbin/ip link set dev tun6to40 up

$/sbin/ip link set dev tun6to41 up

- Add address

$/sbin/ip -6 addr add [IP address1을 통해 계산한 IPv6 address1] dev tun6to40

$/sbin/ip -6 addr add [IP address2을 통해 계산한 IPv6 address2] dev tun6to41

- Set route

$/sbin/ip -6 route add 2000::/3 via ::203.254.38.130 dev tun6to40 metric 1

$/sbin/ip -6 route add 2000::/3 via ::203.254.38.130 dev tun6to41 metric 1