논문 2011-48TC-7-3

호스트 ID 기반 통신을 위한 기본 MOFI 테스트베드 구축

(Basic MOFI Testbed Implementation for Host ID-based Communication)

정 회 진*, 민 석 홍*, 이 재 용**, 김 병 철**

(Whoi Jin Jung, Seok Hong Min, Jae Yong Lee, and Byung Chul Kim)

요 약

최근 들어 미래인터넷에 대한 관심과 연구가 활발히 진행 중에 있다. 국내에서도 미래인터넷 연구의 한 분야로서 이동성 및 무선 환경 위주의 네트워크 환경으로의 변화에 초점을 맞춘 미래인터넷 아키텍처로 MOFI 가 제안되었다. MOFI 는 Mobile Oriented Future Internet 의 약자로 name 기반의 통신을 제공하며 라우팅 scalability 가 보장되고 이동성이 제공되는 데이터 전달 구조이다. 본 논문에서는 MOFI 구조 연구의 일부분으로 HID 기반 통신을 지원하는 기본 MOFI 테스트베드를 구축하고, 기본적인 인터넷 서비스 (PING, WWW)의 실험을 통해 HID 기반의 통신 가능성을 검증하였다. 테스트베드를 구축함에 있어 VirtualBox 라는 가상 머신을 사용하였고, 또한 Click Modular Router를 사용하여 패킷 처리 및 HCP 헤더의 추가 및 변환을 구현하였다.

Abstract

In recent years, the interest and research for Future Internet are rapidly increasing. In domestic, MOFI (Mobile Oriented Future Internet) is proposed as one architecture of Future Internet. MOFI is a data transmission architecture which provides a mobility, name-based communication and routing scalability. In this paper we implement a basic MOFI testbed that supports HID-based communication, and verify the feasibility of HID-based communication through experimentation of general service such as PING and WWW service. We used “VirtualBox” as a virtual machine and implement a packet processing and a HCP header addition and translation function using “Click Modular Router”.

Keywords: Future Internet, MOFI, Click Modular Router

Ⅰ. 서 론

현재 인터넷의 한계를 극복하고 미래 사회를 위한 인프라로서의 인터넷을 개발하기 위한 미래인터넷 연구가 최근 유럽, 미국, 일본 등을 중심으로 활발히 이루어지고 있으며, 국내에서도 이미 미래인터넷 포럼[1] 및 ETRI 와 같은 주요 연구기관에서 연구를 진행하고 있다.

미래인터넷에서는 유선 단말을 기반으로 하는 기존 인터넷보다는 무선 이동 단말 (스마트폰, PDA, 센서노드 등) 위주의 통신이 주를 차지할 것으로 보인다. 이러한 미래인터넷에서는 무선 단말의 이동성을 효율적으로 지원되는 시스템을 가져야 하며, 호스트 네임 기반의 통신을 지원할 수 있도록 하여 사용자 편의성을 향상시키는 것이 필요하다. Locator / Identifier 의 분리 방안을 통한 라우팅 확장성 확보 및 트래픽 엔지니어링, 멀티호밍 등의 문제 해결도 매우 중요한데 이를 위해 IETF 및 여러 연구 기관을 중심으로 많은 연구가 이루어지고 있다.

국내에서도 이러한 연구의 일환으로 이동성 위주의

* 학생회원, ** 평생회원, 충남대학교 정보통신공학과
(Chungnam National University)

※ 본 연구는 방송통신위원회의 산업원천기술개발사업의 연구결과로 수행되었음(KCA-2011-10913-05004: 미래인터넷에서의 이동환경 및 네트워크 다양성 지원구조 연구)

접수일자: 2011년5월3일, 수정완료일: 2011년7월13일

망 구조 연구가 진행되고 있으며, ETRI를 중심으로 MOFI 라는 구조가 제안되었다. MOFI 는 Mobile Oriented Future Internet 의 약자로서 ETRI 주관 하에 진행하고 있는 미래인터넷 연구 중 “Mobility Control” 분야의 연구이다.

본 논문에서는 MOFI 구조에서 가장 중요한 HID 기반 통신 가능성 검증을 위한 테스트베드를 구축하고, 이를 통해 호스트 ID (HID) 기반의 통신이 이루어짐을 검증하였다. 즉, HID 기반으로 통신이 이루어지기 때문에 종단 단말이 사설망 같은 동일한 주소 체계를 가지더라도, 해당 액세스 망에서만 유일성이 보장되고 HID 가 다르면 통신이 이루어짐을 검증하였다.

이를 위해 HID 기반의 통신이 가능한 기본 MOFI 테스트베드를 구축하였다. 구축된 MOFI 테스트베드를 활용하여 대표적인 인터넷 서비스인 PING, WWW 서비스가 HID 기반의 통신으로 정상 동작하는 것을 검증하였다. 호스트 단에서 HID 기반의 통신을 지원하기 위하여 VirtualBox [9] 라는 가상머신을 사용하여 MOFI_Agent 를 설계 및 구현하고, HID 패킷 처리를 위한 기능을 호스트와 AR 에 Click Modular Router [2](이하 Click 으로 기술)를 활용하여 구현하였다.

다음 II 장에서는 미래인터넷 기술로서 Locator / Identifier 분리 방안 및 이동성 지원 연구에 대한 관련 연구를 소개하고, III 장에서는 MOFI 아키텍처에 대해 기술한다. 이어서 IV 장에서는 Click 에 대한 소개와 기본 MOFI 테스트베드 구축에 대한 내용을 기술한다. 그리고 V 장에서 실험을 통한 결과를 분석하고 마지막으로 VI 장에서 결론 및 향후 연구 계획으로 논문을 마무리한다.

II. 관련 연구

현 인터넷의 문제점을 해결하기 위한 미래인터넷 연구에서는 단순히 DFZ (Default Free Zone) 에서의 라우팅 테이블의 감소를 위한 목적뿐만 아니라, 이동성, 멀티호밍, 트래픽 엔지니어링 등의 서비스를 지원하기 위해 새로운 라우팅 및 어드레싱 아키텍처를 연구하고 있다. 현재의 IP 주소 체계는 ID (Identifier) 와 LOC (Locator) 이 분리되어 있지 않음으로 이동성 및 멀티호밍 등을 지원하기 위해 MIP (Mobile IP) 와 같은 추가적인 프로토콜을 사용한다. 하지만 미래인터넷을 위한 다양한 장점을 가지는 새로운 아키텍처를 설계할 때 ID / LOC 분리 구조로 새로운 라우팅 / 어드레싱 아키텍처를 IRTF (Internet Research Task Force) 의 RRG (Routing Research Group) 에서 연구하고 있으며, 다수의 ID / LOC 분리 방안이 새롭게 제안되고 있다. 현재 제안된 ID / LOC 분리 방안들은 대표적으로 LISP (Locator Identifier Separation)[3], HIP (Host Identity Protocol)[4], GLI-Split (Global locator, Local locator, Identifier)[5] 등이 있다.

1. LISP [3]

LISP 은 map-and-encap 기법으로 패킷이 백본 망에 라우팅될 때 추가적인 IP 헤더를 가지고 인캡슐레이션하여 터널링을 통하여 전달하는 기법이다. LISP 에서는 두 개의 주소 공간을 가지게 되는데, 하나는 액세스 망에서 사용하는 End-point ID (EID) 이고, 또 하나는 Routing Locator (RLOC) 란 백본 망에서의 라우팅을 위해 사용하는 주소 공간이다. 그림 1 과 같이 LISP 은 Application 계층에서 하위 Network 계층까지 기존 IP 주소를 ID 로 사용하고, LOC 은 추가적인 Network 계층을 그 하위 계층에 추가해 edge 라우터의 IP 주소를 사용하게 된다. 기본적으로 백본 망과 액세스 망이 연결되는 edge 라우터인 ITR (Ingress Tunnel Router), ETR (Egress Tunnel Router) 에서 추가적인 터널링 기능을 처리한다. 이런 LISP 은 호스트 기반의 방안인 HIP 와는 달리 종단 호스트들의 변경은 필요하지 않고, edge 라우터인 ITR, ETR 에서 목적지 호스트의 EID 와 RLOC 을 매핑시켜주는 기능, 즉 EID 를 사용하는 inner 헤더를 RLOC 을 사용하는 outer 헤더로 인캡슐레이션하는 기능이 필요하게 된다. LISP 에서는 이와 같은 터널링 기능 이외에 EID 와 RLOC 을 매핑시켜주

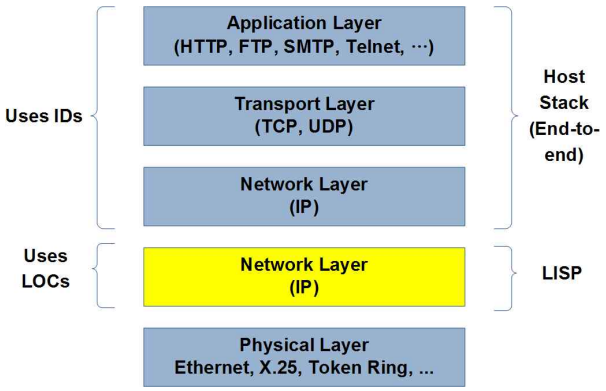


그림 1. LISP 프로토콜 스택
Fig. 1. LISP Protocol Stack.

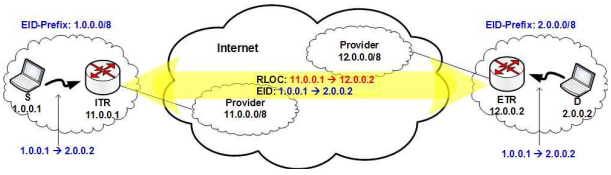


그림 2. LISP 기본 동작
Fig. 2. LISP Basic Operation.

는 별도의 시스템이 필요한데, 현재 LISP-ALT [8]등의 다양한 매핑시스템이 제안되어 있다.

그림 2는 LISP의 기본 동작을 나타낸다. 이 예제는 송신지인 S에서 수신지인 D 에게 데이터를 보내려고 하는 상황이다. S의 주소는 1.0.0.1이고, D의 주소는 2.0.0.2 이다. 이때 S는 ITR 을 거치면서 송수신지 주소가 [11.0.0.1, 12.0.0.2]로 변경된다. 다음 ETR로 데이터가 전송되고 ETR 은 다시 송수신지 주소를 [1.0.0.1, 2.0.0.2] 로 변경한 후, 종단 수신지인 D 로 데이터를 전송하게 된다.

2. HIP [4]

HIP 는 호스트 기반의 방안으로 ID / LOC 의 분리뿐만 아니라 암호화된 공개키를 사용하는 Host ID (HI)를 해쉬값으로 나타내는 128 비트의 Host ID Tags (HIT) 를 사용하는 프로토콜이다. 기존의 TCP / IP 스택에서 Transport 계층과 IP 계층 사이에 HIP 계층을 새롭게 추가하여, 기존 IP 계층의 IP 주소는 LOC 정보로 사용하고 HI 는 종단의 ID 로 사용하게 된다. 그림 3 은 기존의 소켓과 변화된 소켓을 보여준다. 소켓이 기존과 같이 IP 주소가 아닌 HI에 바인딩 되며 이 HI 값은 커널에서 IP 주소로 바뀌게 된다.

HIP의 기본 동작은 그림 4와 같다. Initiator가 responder 에게 트리거 패킷을 보낸다. Responder는 암호화된 메시지인 퍼즐과 Diffe-Hellman key, signature를 함께 보낸다. Responder 로부터 패킷을 받은

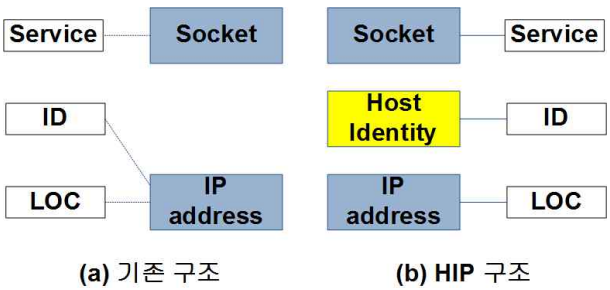


그림 3. HIP 소켓
Fig. 3. HIP Socket.

그림 4. HIP Association 의 기본 과정
Fig. 4. Basic Procedure of the HIP Association.

initiator는 공개키를 사용하여 퍼즐의 답을 계산하게 되고 이를 responder에게 보낸다. 이를 받은 responder는 association의 성립을 의미하는 signature를 initiator에게 보냄으로써 initiator와 responder 간의 HIP association이 성립되게 된다.

HIP 는 IP 계층과 Transport 계층 사이에 새로운 계층을 추가하여 ID / LOC 을 분리함으로써 보안성, 이동성, 멀티호밍 등의 장점을 가진다. 하지만 HIP를 지원하지 않는 기존의 단말과 호환되지 않으며, DNS 의 확장과 새로운 기능을 위한 Rendezvous Server (RVS) 등이 새롭게 추가되어야 하는 단점이 존재한다.

3. GLI-Split [5]

GLI-Split 방안은 map-and-rewritable 기법을 통하여 ID / LOC 을 분리하고, 백본 망과 액세스 망을 분리한다. 이 방안의 주소 체계는 global locator (GL), local locator (LL), identifier (ID) 로 구분하여 현재의 IPv6 주소에 인코딩함으로써 새로운 라우팅 프로토콜이 필요하지 않고 또한 기존의 legacy IPv6 망과의 호환성을 제공하게 된다. 그림 5 는 GLI 주소 체계로서 Identifier address, Local Address, Global Address 의 세 가지 주소 체계를 가진다. ID 는 종단 호스트의 식별자로 사용

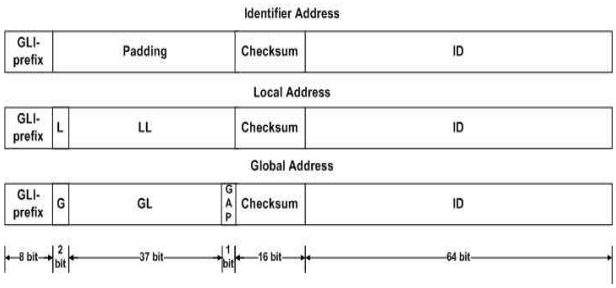


그림 5. GLI 주소 체계
Fig. 5. GLI Address Structure.

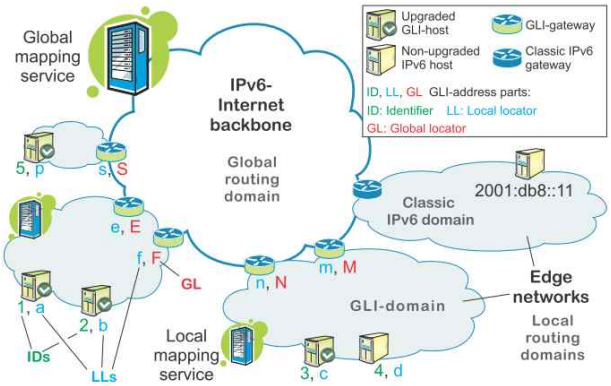


그림 6. GLI-Split 참조 망
Fig. 6. GLI-Split Reference Network.

되고 LL, GL 은 각각 액세스 망, 백본 망에서의 라우팅을 위한 LOC 로 사용된다.

그림 6 과 같은 망구조에서 정수로 표시한 1, 2, … 는 호스트의 ID 를 나타내고, 영문소문자는 LL, 영문대문자는 GL 을 나타낸다. ([ID, LL, GL] 로 표기) 예를 들어 [1, a] 를 가진 호스트가 동일한 도메인 내에 위치한 [2, b] 호스트에게 데이터를 전송하는 경우는 백본 망을 거치지 않으므로 GL 이 필요 없이, 송신 호스트가 Local Mapping Service 에서 수신 호스트의 LL 을 받아오게 되고 해당 LL 을 이용하여 로컬 라우팅을 통하여 전송하게 된다. 만약 [1, a] 호스트가 다른 도메인에 위치한 [3, c] 호스트로 데이터를 전송하는 경우는 Local Mapping Service 가 수신 호스트의 위치를 Global Mapping Service 에게 받아오게 되고 수신 호스트가 위치한 edge 라우터의 GL 을 송신 호스트에서 보내는 데이터의 수신지 GL 로 사용하여 전송하게 된다. GLI-Split 은 또한 기존 IPv6 망과의 호환성 및 주소 symmetry 문제, 멀티 호밍 등을 제공하기 위하여 GAP, Checksum 필드 등을 사용한다.

Ⅲ. MOFI 시스템

본 논문에서의 MOFI 아키텍처 및 프로토콜은 “MOFI Architecture and Protocols release 2.2”[6]를 기준으로 기술한다.

MOFI 는 미래인터넷을 최종적인 컨버전스 네트워크로 간주하고 미래인터넷에서 액세스의 주가 될 이동환경을 효율적으로 지원할 수 있는 데이터 전달 구조 및 절차를 제안하는데, 주요한 설계 원칙은 아래와 같다[7].

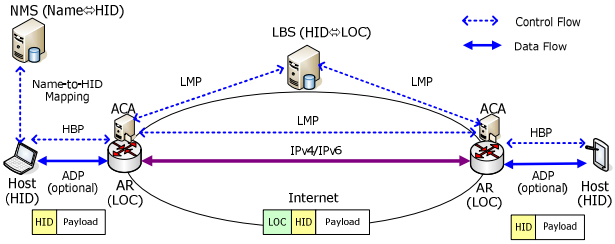


그림 7. MOFI 참조 모델
Fig. 7. MOFI reference model.

- 이동단말이 주가 되는 이동성 중심의 네트워크
- 위치 기반 라우팅을 통한 ID 기반의 통신
  - Identifier 와 locator 의 분리
  - 주소가 고정되지 않는 호스트
  - ID 기반의 종단간 범용 통신과 LOC 기반의 지역 라우팅
- 액세스 망과 백본망의 분리를 위한 프로토콜 분리
- Mobility control plane 과 data delivery plane 의 분리를 위한 기능 분리

MOFI 아키텍처는 그림 7 과 같이 기본적으로 호스트와 Access Router (AR), Name-HID Mapping System (NMS), Locator Binding System (LBS) 등으로 구성된다. 사용자는 하나 이상의 단말을 통해 미래인터넷에 접속할 수 있으나 복잡성을 피하기 위해 MOFI 에서는 단일 사용자가 하나의 단말을 사용하는 경우를 가정한다.

AR 은 백본 망과 액세스 망을 연결해 주는 역할을 수행한다. MOFI 에서 AR 은 기존의 게이트웨이 기능에 header translation 이라는 기능이 추가적으로 구현된다. 또한 AR 은 백본 망에서 locator 에 기반하여 현 인터넷과 마찬가지로 데이터를 라우팅하여 목적지까지 전달해 주는 기능을 수행한다. MOFI 에서의 식별자는 기본적으로 name 과 HID, locator 로 구분된다. Name 은 사용자 단말, 서비스, 컨텐츠 등을 구분하기 위한 사용자가 인식할 수 있는 이름 형태의 식별자이고, HID 는 해당 단말이 네트워크를 사용하기 위한 고정길이 식별자를 의미한다. 사용자는 name 에 기반하여 통신을 시작하고 이때 name 과 HID 의 매핑 기능을 담당하는 것이 NMS 이다. 즉 NMS 는 현 인터넷에서의 DNS 와 비슷한 기능을 담당한다. LBS 는 LOC (locator) 정보와 HID를 매핑해 주는 기능을 담당하지만 본 논문에서는 백본 망을 IP 망으로 가정하고, IP를 LOC 로 그대로 사

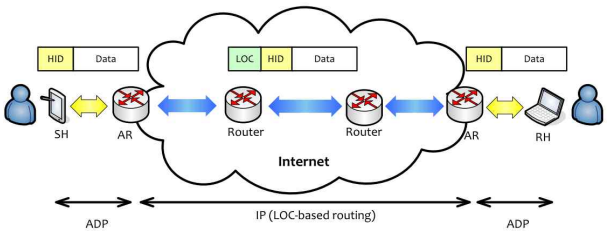


그림 8. MOFI 에서의 데이터 전달
Fig. 8. Data Transmission in MOFI.

용한다.

그림 8 은 MOFI 구조에서의 데이터 전달이다. 송신 호스트 SH 는 데이터를 전송하기 위해 송수신 객체의 name 을 NMS 에 문의해 HID를 받아 오고, ADP (Access Delivery Protocol) 즉, 송신 액세스 망에서의 라우팅 프로토콜을 통해 AR 로 전송하면 AR 은 LBS 를 통하여 수신 HID 에 해당되는 LOC 정보를 받아온다. 받아온 LOC 정보를 통하여 백본 망에서의 라우팅을 수행하고 데이터를 수신 호스트 RH 가 속해 있는 액세스 망의 AR 까지 전달하게 된다. RH 가 속해 있는 AR 은 수신 액세스 망의 ADP를 통해 RH 로 데이터를 최종적으로 전송한다. 이때 호스트와 AR 간의 프로토콜인 ADP 는 액세스 망에 따라 선택적으로 적용된다.

본 논문에서는 MOFI 시스템의 data plane 만 기술하고 control plane 의 설명은 MOFI 문서 [6]를 참조하면 된다.

IV. 기본 MOFI 테스트베드 구축

기본 MOFI 테스트베드를 구축함에 있어 모든 기능을 구현하는데 제약이 있으므로 다음과 같이 가정한다.

- NMS를 통한 name-to-HID resolution 이후부터 HID 기반 end-to-end 통신을 한다.
- Access Delivery Protocol (ADP)은 임의의 routable multi-hop 프로토콜이 사용 가능한데, 본 테스트베드에서는 사설 IPv4 망을 사용을 가정한다.
- 종단 호스트는 HID 만을 사용하여 통신한다.
- Control plane 은 구현하지 않고 data plane 만 구현한다. 즉 HID-to-LOC resolution을 위해 각 호스트가 static mapping table을 가지고 있다.
- 종단 호스트의 이동은 고려하지 않고 HID 기반의 데이터 전달 가능성만을 검증한다.
- MOFI 용 Kernel 구현 대신, Click을 사용하여 구현한다.

1. Click Modular Router 소개

Click[2]은 MIT LCS's Parallel and Distributed Operating System 그룹과 Mazu Networks, ICSI Center, UCLA 가 공동으로 개발한 것으로, 유연한 조작이 가능한 패킷 처리기를 만들기 위한 새로운 소프트웨어 아키텍처이다. 기본적으로 Click 은 다양한 엘리먼트의 조합으로 하나의 라우터, 즉 패킷 처리기를 구성한다. Click 은 패킷 처리를 위한 다양한 엘리먼트를 기본적으로 제공하고 필요한 경우 기존의 엘리먼트를 수정하거나 새로운 엘리먼트를 추가할 수 있는 확장성도 제공한다. 또한 Click 은 사용자 입장에서 굉장히 유연하므로 엘리먼트들을 구성하는 문법만 틀리지 않다면 패킷 처리기의 구성 동작에 대해서는 관여하지 않고, 동작에 대한 모든 고려사항을 사용자에게 맡겨둔다. 이것은 사용자가 원하는 어떤 동작의 구성이라도 설계가 가능하다는 장점이 있다는 것이다.

Click 의 구성은 그림 9에서와 같이 다수의 엘리먼트(element)와 연결 (connection)으로 구성된다. 사용자는 원하는 기능이 있는 엘리먼트를 선택하여 사용하고, 패킷의 흐름에 따라 각 엘리먼트들을 연결하면 간단한 패킷 처리기를 설계할 수 있다. 그림 9에서 FromDevice 엘리먼트는 리눅스 머신의 네트워크 인터페이스 패킷을 가져오는 기능을 하고, Counter 엘리먼트는 패킷의 수를 카운팅하고, Discard 엘리먼트는 해당 패킷을 버리는 기능을 담당하는 엘리먼트이다. 즉, 네트워크 인터페이스에 들어오는 패킷의 수를 카운팅하고 해당 패킷을 버리는 기능을 하는 간단한 패킷 처리기의 설계 예이다. 그림 9와 같이 다수의 엘리먼트들을 적절하게 연결하여 packet classification, queuing, scheduling, interfacing 등의 다양한 패킷 처리 기능을 구현할 수 있다.

그림 10은 패킷 처리기를 구성하는 엘리먼트의 구성

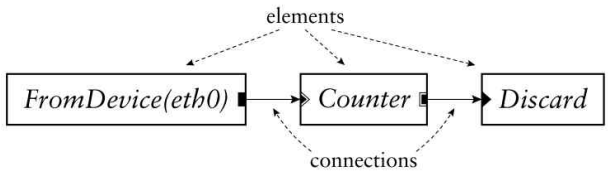


그림 9. 간단한 Click 패킷 처리기의 구성 예시
Fig. 9. Configuration example of simple packet processor using Click.

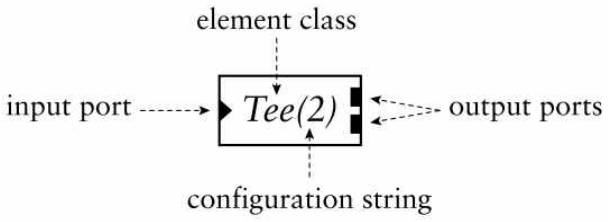


그림 10. 엘리먼트의 구성
Fig. 10. Structure of an Element.

을 보여준다. 엘리먼트는 element class 라는 엘리먼트의 이름과 input, output port, configuration string 으로 구성된다. Configuration string 은 해당 엘리먼트가 가질 수 있는 다양한 속성들을 나타낸다. 엘리먼트는 다수의 입출력 포트를 가질 수 있고, 기능에 따라 push, pull, agnostic port 로 구분한다. 모든 포트들은 적어도 하나 이상의 연결을 가져야 한다. Agnostic port 는 상황에 따라 입력 또는 출력 포트로 동작할 수 있다. 한 엘리먼트의 출력 포트는 반드시 다른 엘리먼트의 입력 포트와 연결을 맺어야 한다. 삼각형 모양의 포트는 입력포트, 사각형 모양의 포트는 출력포트를 나타낸다. 또한 push 및 pull 포트로 연결을 맺을 때 다음과 같은 제약사항이 있다.

- Push 출력 포트는 반드시 push 입력 포트와 연결되어야 한다.
- Pull 출력 포트는 반드시 pull 입력 포트와 연결되어야 한다.
- 한 엘리먼트에서 agnostic 포트는 push 또는 pull 로만 구성되어야 한다.
- Push 출력 포트와 pull 입력 포트는 반드시 한번만 연결되어야 한다.

Click 은 http://read.cs.ucla.edu/click/ 에서 다운로드가 가능하고 현재까지 제공된 가장 최신 버전은 Ver. 1.8 이다. Click 은 리눅스 머신과 BSD 에 제공되는 오픈 소스로서 누구나 참여하여 새로운 엘리먼트나 프로젝트를 새롭게 구현하거나 제공할 수 있는 장점이 있다. 엘리먼트는 하나의 C++ 오브젝트이고, connection 은 엘리먼트 오브젝트의 포인터를 의미한다.

Connection을 통한 패킷 처리는 가상 함수를 호출하기 때문에 효율적인 패킷 처리기를 구성하기 위해선 엘리먼트와 connection을 효과적으로 설계하는 것이 요구된다. Click 의 실행 모드는 user-level 과 kernel- level 의 두 가지가 있다. Kernel-level Click 은 리눅스의 라우팅 모듈을 완전히 대체함으로써 처리 속도가 빠른 장점이 있는 반면, 실행 시 루트의 권한이 필요하고 잘못된 라우터의 설계를 실행 시 시스템이 해를 입을 수 있는 단점이 있다. User-level Click 은 하나의 리눅스 데몬으로 동작하게 되고, Kernel-level 보단 처리 속도가 느리지만, 설치가 쉽고 디버깅이 쉬운 장점이 있다. 하지만 요즘 PC 들은 성능이 예전보다 많이 좋아졌기 때문에 user-level 의 처리 속도도 충분히 보장 받을 수 있다.

2. 종단 호스트를 위한 MOFI_Agent 설계

종단 호스트들은 전송 계층과 네트워크 계층 사이에 그림 11처럼 HCP (HID based Communication Protocol) 계층이 추가되어야 한다. HCP 계층에서는 HID 기반의 통신을 위한 패킷 헤더 구성을 수행한다. HCP 헤더의 크기는 향후 IPv6 와의 호환성을 위해 IPv6 헤더 크기인 40 바이트를 사용한다. HCP 헤더에 송수신 호스트의 HID 가 들어가게 된다. 본 논문에서 ADP 는 사설망 IPv4 를 사용한다고 가정하였다.

MOFI_Agent 란 MOFI 종단 호스트에 필요한 HCP 헤더 추가 및 주소 변환 기능을 담당하는 에이전트로 하나의 리눅스 머신에 가상머신을 탑재하고 Click을 이용하여 헤더 추가 및 변환 기능을 수행하도록 구현하였다. 그림 12는 종단 호스트의 실제 인터페이스 구성을 보여준다. 여기서 tap0 인터페이스는 가상 호스트 (VM)의 가상 인터페이스명이고, eth0 는 실제 호스트 (Host)의 인터페이스명이다. VM에서 패킷을 생성하면 Host에서 HCP processing module을 통하여 HCP 헤더의

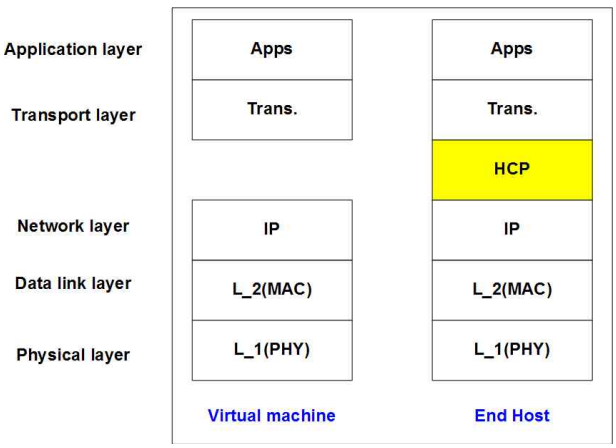


그림 11. 종단 호스트 프로토콜 스택
Fig. 11. End Host Protocol Stack.

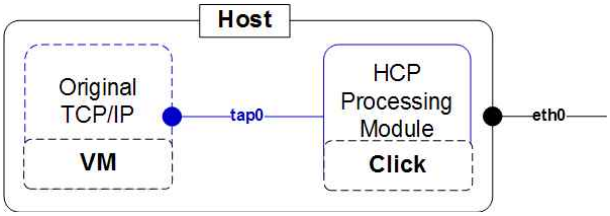


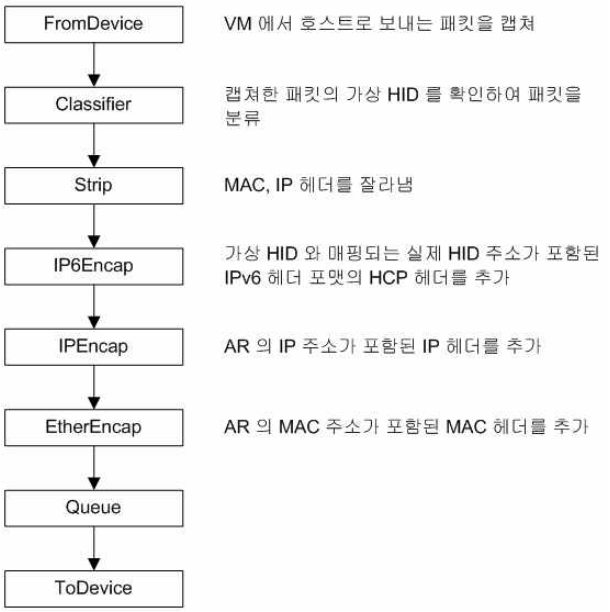
그림 12. MOFI_Agent 인터페이스
Fig. 12. MOFI_Agent Interface.

추가와 변환을 하게 된다. VM 에서는 HID를 사용하지 않고 임의의 IPv4 주소를 사용한다. VM에서 실제 호스트로 패킷이 전송되면 실제 호스트에서 호스트매핑테이블을 검색하여 VM 의 IPv4 주소에 해당되는 HID 주소를 매핑하고, 해당 HID 에 기반하여 HCP 헤더의 추가 및 변환 기능을 수행한다. 위와 같은 기능을 가진 MOFI_Agent를 구현하여 HID 기반의 통신 기능이 없는 호스트가 MOFI_Agent를 통하여 HID 기반의 통신이 가능하게 구현하였다. 즉 물리적으로 하나의 호스트 상에 HID 기반 통신이 가능하지 않은 VM 에 HID 기반 통신이 가능한 MOFI_Agent를 Click 을 사용하여 추가 구현한 것이다.

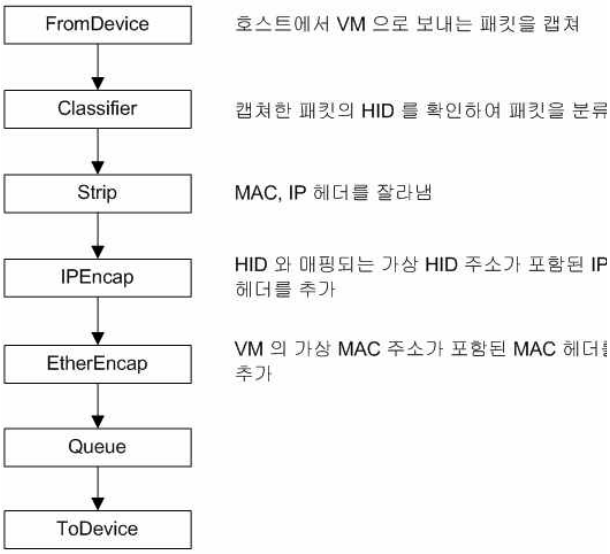
그림 13은 Click 을 이용하여 HCP processing module 을 구현한 것이다. 이 모듈은 VM에서 실제 호스트로 전송되는 패킷 처리와 실제 호스트에서 VM 으로 전송할 때의 패킷 처리로 구성된다.

VM에서 실제 호스트로 전송되는 패킷처리를 먼저 보면, FromDevice 엘리먼트는 VM에서 실제 호스트로 보내는 패킷을 캡쳐하고, Classifier 엘리먼트에서 캡쳐한 패킷의 가상 HID를 확인하여 패킷을 분류한다. 다음 Strip 엘리먼트로 MAC, IP 헤더를 잘라내고, IP6Encap 엘리먼트는 가상 HID 와 매핑되는 실제 HID 가 포함된 IPv6 헤더 포맷의 HCP 헤더를 추가한다. 다음 IPEncap 엘리먼트를 통하여 AR 의 IP 주소가 포함된 IP 헤더를 추가하고, EtherEncap 엘리먼트로 AR 의 MAC 주소가 포함된 MAC 헤더를 추가한 후, Queue 엘리먼트에 패킷을 저장하고 최종적으로 ToDevice 엘리먼트를 통하여 네트워크 인터페이스로 패킷을 포워딩한다.

다음 실제 호스트에서 VM 으로 전송되는 패킷 처리를 보면, FromDevice 엘리먼트는 호스트에서 VM 으로 전송되는 패킷을 캡쳐하고, Classifier 엘리먼트는 캡쳐한 패킷의 HID를 확인하여 패킷을 분류한다. 다음 Strip 엘리먼트로 MAC, IP 헤더를 잘라낸다. 이어



(a) VM에서 호스트로 패킷을 전송할 때 처리



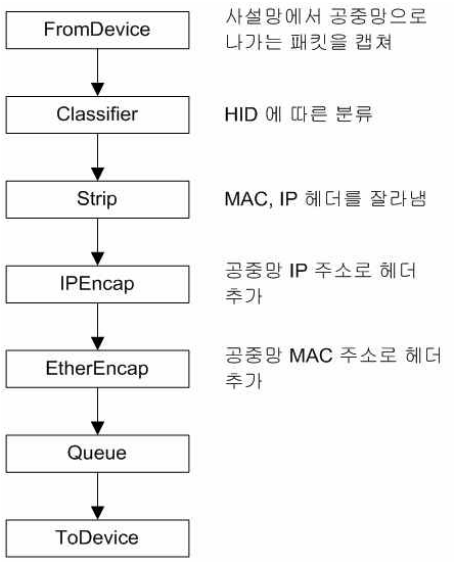
(b) 호스트에서 VM 으로 패킷을 전송할 때 처리

그림 13. MOFI_Agent Click 설계
Fig. 13. Click design for MOFI_Agent.

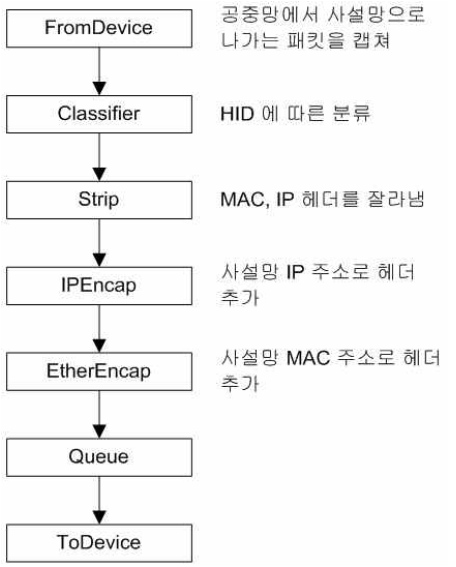
IPEncap 엘리먼트로 HID 와 매핑되는 가상 HID 주소가 포함된 IP 헤더를 추가하고, EtherEncap 엘리먼트로 VM 의 가상 MAC 주소가 포함된 MAC 헤더를 추가한 후, Queue 로 전송하고 최종적으로 ToDevice 엘리먼트를 통하여 가상 인터페이스인 tap0 인터페이스로 패킷을 포워딩한다.

3. AR을 위한 Click 설계

본 논문에서는 액세스 망을 사설 IPv4 망으로 가정



(a) 사설망에서 공중망으로 나가는 패킷 처리



(b) 공중망에서 사설망으로 나가는 패킷 처리

그림 14. Click AR 설계
Fig. 14. Click design for AR.

하였다. 그래서 AR 은 액세스 망인 IPv4 사설망에서 인터넷 망으로 나가는 패킷에 대한 HID-to-RLOC 변환 기능과 그 역기능인 RLOC-to-HID 변환 기능이 구현되어야 한다.

그림 14 에서 (a) 는 액세스 망에서 인터넷 백본 망으로 나가는 패킷에 대한 처리이고, (b) 는 반대로 들어오는 패킷에 대한 처리이다. 먼저 액세스 망에서 백본 망으로 나가는 패킷을 FromDevice 엘리먼트로 캡쳐하고, Classifier 엘리먼트로 HID 에 따라 분류한다. 다음 Strip 엘리먼트로 MAC, IP 헤더를 잘라내고, IPEncap, EtherEncap 엘리먼트로 공중망 IP 헤더와, MAC 헤더를 추가한다. 그리고 Queue 엘리먼트로 보낸 후 ToDevice 엘리먼트를 통해 네트워크 인터페이스로 패킷을 포워딩한다. 그림 14-(b)에서 패킷에 대한 처리는 그림 14-(a) 의 구현과 동일하고 다른 점은 IPEncap, EtherEncap에서 사설망 IP 헤더와 MAC 헤더를 추가하고 최종적으로 네트워크 인터페이스로 패킷을 포워딩하는 것이다.

4. 테스트베드 구축

테스트베드는 그림 15 와 같이 총 5 대의 리눅스 머신으로 구축하였다. 2 대의 머신은 AR1, AR2를 구현하였고, 3대의 머신은 HID1, HID2, HID3 호스트를 구현하였다. 액세스 망은 앞서 언급하였듯이 사설 IPv4 망을 사용하고, 백본 망은 KREONET 연구망을 사용하였다. 호스트의 VM에서 가상 HID 는 임의의 IPv4 주소를 사용하였고, 실제 HID 는 IPv6 주소와의 호환성을 고려하여 128 비트를 사용하였다. 종단 호스트의 각 VM 에서 사용한 가상 HID 와 실제 HID 의 매핑 정보는 표 1 에 명시되어 있다. 그림 15에서 HID2 호스트와 HID3 호스트는 서로 다른 사설망에 위치한 호스트로써 사설 IP 주소가 192.168.2.12 로 동일한 주소를 사용하도록 설정하였다. 현 인터넷에서는 NAT를 사용해 사설망에 위치한 호스트가 외부망 서버 등과의 통신이 가능하긴 하지만, 사설망 호스트로의 통신은 불가능하며 동일 사설 IP 주소를 가지는 노드가 있는 경우 통신이 안 된다. 그러나 본 테스트베드는 HID 기반으로 통신을 하기 때문에 서로 다른 사설망에 위치한 호스트가 동일한 사설 IP 주소를 사용하더라도 통신에 지장이 없는 것을 다음 장의 실험에서 확인할 수 있다.

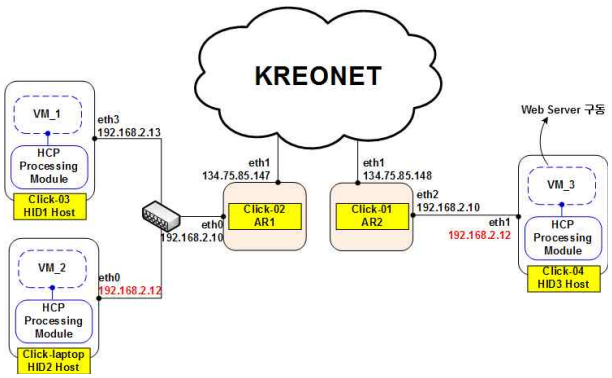


그림 15. 테스트베드 구성도
Fig. 15. Testbed topology.

표 1. 가상 HID 와 실제 HID 매핑 정보
Table 1. Mapping table between Virtual HID and real HID.

구분	가상 HID	실제 HID
AR1	없음	없음
AR2	없음	없음
HID1 호스트	11.11.11.11	111...111 (128 bits)
HID2 호스트	22.22.22.22	222...222 (128 bits)
HID3 호스트	33.33.33.33	333...333 (128 bits)

패킷 전송의 흐름을 간단하게 설명하면 HID1 호스트에서 HID3 호스트로 패킷을 전송하는 경우, HID1 호스트 VM_1 에서 생성된 패킷의 가상 HID를 확인한 후 실제 호스트에서 HID 매핑 테이블을 확인하여 HCP 헤더를 추가한다. 이어서 AR1 으로 패킷이 전송되고 AR1 은 HCP 헤더는 변경하지 않고 IP 헤더를 공중망 주소 (locator 정보)에 맞게 변환한다. 다음 AR2 는 해당 패킷을 다시 사설망 주소로 변환해 주고, 자신의 액세스 망에 속한 HID3 호스트에 패킷을 전송한다. HID3 호스트는 최종적으로 HID를 다시 가상의 HID 로 변경한 후 VM_3 에 패킷을 전송해 준다.

테스트베드를 구축함에 있어 상기 구현 외에 HID3 호스트에는 추후 HID 기반의 WWW 서비스 실험을 위한 웹 서버를 구동시켰다. 다음 장은 구축된 테스트베드를 활용하여 PING 과 WWW 서비스 실험한 결과와 그에 대한 분석을 기술한다.

Ⅴ. 실험 및 결과

1. PING 실험

PING 은 Packet Internet Groper 의 약자로서 일반적으로 네트워크에 있는 장비가 작동 중인지 아닌지, 또는 패킷이 해당 장비에 도달하는 데 얼마나 걸리는 지 등을 알아보는 데 많이 사용된다. HID 기반의 통신을 검증하기 위하여 본 논문에서도 기본적인 PING 애플리케이션을 활용하였다.

그림 16 은 HID2 호스트에서 HID1, HID2, HID3 호스트로 ping을 실험해 본 결과이다. HID2에서 HID1 으로의 ping 은 local routing을 통하여 즉, 하나 이상의 AR을 거치지 않고 패킷이 전달되고, HID2에서 HID3로의 ping 은 AR1, AR2를 거쳐 패킷이 전달되므로 앞의 local routing 보다는 RTT 값이 조금 더 큰 것을 확인할 수 있다. 그림 16의 (a), (b), (c)에 나타나듯이 HID2에서

```
PING HID1 (11.11.11.11) 56(84) bytes of data.
64 bytes from HID1 (11.11.11.11): icmp_seq=1 ttl=250 time=1.78 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=2 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=3 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=4 ttl=250 time=1.76 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=5 ttl=250 time=0.500 ms
... 중략 ...
64 bytes from HID1 (11.11.11.11): icmp_seq=96 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=97 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=98 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=99 ttl=250 time=0.000 ms
64 bytes from HID1 (11.11.11.11): icmp_seq=100 ttl=250 time=0.000 ms

--- HID1 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99815ms
rtt min/avg/max/mdev = 0.000/0.543/6.742/1.460 ms
```

(a) HID2 호스트 -> HID1 호스트로의 ping 실험 결과

```
PING HID2 (22.22.22.22) 56(84) bytes of data.
64 bytes from HID2 (22.22.22.22): icmp_seq=1 ttl=64 time=0.133 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=2 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=3 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=4 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=5 ttl=64 time=0.615 ms
... 중략 ...
64 bytes from HID2 (22.22.22.22): icmp_seq=96 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=97 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=98 ttl=64 time=1.93 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=99 ttl=64 time=0.000 ms
64 bytes from HID2 (22.22.22.22): icmp_seq=100 ttl=64 time=1.01 ms

--- HID2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99829ms
rtt min/avg/max/mdev = 0.000/0.265/2.293/0.527 ms
```

(b) HID2 호스트 -> HID2 호스트로의 ping 실험 결과

```
PING HID3 (33.33.33.33) 56(84) bytes of data.
64 bytes from HID3 (33.33.33.33): icmp_seq=1 ttl=250 time=1.46 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=2 ttl=250 time=0.000 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=3 ttl=250 time=0.503 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=4 ttl=250 time=0.000 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=5 ttl=250 time=0.000 ms
... 중략 ...
64 bytes from HID3 (33.33.33.33): icmp_seq=96 ttl=250 time=1.65 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=97 ttl=250 time=8.52 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=98 ttl=250 time=7.02 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=99 ttl=250 time=1.27 ms
64 bytes from HID3 (33.33.33.33): icmp_seq=100 ttl=250 time=11.4 ms

--- HID3 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99831ms
rtt min/avg/max/mdev = 0.000/1.063/11.466/2.222 ms
```

(c) HID2 호스트 -> HID3 호스트로의 ping 실험 결과

그림 16. PING 실험 결과
Fig. 16. PING test result.

HID1 으로의 평균 RTT 는 약 0.543 ms, HID2에서 HID2 로의 평균 RTT 는 약 0.265 ms, HID2에서 HID3

로의 평균 RTT 는 1.063 ms 이다. 그림 16 의 로그 메시지에서 볼 수 있듯이 HID 기반의 통신이 정상 동작하는 것을 알 수 있다.

또한 HID2 호스트에서 HID3 호스트로 ping 실험할 때 캡처한 패킷을 분석한 결과는 그림 17 과 같다. 실험에서 사용한 HID2, HID3 호스트의 사설망 주소는 192.168.2.12 로 동일하게 설정하였다. 그리고 AR1, AR2 의 사설망 주소는 192.168.2.10 으로 동일하고, 공중망의 주소는 134.75.85.147, 134.75.85.148 로 설정하였다. 그림 17 은 액세스 망에서의 패킷과 백본 망에서의 패킷을 구분해서 분석한 것이다. 즉 송신지 액세스 망의 HID2 호스트에서 AR1 으로의 전송되는 패킷과, 백본 망에서의 AR1에서 AR2로 전송되는 패킷, 그리고 수신지 액세스 망의 AR2에서 HID3 호스트로 전송되는 패킷을 분석하였다. 기존 인터넷에서 ping을 실험하면 IP 헤더 위에 ICMP 헤더가 오지만, 본 실험에서는 IP 헤더와 ICMP 헤더 사이에 HID를 포함한 HCP 헤더가 추가되어 있는 것을 확인할 수 있고, 또한 종단 호스트에서의 HID는 변경되지 않는 것을 확인 할 수 있다. 그림 17에서 패킷 덤프한 헥사코드에서 다양한 색깔로 MAC, IP, ICMP, HCP 등의 각 헤더를 구분하여 보였다. 이로써 PING 서비스가 HID 기반으로 정상 동작하는 것을 검증했다.

그림 17. PING 패킷 분석
Fig. 17. PING packet analysis.

2. WWW 실험

웹서비스 실험을 위하여 HID3 호스트에 웹서버를 구동시켰다. 위의 PING 서비스는 ICMP 프로토콜을 이용한 서비스이지만, WWW 서비스는 현 인터넷에서 가장 많이 쓰이는 TCP 서비스이다. TCP 의 대표적인 서비스인 WWW 서비스를 실험함으로 기존의 TCP 기반 서비스들이 HID 기반으로 동작할 수 있음을 검증했다.

앞서 언급하였듯이, HID3 호스트에 실험을 위한 웹

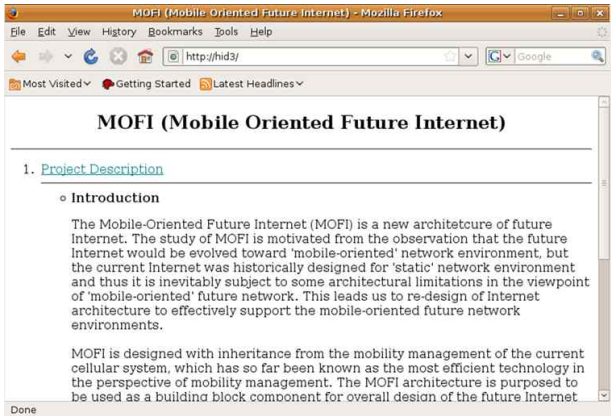


그림 18. WWW 실험 결과
Fig. 18. WWW Test Result.

서버를 구동시켰다. 웹서버를 구동시킨 후, 각 HID 호스트에서 그림 18처럼 웹 브라우저의 주소창에 http://HID3/ 를 입력함으로 HID 기반의 웹서비스가 가능함을 알 수 있다. 또한 홈페이지의 하위 링크도 정상적으로 링크가 연결되는 것을 확인할 수 있었다. 패킷의 분석은 PING 서비스 실험에서와 동일하므로 본 논문에서는 생략한다.

VI. 결 론

본 논문에서는 현재 인터넷의 문제점들을 해결하기 위한 미래인터넷 연구의 일환인 MOFI 시스템과 ID / LOC 분리 방안의 관련 연구들에 대해서 알아보고, HID 기반의 통신을 검증하기 위한 기본 테스트베드 구축 연구에 대해 기술하였다. 이동성 및 무선 환경을 고려한 MOFI 테스트베드를 구축하기 전 기본 테스트베드를 구축하여 HID 기반 통신의 가능성을 검증하기 위해 일반적인 인터넷 서비스인 PING, WWW 서비스를 실험하였다. 종단 호스트에서 Click 과 VirtualBox 가상머신을 이용한 MOFI_Agent 를 구현하여 기존 호스트들이 HID 기반의 통신을 가능하게 하였고, 또한 AR에서 Click 을 활용하여 HCP 헤더의 추가 및 변환 기능을 구현하였다. 구축한 테스트베드를 활용하여 HID 기반으로 인터넷 서비스를 실험한 결과 HID 기반의 통신 가능성을 충분히 검증할 수 있었다. 본 논문에서의 실험은 고정 호스트에 대해서 진행하였지만, 향후 연구로 이동 호스트를 고려한 테스트베드를 구축하고 실험할 계획이다.

참 고 문 헌

[1] 미래인터넷 포럼 홈페이지, http://fif.kr/
[2] Eddie Kohler, "The Click Modular Router", PhD Thesis, MIT, 2000.
[3] D. Farinacci, V. Fuller, D. Meyer and D. Lewis, "Locator/ID Separation Protocol (LISP)", IETF draft-ietf-lisp-07.txt, April 2010
[4] R. Moskowitz, P. Nikander, "Host Identity Protocol (HIP) Architecture", IETF RFC 4423, May 2006.
[5] M. Menth, M. Hartmann, D. Klein, "Global Locator, Local Locator, and Identifier Split (GLI-Split)", University of Würzburg Institute of Computer Science Research Report Series, No. 470, April 2010.
[6] HY Jung and SJ Koh, "Mobile-Oriented Future Internet (MOFI): Architecture and Protocols", Release 2.2, June 2010.
[7] 정희영, "이동환경 기반의 미래인터넷", 정보과학회지, 제 28권 제 1호, 2010년 1월.
[8] V. FUller, D. Farinacci, D. Meyer and D. Lewis, "LISP Alternative Topology (LISP+ALT)", draft-ietf-lisp-alt-05, October 18, 2010.
[9] VirtualBox 홈페이지, http://www.virtualbox.org/

저 자 소 개

정 회 진(학생회원)
2005년 충남대학교 전기정보통신공학부 학사
2007년 충남대학교 정보통신공학과 석사
2010년~현재 충남대학교 전자전파정보통신공학과 박사과정
<주관심 분야 : 인터넷, 미래인터넷, 이동인터넷, 이동통신>

민 석 홍(학생회원)
2005년 공주대학교 전기전자정보공학과 석사
2010년~현재 충남대학교 전자전파정보통신공학과 박사과정
2004년 한국전자통신연구원 BcN 시험기술팀 위촉연구원
2005년 디지피아㈜ 방송장비팀 연구원
2006년~2009년 ㈜엠티아이 연구2실 전임연구원
<주관심분야 : 무선 메쉬 네트워크, 데이터 통신, NetFPGA>

이 재 용(평생회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1995년 한국과학기술원 전기 및 전자공학과 박사
1990년~1995년 디지콤 정보통신 연구소 선임연구원
1995년~현재 충남대학교 정보통신공학부 교수
<주관심분야 : 초고속통신, 인터넷, 네트워크 성능분석>

김 병 철(평생회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1996년 한국과학기술원 전기 및 전자공학과 박사
1993년~1999년 삼성전자 CDMA 개발팀
1999년~현재 충남대학교 정보통신공학부 부교수
<주관심 분야 : 이동인터넷, 이동통신 네트워크, 데이터 통신>