ISBN 978-89-6211-469-0 98560

네트워크 가상화와 OpenFlow Swithing 기술

일자: 2009년 11월 30일
부서: 슈퍼컴퓨팅본부 기반기술개발실
제출자: 문정훈
jhmoon@kisti.re.kr

KiSTi www.kisti.re.kr
한국과학기술정보연구원
Korea Institute of Science and Technology Information
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

목차

서론

1. 네트워크 발전 동향과 미래인터넷

1-1. 현재 네트워크

인터넷의 발달은 현재 사회 공공 인프라로서의 역할과 산업전반에 걸친 다양한 응용과 필수적인 역할을 담당하고 있다. 현재의 인터넷은 1969년 ARPANET으로부터 시작하여 1981년 TCP/IP 기반구조의 표준이 정의 되고, 1983년 ARPANET에서 TCP/IP를 적용함으로써 시작되었으며, 1991년에 Web이 탄생하게 되었다. 이러한 발전을 거듭한 인터넷은 Web의 등장으로 일반 대중에게 급속히 전파되고, 통신, 신문, 방송에까지 범위가 확대되었고, 전자상거래, e-Banking, 전자정부로까지 확대되고 있다. 이러한 상황은 초기 인터넷의 시작시점에서 고려하지 못한 문제점들이 발생하게 되었고, 이러한 발전은 보안성, 관리성, 이동성, 품질보장 등이 중요한 기능으로 요구되기 시작되었고, 이동통신 인터넷과의 결합하려는 요구가 발행하게 되었다. 이러한 부분들에 대해서는 인터넷 시작 당시 고려하지 못한 것이며, 인터넷의 기본구조를 유지하면서 발전해오는 과정 중 더 복잡해지고 한계점이 생기게 되었다. 이러한 한계점들은 Backward Compatibility를 위한 인터넷 기본구조의 유지로 인한 한계점이며, 앞으로도 인터넷은 시간이 흐름에 따라 점점 더 복잡해질 것이며, 폭발적으로 증가하는 트래픽 양, 이에 따른 보안 문제들이 대두 될 것이다. 이와 같은 구조적인 문제점들을 해결하고자하는 시도가 미래 네트워크 및 미래 인터넷에 대한 연구의 시작이라 할 수 있다.

이러한 인터넷의 제한성과 한계성은 간략하게 다음과 같다 첫 번째로 보안성의 취약이다. 초기 사용자들은 'Trusted Users'로서 해커 및 기타 보안 문제들을 고려하지 않았다. 사용자에 대한 신뢰를 가정하여 많은 자유가 허용되었는데 이러한 구조에서 스팸메일, DDoS 등과 같은 공격과 악성 유저들의 피해를 막기에는 구조적 한계가 있으며, 이러한 한계를 극복하려는 노력은 더 복합 네트워크 형태로 확장되고 있다. 향후 전자 상거래, 은행, 전자정부, 스마트 전력망 등 사회 인프라로서의 역할을 감안할 때 근본적인 대비책이 필요하다. 두 번째로 이동성의 부족이다. 과거에는 유선망에 연결된 단말들이 대부분이었지만, 현재는 휴대폰과 노트북 등의 이동단말이 급증하고 있으며, 미래에는 현재의 상황에 차량 등의 다양한 이동단말이 주류가 될 것으로 예상이 된다. 이러한 이동성의 문제를 해결하기 위한 노력들이 진행되어 오면서 많은 성과를 낸 것이 사실이지만, 기술의 근본적인 한계로 본격적인 네트워크 이동성 서비스가 아직까지 활성화 되지 못하고 있는 실정이다. 세 번째로 안정성/신뢰성의 취약부분이다. 전화망의 경우 신뢰도가 99.999%이며, 1년 동안 허용되는 다운 시간이 5분이하여야 된다. (관리를 위한 의도적 다운을 포함해서), 현재 인터넷망은 전화망의 신뢰에 비해서는 현저히 낮은 상황이다. 이것은 장비 자체의 낮은 신뢰도와 인터넷 구조의 취약성이 더해진 결과이다. 현재의 인터넷은 그 기능이 네트워크보다 단말에 집중되어 있는데, 바이러스에 감염된 단말 또는 해킹 당하는 경우, 네트워크 설정이 잘못된 호스트에 의한 대량의 브로트캐스트 패킷의 발생, 불법 DHCP서버에 의한 고의 또는 실수에 의한 네트워크 접근 차단에 대한 취약성, 관리자의 실수 등이 있을 수 있다. 네 번째로 관리성의 부족이다. 현재는 고도의 숙련된 네트워크 관리자의 능력에 의존하고 있다. 네트워크의 분리, 라우팅설정, 보안설정, 트래픽 엔지니어링, BGP정책 설정, VPN관리 등, 네트워크 설정의 오류의 여지가 있으며, 이러한 결과는 네트워크에 치명적인 결과를 초래할 수 있다. 많은 네트워크 사고가 관리자의 잘못된 네트워크 설정에 기반하고 있다. 다섯 번째로 ISP비즈니

스 모델이 부족하다. 현재의 인터넷은 백본을 소유한 ISP는 망 접속료와 사용료가 수입의 전부이다. IPTV, YouTube 등의 대용량 트래픽을 유발하는 신규 서비스를 위한 망증설이 필요하지만 망증설에 따른 ISP의 경제적 이득이 미미하기 때문에 지속저인 투자가 어렵다. 이 외에도 대용량 전송 문제, 코어 라우터의 전력문제, 대역폭 보장 문제, Congestion Control 문제 등이 있다.

Figure1 Network Paradigm Shift

1-2. 미래 인터넷

미래인터넷의 동기는 앞서 언급한대로 기존 인터넷의 한계성을 극복하고 폭발적으로 늘어나는 트래픽과 다양한 서비스, 특히 보안성, 품질보장성, 이동통신과의 연계성들을 충분히 고려하여 기존 인터넷과는 호환을 전혀 고려하지 않는 가운데 현재, 미래의 요구사항을 수용하여 새로이 설계하는 방향으로 연구가 시작되었고, 이러한 동기를 바탕으로 각국에서 주요 프로젝트로 진행 중에 있다. 미래인터넷의 연구는 현재의 인터넷은 기존에 설치된 망의 전면적인 전환이 어렵기 때문인데, 현재의 인터넷은 Backward Compatibility가 가장 중요한 요구사항이여서 새로운 환경에서의 효과적인 연구와 개발이 어려운 실정이다. 이러한 상황에 따라 미래인터넷은 철저하게 현재 및 미래의 요구사항에 근거하여 새롭게 설계되어야 한다. 그리고 다양한 서비스들을 지원하기 위해서 통신망을 'Clean-Slate' 상에서 설계를 한다는 개념으로 시작하고 있다.

미래인터넷 연구의 시작은 2000년 DARPA에서 NewArch 프로젝트에서 처음 논의가 되어 2005년 미래인터넷 테스트베드인 GENI의 설계를 시작으로 본격적으로 미국에서 추진되었다. 미국의 미래인터넷 프로젝드로는 Future INternet Design (FIND), Global Envrionment for Networking Innovations (GENI)이며, 유럽에서는 FIRE (Future Internet Research and Experimentation), EIFFEL's Future Internet Initiative, EuroNGI & EuroFGI, 일본에서는 NICT's NeW Generation Network (NWGN), Japan Gigabit Network2 (JGN2)가 있으며, 우리나라에서는 Future Internet Forum (FIF, 2006)이 진행중에 있다. 세계 각국의 미래인터넷을 위한 연차별 계획은 아래의 그림과 같다.

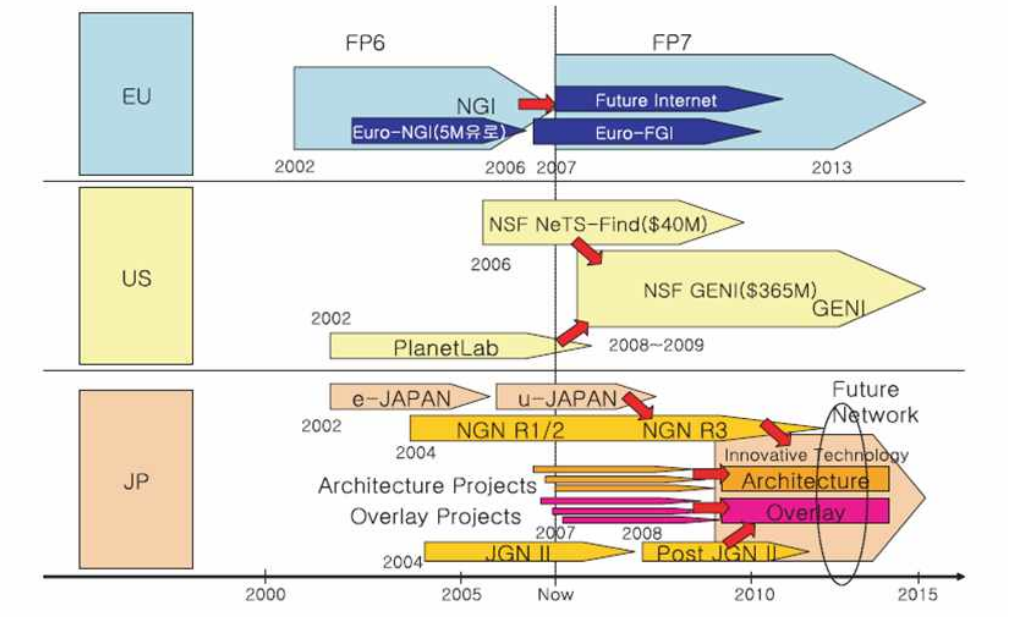


Figure2 Future Internet global roadmap

2. 네트워크 가상화

기존 인터넷의 한계점에 대한 인식과 미래사회에서 인터넷의 사회 인프라로서 주도적 역할을 고려하여 미래인터넷의 연구가 시작됨에 따라 미래인터넷 연구는 지금까지와는 전혀 다른 새로운 네트워크 아키텍쳐로의 발전 방향과 여러 아키텍쳐를 동시에 수용 할 수 있는 가상화 기반의 네트워크로의 발전 방향이 있다

현재의 네트워크 환경에서는 연구자가 실제 대규모 네트워크의 운용과 여러 가지 실험을 할 수 있어야 하지만, 현재 ISP의 네트워크를 운용해보고 실제 환경에서 실험해보는 것은 현실적으로 불가능하다. 차선책으로 실험연구망을 이용한다 하더라도 상업용 라우터로 구성된 연구망 라우터를 액세스하거나 자신이 원하는 프로그램을 네트워크 장비에 설치하는 것은 불가능하다. 이를 극복하기 위해서 여러 연구그룹에서는 하드웨어/소프트웨어 기반의 네트워크 가상화 개념으로 연구 중에 있다. 이러한 네트워크 가상화는 일반 연구자라도 자신만의 네트워크 환경을 구성하여 새로운 알고리즘과 기능을 구현할 수 있게 하면서도 다른 사용자의 네트워크 트래픽에는 전혀 영향을 미치지 않고 자유롭게 사용할 수 있는 중요한 연구 환경의 구성이다.

따라서 네트워크 가상화는 하나의 공유 네트워크상에서 사용 가능한 자원 즉, 네트워크, 네트워크 하드웨어, 네트워크 미디어, 네트워크 소프트웨어 등을 특정 목적을 위해 구성된 다수의 네트워크로 논리적으로 분리하는 기술로 정의 할 수 있다. 이것은 응용 프로그램 또는 사용자에게 자원을 물리적 특성을 보이지 않게 하고 논리적 자원만 보이게 하는 기술로도 설명 할 수 있는데, 오래전부터 서버 클러스터링과 파티셔닝, 가상머신, 그리드 컴퓨팅, 클러스터 컴퓨팅 등으로 진행되고 있으며, 네트워크 상에서는 VLI, VPN, VLAN 등으로 진행되고 있다. 물리적으로 서로 다른 하드웨어들을 하나의 공통된 형태의 가상 장치들로 만들거나 하나의 물리적 하드웨어 상에 여러개의 가상장치들을 생성할 수 있다.

따라서 가상화에 기초하여 하나의 물리적인 네트워크상에 다수개의 가상의 독립된 네트워크가 공존함으로써, 복수개의 네트워크 아키텍쳐와 서비스가 하나의 인프라상에서 동시에 동작할 수 있다. 사용자의 입장에서 보면 가상화를 통해 공유자원을 사용자간 서로 영향을 받지 않고 사용할 수 있으며, 인프라의 복잡도를 단순화하여 사용할 수 있는 장점이 있다.

특히 연구자의 입장에서는 여러 가상의 네트워크가 하나의 네트워크 인프라상에 공존할 수 있으므로 다양한 네트워크 아키텍쳐에 대한 연구를 할 수 있다. 또한 네트워크 사업자 입장에서는 고객이 요구하는 서비스를 신속히 구축할 수 있으며, 네트워크의 사용효율을 높일 수 있고, 네트워크 구축 및 관리의 자동화를 향상시킬 수 있으며, 유연성의 증가로 구축 및 운영비를 절감할 수 있다.

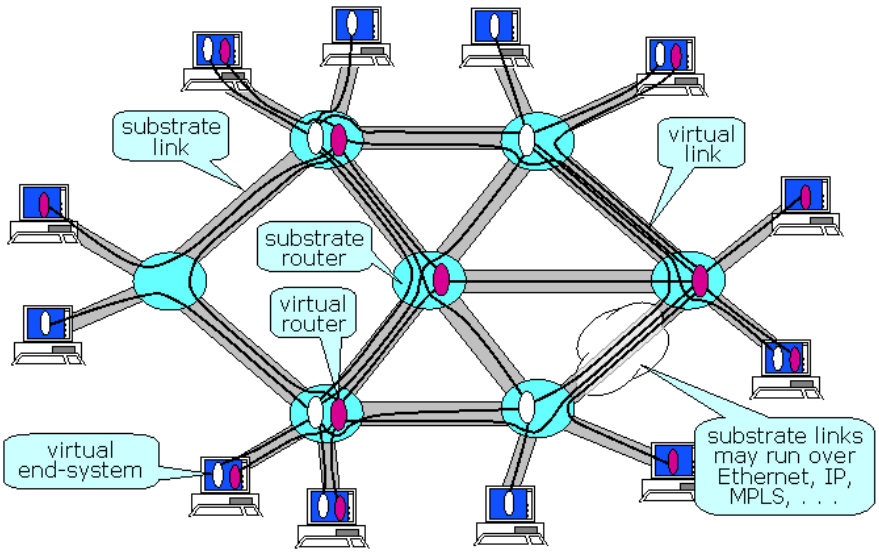


Figure3 Network Virtualization

미래인터넷에 대한 연구와 네트워크 가상화 분야에 대한 연구가 지속되고 있는 가운데 선진국에서는 미래인터넷 프로젝트들을 통하여 미래인터넷분야에 대한 기술 및 표준화 선점을 위해 연구에 주력하고 있다.

이러한 배경속에서 스탠포드 대학에서 OpenFlow라는 새로운 개념의 네트워크 테스트 환경을 개발했는데 이것은 실제 네트워크에 전혀 영향을 주지 않으면서 실제 네트워크 환경에서 연구자의 새로운 네트워크 프로토콜 및 실험이 가능한 기술이다. 실제 네트워크 환경에 적용이 가능하며, 실제 네트워크 환경에서 상용 트래픽에 영향을 주지 않는 가운데 연구를 수행할 수 있는 장점이 있다. OpenFlow는 인터넷 리엔지니어링시 실제로 어떻게 작동하는가를 연구하는 Clean-Slate 계획의 일부분이며, OpenFlow기술은 스위치 상의 flow table, Controller, OpenFlow protocol의 3개의 주요 부분으로 구성되어 있다. 이번 기술문서는 OpenFlow기술을 중심으로 미래인터넷, 네트워크 가상화 동향분석과 OpenFlow에 대한 기술을 분석한다.

2-1. 국내외 동향

미래인터넷에 대한 연구와 네트워크 가상화 분야에 대한 연구가 지속되고 있는 가운데 선진국에서는 미래인터넷 프로젝트들을 통하여 미래인터넷분야에 대한 기술 및 표준화 선점을 위해 연구에 주력하고 있다. 이러한 미래인터넷의 동향에 대해 대표적으로 미국의 GENI 유럽의 FIRE, 일본의 AKARI, 한국의 FIF 등을 살펴본다.

- 미국

미국은 2000년 DARPA의 NewArch 프로젝트의 제안으로 미래인터넷에 대한 프로젝트가 시작되었다. 여기에서 현재, 미래의 요구사항을 바탕으로 인터넷을 재설계하는 개념으로 시작되었다. 그리고 2003년 NSF에서 Clean-Slate 프로젝트가 논의 되면서 2005년 8월 NSF가 GENI Initiative를 소개하면서 본격적인 미래인터넷의 연구와 투자가 시작되었다. 2006년 NSF에 의해 FIND 프로젝트를 시작으로 미래인터넷을 위한 혁신적인 아키텍쳐 연구를 목적으로 2006년 26개 과제로 시작하여 2009년 4월 현재 42개 과제가 수행 중에 있다. 1단계로서 개별 아키텍쳐 컴포넌트들에 대한 다양한 연구를 시작으로 2단계에서는 1단계의 아키텍쳐 컴포넌트들이 통합된 전체 아키텍쳐 연구로 진행되고, 3단계로서는 유력한 미래인터넷 아키텍쳐 후보에 대한 본격적인 실험을 진행하는 계획이다. 2009년부터 1단계 연구를 지속하여 2단계로 진입할 예정에 있다.

· GENI

미국의 NSF는 미래인터넷 연구를 위해서는 다양하면서도 새로운 네트워크 아키텍쳐들을 동시에, 대규모 사용자들을 대상으로 하여 검증할 수 있는 대규모의 실제적 테스트베드가 필요하다고 판단되어 2005년부터 미래인터넷 테스트베드인 GENI의 개념 설계안을 완성하여 2008년도에 프로토타이핑 과제를 시작했다. 5가지의 GENI 프레임워크 제안서를 선정하여 Control Framework A, B, C, D, E로 구현하고 완성도에 따라서 최선의 구조를 선택하는 방식을 택해서 진행 중에 있다.

GENI의 5개 Control Framework Cluster는 다음과 같다.

- Cluster A Integration (uses DETER Control Framework)
- Cluster B Integration (uses PlanetLab Control Framework)
- Cluster C Integration (uses ProtoGENI Control Framework)
- Cluster D Integration (uses ORCA Control Framework)
- Cluster E Integration (uses ORBIT Control Framework)

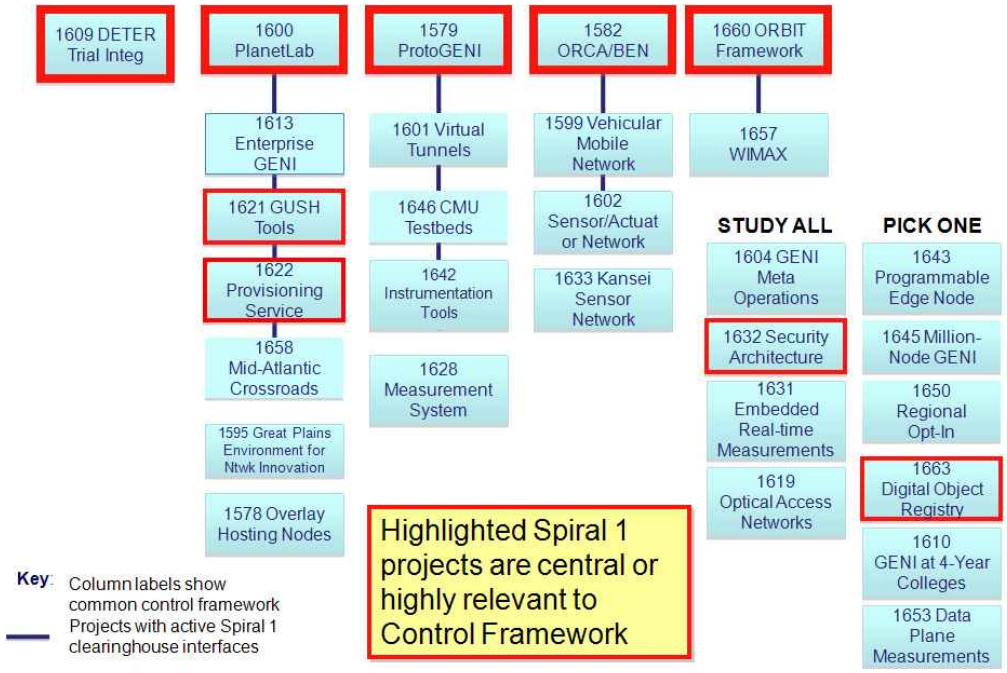


Figure4 GENI Spiral 1 Integration : 5 Control Framework Clusters

GENI에서는 자원 가상화 기반의 네트워크 구조를 통해 복수개의 미래인터넷 실험이 동시에 수행 될 수 있도록 하고 있으며, GENI는 기존의 3계층 이상에서의 네트워크 연구를 지원하는 PlanetLab의 개념을 확장하여 물리계층까지 가상화 개념을 적용하고 있다. GENI의 아키텍쳐는 3가지 레벨로 나눌 수 있는데 상위에는 GENI 설비를 액세스 할 수 있는 GENI사용자 서비스, 하위레벨에는 라우터, 링크, 프로세서 등 물리적 장비의 집합인 물리기판(Physical Substrate), 중간레벨에는 GMC (GENI Management Core)가 존재하는데, GMC는 추상화, 인터페이스, 코어 서비스 등 GENI 아키텍쳐 구성에 필요한 프레임워크를 정의한다. GMC에서 정의하고 있는 추상화는 컴포넌트, 슬라이스, 애그리케이트가 있으며, 이들의 정의는 다음과 같다.

컴포넌트는 GENI의 기본 구성요소로서 프로그래밍이 가능한 에지노드, 코어노드, 액세스 포인트 등이 컴포넌트에 해당된다. 컴포넌트는 물리적인 자원, 예를 들어 CPU, 메모리, 디스크, 링크 대역폭 등과 논리적 자원인 파일 디스크립터, 포트번호 등을 포함하는 자원들의 모음으로 구성된다. 각 컴포넌트는 컴포넌트 매니저인 CM을 통해 제어되고, CM은 컴포넌트상의 가상화된 자원인 슬리버를 생성/제어할 수 있는 인터페이스 제공, CPU, 메모리 점유율과 같은 컴포넌트 상태 정보를 보고하기 위한 자원 모니터링 인터페이스를 제공한다.
슬리버는 컴포넌트들을 구성하는 자원들의 일부분으로 구성된 부분 집합으로서 가상서버, 가상 라우터, 가상 스위치, 가상 액세스 포인트 등이 있다. 슬라이스는 사용자가 GENI상에서 실험하기 위해 부여 받은 자원들인 슬리버들의 모음이다. 사용자는 슬라이스상에서 실험을 수행한다. 애그리게이트는 GENI에서 서로 연관된 컴포넌트들을 모아 놓은 하나의 집합을 나타내는 GENI의 객체이다. 컴포넌트들의 집합을 관리하고 자원할당을 코디네이션 한다.

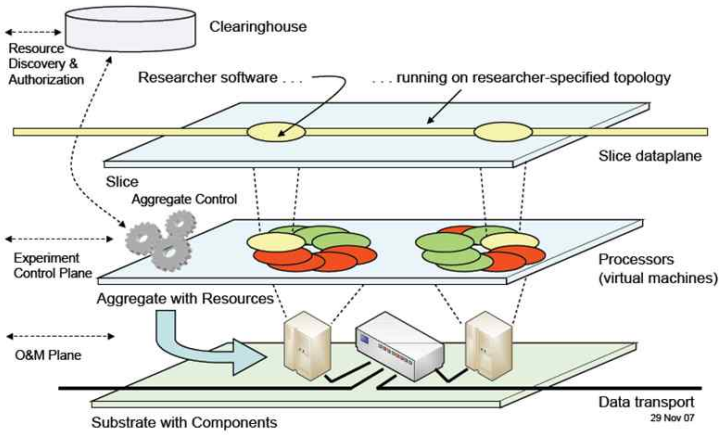


Figure5 Substrate, Aggregates, Slices of GENI

- 유럽

유럽의 미래 인터넷은 2006년 FP6 산하 EURO-NGI, EURO-FGI과제에서 미래인터넷 연구를 시작하였다. 2007년에 들어서 FP7을 통해서 본격적으로 연구를 시작하여 2008년 FP7 ICT분야 68개 프로젝트들이 미래인터넷 연구에 참여하여 FIRE, 4WARD, SENSEI, Trilogy 등의 개념으로 연구를 진행 중에 있다. 유럽의 미래인터넷의 연구범위는 혁신적인 미래인터넷연구와 광의의 미래인터넷 연구를 병행하는 특징을 가지는데 기존 인터넷 아키텍처를 크게 수정하지 않는 범위에서 1계층, 2계층 네트워크 기술 및 미디어, 서비스에 대한 연구를 수행하고 있다. 연구 형태로 보면 대체로 미국과 일본은 미래인터넷의 과제는 대부분 학교, 국책연구소 중심이지만 유럽의 경우 노키아, 에릭손, 알카텔-루슨트 등의 기업들의 활발한 참여가 특징이다. 이러한 배경은 혁신적인 미래인터넷 연구와 더불어 보다 넓은 차원에서 현재 인터넷을 기반으로 미래인터넷을 설계하는 연구 방향을 통해 많은 네트워크 기업들의 참여가 있는 것으로 보인다.

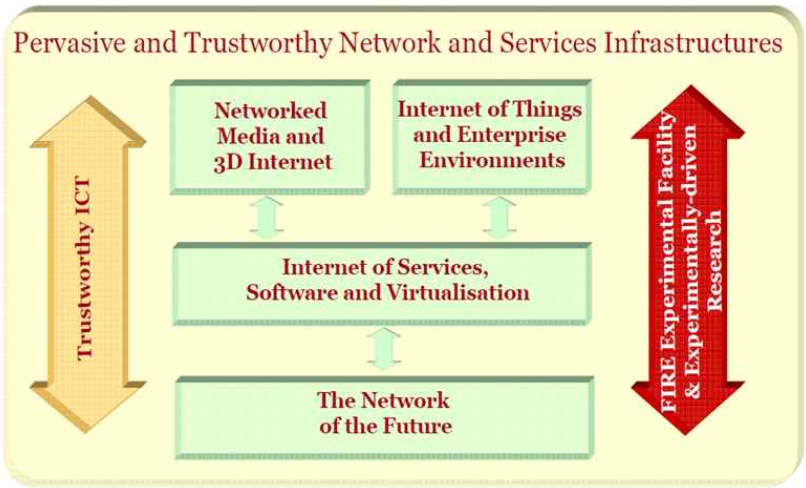


Figure6 Future Internet of FP7 ICT by EU

· FIRE (Future Internet Research and Experimentation)

현재 유럽의 FP7 ICT 산하 연구과제들로 구성된 유럽의 미래인터넷 프로젝트인 FIRE는 미래인터넷의 기본적인 요구사항인 현재 네트워크에 대한 한계성으로 새로운 네트워크 환경에 대한 요구와 장기적 관점에서의 접근, 그리고 시스템 레벨 차원의 실제적인 접근을 기본적인 방향으로 삼고 있다. Clean-Slate 기반의 새로운 미래인터넷 모델을 통한 다양한 분야의 융합 연구의 시너지 효과를 기대하고 있다.

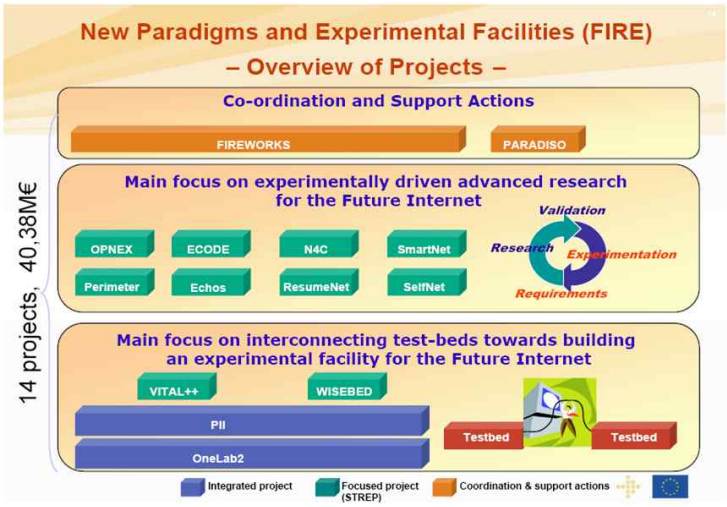


Figure7 Overview of FIRE Projects

- 일본

일본의 경우 미래인터넷에 대한 연구는 현재 인터넷에 기반한 차세대 네트워크에 대한 연구(ITU-T 기반)가 진행되고 있다. 이것은 현재의 인터넷에 기반한 것인데 NGN → NXGN(Next Generation Network)로의 진화를 의미하고, 일본의 미래 네트워크는 신세대 네트워크로 구분하여서 2007년부터 NWGN (NeW Generation Network)에 대한 연구가 시작되었다. 일본 미래인터넷의 특징은 NXGN과 NWGN의 동시진행인데 NWGN의 경우 미래인터넷으로 구분되며, 새로운 패러다임 기반의 장기적 과제 형식이다, 국책 연구소 중심의 연구로서 NICT내의 네트워크 아키텍쳐 그룹의 주도로 진행되고 있다. 또한 2011년까지 NWGN의 설계를 완료하고 구현/실험하여 최종 2015년까지 완성을 목표로 하고 있다.

이와 함께 NXGN은 현재 인터넷의 점진적 개선에 대한 연구개발로서 2008년 NWGN 개념 설계서 Ver1.1을 발표하였으며, 여기에는 광패킷 스위칭, 광패스 스위칭이 결합된 옵티컬 코어망, 새로운 광 액세스망 구성, 패킷기반 무선다중접속방법, ID/Locator 분리 주소체계, Cross Layer구조, 보안구조, QoS 라우팅 그리고 단순해진 IP Layer 구조, 네트워크 가상화 연구 등이 포함되어 있다.

이러한 미래인터넷 연구에 대해서 AKARI 프로젝트를 중심으로 설계, 구현, 테스트, 실험 등의 연구를 진행 중에 있다

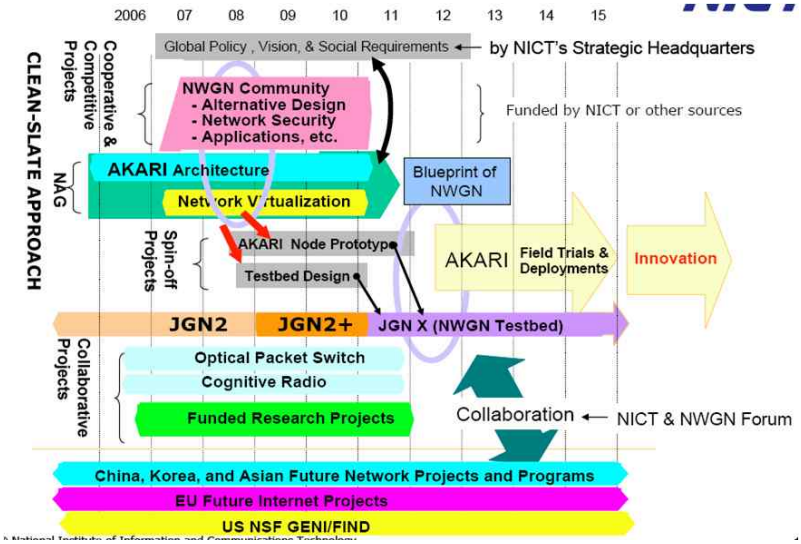
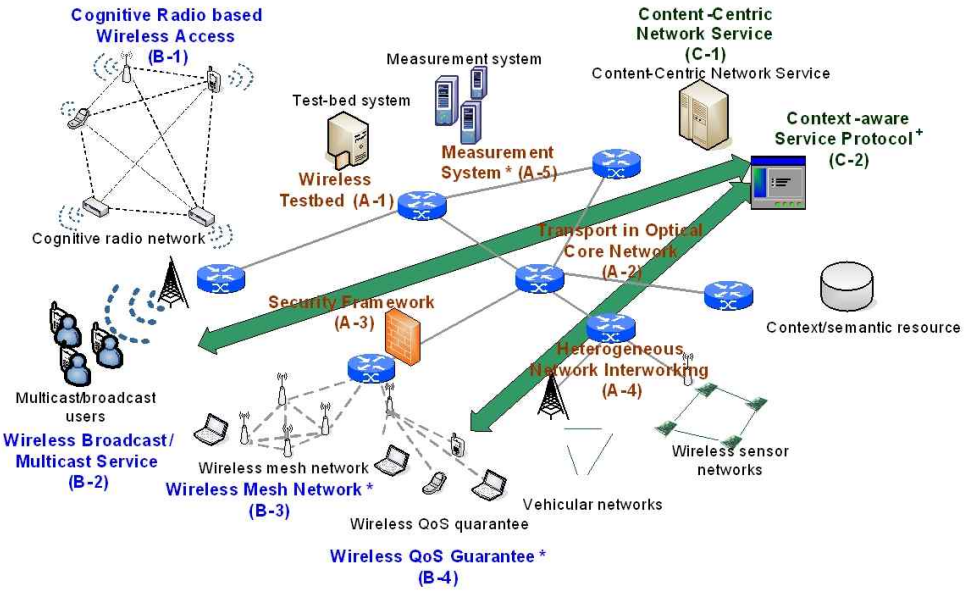


Figure8 AKARI & NWGN R&D Plan

- 한국

2006년 9월 미래인터넷 포럼 (FIF)이 발족되어 미래 인터넷 핵심기술 연구를 목적으로 원천 기술 개발 과제, 표준화 연계 과제 등을 수행하고 있으며, 새로운 형태의 미래인터넷 아키텍쳐를 설계하고 해당 요소기술들의 개발과 개발된 요소기술에 대한 표준화를 추진하고 미래인터넷 연구 개발 및 산업발전을 위한 연구주제의 기획을 세부 목표로 추진 중에 있다. 구체적인 연구 내용으로는 아키텍쳐, 무선기술, 서비스, 테스트베드, 정책 등의 5개 워킹 그룹을 구성하여 진행 중에 있다. 미래인터넷 아키텍쳐 연구분야에서는 Femtocell-based Testbed, Optical 망 관리 기법 연구, DTN환경에서의 Routing 프로토콜 제안, 서울대 DTN Testbed, 간접적인 망 성능 측정을 위한 설계 등이 진행 중에 있으며, 무선기술연구분야에서는 Collaborative Spectrum Sensing for CR system, CR ad hoc MAC Scheme, Cognitive Radio의 주파수 공유 및 활용 방안 연구, WMN Protocol Implementation, KAIST Mesh Testbed : Common Code for Future Internet, Dealing with Sudden Bandwidth Changes in TCP의 연구가 진행 중에 있다. 또한 미래인터넷 서비스 연구분야에서는 컨텐츠 기반 네트워킹 아키텍쳐, 사회적 관계 인식 라우팅 프로토콜 과제가 진행 중에 있다. 테스트베드 분야에서는 충남대, 경남대, 광주과기원 등에서 테스트베드 환경에 대한 연구를 진행 중에 있으며, 정책분야에서는 연구기획 및 표준화 분야에서 미래인터넷 대형 연구과제 기획 (필수요소 기술 및 세부 연구 개발 기술 발굴)과 대형 연구 기획 보고서 과제가 진행 중에 있다. 국내의 상황은 아직 국가적인 미래인터넷 연구 계획의 체계적인 모델은 없으며, 지식경제부, 교육과학기술부, 방송통신위원회, 행정안전부 등의 개별 부처 차원의 개별과제로서 미래인터넷 연구를 수행중이다.



Internet Architecture : A Wireless Access : B Service and Application : C	Seoul National University KAIST (*) ICU (+)

Figure9 Research topics and areas of FIF

3. OpenFlow 기본 개념

현재의 네트워크 환경은 전세계적으로 이미 대규모의 네트워크 장비들이 설치되어 있고, 기존 프로토콜들이 운용 중에 있기 때문에 새로운 네트워크의 혁신은 실체 환경에는 크게 영향력을 미치지 못하고 있다. 왜냐하면 높은 진입장벽으로 인해 새로운 네트워크 아이디어들은 실험실 환경에서만 적용될 뿐이며, 연구자들의 새로운 시도와 연구는 실제 네트워크 환경과는 괴리가 크기 때문이다. 따라서 오늘날 새로운 네트워크 연구개발과 적용을 위한 대규모 또는 광범위하면서도 실제적인 실험방법은 없다.

이러한 상황을 보통 네트워크의 화석화라고 하는데 이러한 문제에 대해서 미국의 미래인터넷 테스트베드인 GENI처럼 프로그래머블 네트워크를 만드는 차원에서 OpenFlow가 개발되었다. 미국의 GENI는 FIND 프래그램의 R&D를 테스트할 수 있는 미래인터넷 테스트베드로서 새로운 네트워크 아키텍쳐와 분산 시스템들과 함께 Nationwide 규모의 연구와 실험을 실제 환경에서 할 수 있도록 테스트베드를 제공한다.

따라서 OpenFlow는 연구자들에게 실제 환경하에서 실험 및 테스트를 가능케하는 기술로서 먼저 캠퍼스 네트워크에 실험적으로 운영되고 있으며, 대학 간 네트워크로 확장되고 있으며, 연구망을 활용한 원거리 환경하에서도 데모와 테스트들이 진행되고 있다.
먼저 OpenFlow의 기본개념으로서 다수의 독립 네트워크 환경의 제공이 되어야 하고, 가상화 기반의 프로그래머블 네트워크 환경이 제공되어야 한다. 이것은 프로그래머블 스위치와 라우터를 의미하는데 OpenFlow는 실제 네트워크 환경에서 일반 트래픽에 영향을 미치지 않고, 원하는 라우터와 스위치를 통해 테스트에 필요한 환경을 임의로 설정하여 새로운 기술들을 테스트할 수 있게 한다. 따라서 이러한 상황을 정리하고 향후 미래에 요구되는 기본개념들을 정리하면 다음과 같다.

- Isolation
- Virtualization & Programmable
- Open development environment
- Flexible definition of flow

OpenFlow Switch의 구체적인 아이디어는 다음과 같이 정리된다. 대부분의 이더넷 스위치 또는 라우터들을 flow table을 가지고 있는데, 벤더들의 제품들은 서로 다른 형태의 flow table을 가지고 있다. 이러한 상황을 하나의 공통된 flow table을 사용할 수 있다면 flow 기반하에서 다양한 활용이 가능하다. 따라서 OpenFlow의 목적은 대다수의 스위치와 라우터에서 공통으로 사용될 수 있는 공용 flow table을 만드는 것에 있다.
이러한 방법을 사용할 때 네트워크 관리자는 상용 트래픽과 연구용 트래픽을 분리하여 운영할 수 있게 되는데, 이러한 flow기반으로 연구자들은 자신들의 flow를 통해 원하는 라우터 또는 스위치에서 자신들만의 작업을 할 수 있게 된다. 따라서 연구자들은 새로운 라우팅 프로토콜, 시큐리티 모델, 어드레싱 스킴 등의 새로운 시도들을 할 수 있으며, 이러한 경우에도 상용 트래픽과 독립적으로 분리되어 있어서 서로 간 영향을 주지 않고 실제 망 환경에서의 실험이 가능하게 된다.

또, OpenFlow기술은 제어평면 (Control Plane)과 데이터평면 (Data Plane)을 분리하는 모델로서 Switch의 datapath는 Flow Table, Action으로 구성되며, 각각의 flow entry와 함께 조합을 이루게 된다. 이러한 action의 집합은 OpenFlow Switch의 확장성을 용이하게 할 수 있으며, 고성능 저비용의 datapath 환경을 구축할 수 있다.

3-1. OpenFlow Switch의 구성

기본적으로 OpenFlow Switch의 구성은 3부분으로 구성된다. Flow table, Secure channel 그리고 제어를 담당하는 Controller로 구성된다. 그리고 Controller와 OpenFlow Switch간은 OpenFlow protocol을 통해 필요한 통신을 한다. 아래의 구성도는 간단한 OpenFlow Switch와 Controller 기반의 개념도이다.

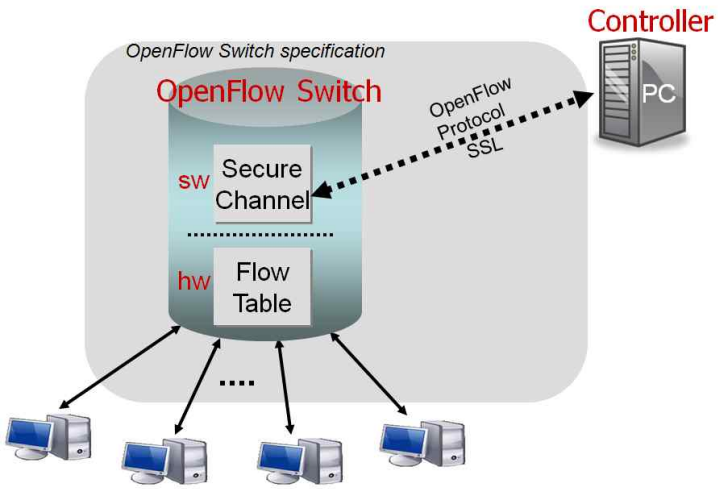


Figure10 Basic Concept of OpenFlow Switch and Controller

OpenFlow swtich에서 사용되는 flow의 정의는 다양하게 될 수 있는데 이러한 flow들에 대한 사용은 flow table의 기능 구현에 의해서 정의된다. 예를 들어 어떤 하나의 flow는 TCP connection을 맺을 수 있고, 특정 MAC 또는 IP address에서 오는 모든 패킷들을 같은 VLAN 태그를 붙일 수 있으며, 같은 스위치 포트에서 들어오는 모든 패킷들에 대해서 같은 일을 할 수 있다. 그리고 Non-IPv4 패킷들에 대해서도 정의된 action을 수행함으로써 그 기능을 할 수 있다. 따라서 flow는 매칭이 되는 모든 패킷들에 대해서 정의된 규정에 따라 처리를 할수 있게 된다.

3-1-1. Flow Table

Flow table은 flow entry의 집합으로서 구성이 되는데 각각의 flow entry들은 action 필드가 포함되어 있으며, 스위치로 incoming되는 패킷들에 대해서 어떻게 처리를 하는지 정의된다. 기본적으로 모든 패킷들은 스위치에서 flow table상의 flow entry와 비교되고 비교값에 의해서 정의된 action에 따라 처리된다. 만약 flow table의 flow entry에 해당 패킷에 대한 정의가 되어 있지 않다면 이 패킷은 Secure channel을 통해 Controller로 보내지며, Controller는 전송되어 온 패킷에 대해서 어떻게 처리 할 것인가를 결정하여 OpenFlow

Switch의 flow table에 새로운 flow entry를 추가 또는 삭제를 통하여 이 패킷에 대한 처리를 수행 한다.

- **Flow table**의 entry 구성은 다음과 같은 3개의 fields로 구성된다.
  - · Packet Header : flow를 정의한다.
  - · Action : 패킷들의 다음 프로세스를 정의한다.
  - · Statistics : 각각의 flow들의 packet의 수 (Packet Counter), byte의 수 (Byte Counter), 마지막 패킷이 매칭된 시간 등에 대한 정보 (Duration time of session) 등을 나타낸다.

아래의 그림은 Type0 OpenFlow Switch의 flow table entry이다.

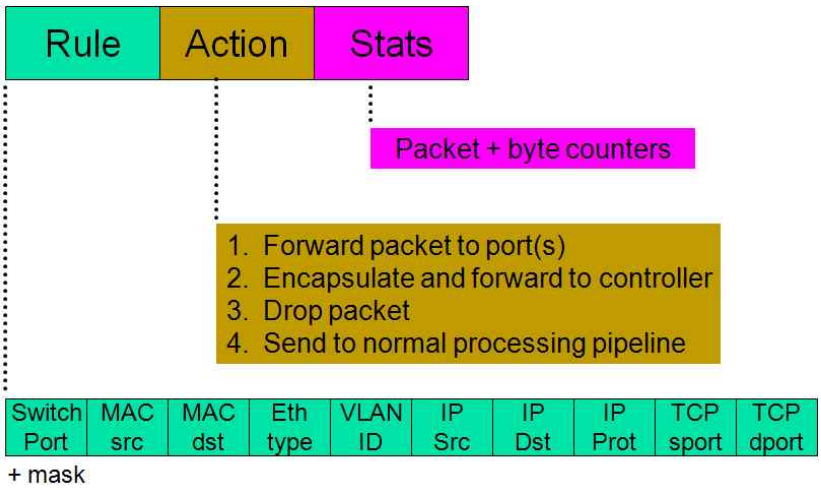


Figure11 Flow table entry of Type0 OpenFlow Switch

이러한 각각의 정의에 따라서 첫 번째 'Type0' 스위치를 정의하고 이에 따라서 flow header는 10-tuple로 정의하여 사용한다. 아래의 그림은 Type0 OpenFlow Switch의 10-tuple 그림이다.

In port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

Figure12 The header fields matched in a "Type0" OpenFlow Switch

위의 그림과 같이 TCP flow의 경우 10개의 field가 매칭이 될 수 있는데, 각 header field는 wildcard로서 flow aggregation을 허용할 수 있다. 예를 들면, 특정 VLAN상의 모든 트래픽에 대해서 적용할 수 있는 VLAN ID를 정의 할 수 있다. 따라서 10-tuple의 정보를 통해서 incoming port로 들어오는 패킷들에 대해서 다양한 처리를 할 수 있게 된다.

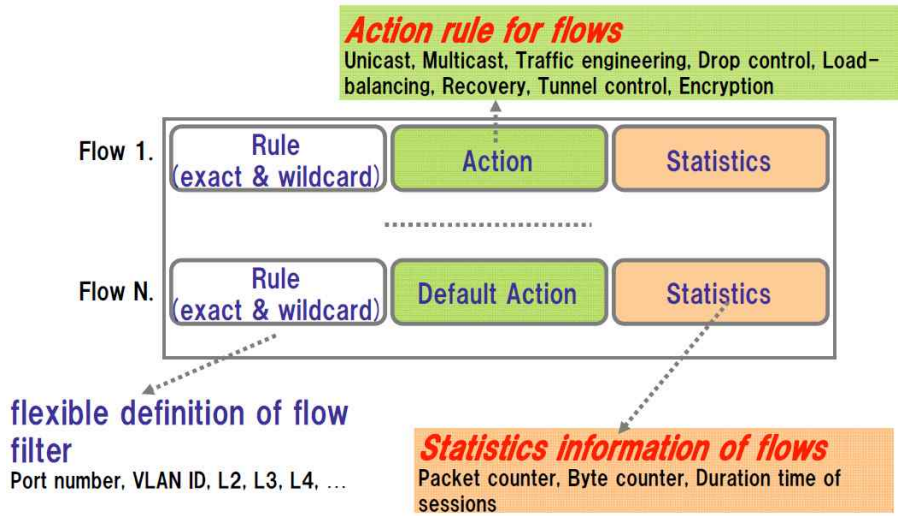


Figure13 Definition of Flows

간단한 패킷 매칭 과정에 대한 소개는 아래의 그림과 같다.
OpenFlow Switch 내의 flow table과 incoming 패킷의 매칭과정을 다음의 그림과 같이 알 수 있다. 네트워크상에서 해당 스위치로 패킷이 들어오게 되고 802.1d STP processing을 거쳐 flow lookup를 통해 match될 경우 정의된 aciton을 적용하고, match가 되지 않을 경우 Secure channel로 보내진다.

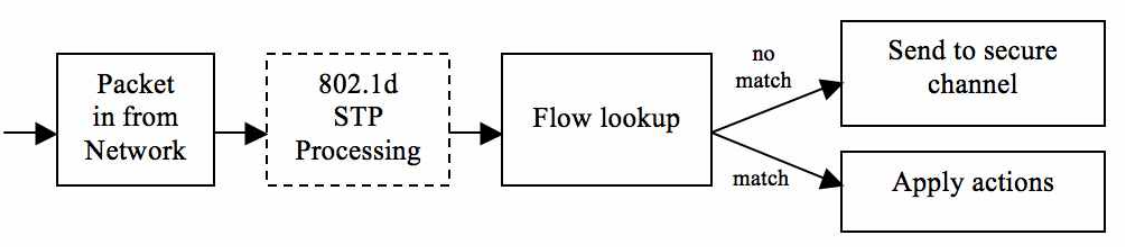


Figure14 Matching process

이런 과정을 통하여 모든 incoming 패킷들에 대해서 동일한 과정을 거치게 되고 flow entry를 통해 해당 패킷들을 구분하게 된다.

각각의 flow entry들은 간단한 action과 조합을 이루게 되는데 이러한 조합을 통하여 총 4가지 기본 작업을 수행하게 되며, 4가지 작업 중 3가지는 Dedicated OpenFlow Swtich에서 동작하며, OpenFlow enabled Switch에서는 4가지 작업 모두 수행하게 된다.

- Dedicated OpenFlow Switch : 단순한 형태의 OpenFlow Switch로서 dumb형태의 datapath를 가지며, Normal Layer2와 Layer3 프로세싱을 지원하지 않는다, 그리고 외부 원격 Controller에 의해 정의된 포트 사이에 대해 패킷의 포워드 기능을 가진다. 아래의 ①~③번까지의 작업을 수행 한다.
  ① Forward : 패킷들을 주어진 포트로 보낸다, 이것은 주어진 패킷에 대해서 네트워크를 통해 라우팅하는 것과 같은 의미이다.

② Encapsulate and forward : 이런 경우는 패킷을 캡슐화하여 Controller로 보낸다. 이 과정은 먼저 패킷을 Secure channel로 보내고 그런 다음 캡슐화되어 Controller로 보낸다. 이러한 경우는 incoming 패킷들 중에서 flow entry에 매칭 데이터가 없는 경우 이루지는 과정이다.

③ Drop : 보안적으로 사용되어 질 수 있는데 예를들어 DoS 공격이라든지, End-hosts로부터 나오는 브로드캐스트 중 가짜를 줄이기 위해 패킷을 'Drop'하는 기능을 한다.

④ Forward normal pipeline : OpenFlow Switch를 통해 연구용 트래픽과 일반 트래픽을 분리하는데 일반 트래픽의 경우 OpenFlow enabled switch를 통해 Normal Layer2와 Layer3 pipeline으로 forwarding한다. (이 기능은 OpenFlow enabled switch에서 적용된다.)

- **OpenFlow-enabled switch** : OpenFlow Protocol, Secure channel, Flow table을 기존 상업용 라우터와 스위치에 추가하여 만든 OpenFlow Switch이다. 기본적으로 flow table은 상업용 스위치 또는 라우터의 TCAM (Ternary Content Addressable Memory)을 사용하고, Secure channel과 OpenFlow protocol은 시스템 OS에서 제공 할 수 있도록 개발되어 있다. 다음 그림은 OpenFlow-enabled Switch를 기반으로 구성된 네트워크 구성도이다. 여기에서는 상업용 스위치와 액세스 포인트를 컴포넌트로 사용했다. 이 경우 모든 OpenFlow Switch의 flow table은 하나의 외부 Controller에 의해서 control된다. OpenFlow protocol은 Controller가 다수개의 스위치를 관리할 수 있도록 기능을 제공하고 기본적인 4가지 작업 중 4번째 작업을 추가적으로 할 수 있다.

④ Forward normal pipeline : OpenFlow Switch를 통해 연구용 트래픽과 일반 트래픽을 분리하는데 일반 트래픽의 경우 OpenFlow enabled switch를 통해 Normal Layer2와 Layer3 pipeline으로 forwarding한다.

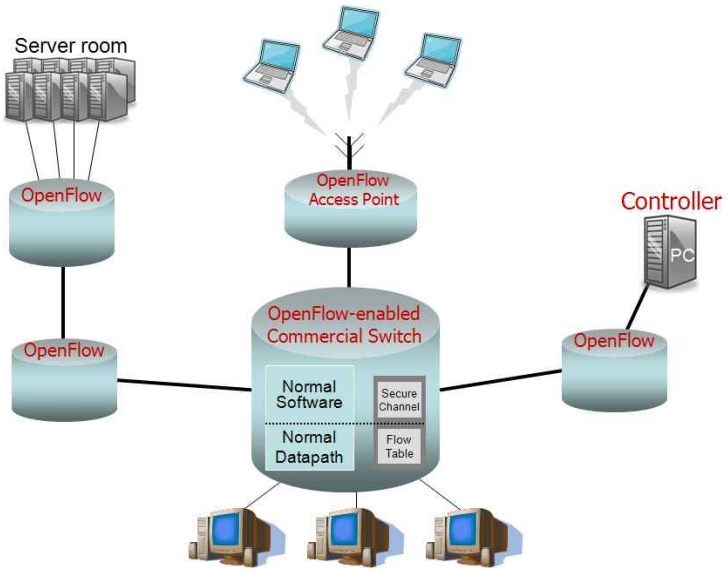


Figure15 Example for Network of
OpenFlow-enabled commercial switches and routers

지금까지 'Type0' OpenFlow Switch 환경에 대한 기본적인 개념이었으며, 향후 추가적인 기능과 개념들을 정의하여 'Type1' OpenFlow Switch로 정의 될 예정이다.

지금까지 기본적인 개념을 소개했으며, 실제적으로 flow entry의 action fields에서 수행되는 4가지 명령과 Secure channel과 Controller간 OpenFlow protocol을 통한 명령의 구체적인 예는 다음과 같다.

- Required Action : Forward

보통 OpenFlow Switch에 요구되는 action으로서 Type0 switch는 반드시 들어온 패킷을 물리적 포트나 다음 가상 노드로 보내야 한다. 따라서 다음과 같은 action들이 정의 된다.

· ALL : Incoming 인터페이스를 제외한 모든 인터페이스로 패킷을 보낸다.
· CONTROLLER : 패킷을 보안적으로 캡슐화해서 Controller로 보낸다.
· LOCAL : 패킷들을 로컬 네트워킹 스택으로 보낸다.
· TABLE : packet-out 메시지를 위해 flow table 내에서 action을 수행한다.
· IN-PORT : 패킷을 들어온 포트로 다시 보낸다.

- Optional Action : Forward

이외에 OpenFlow-enabled switch의 경우 다음과 같은 2가지의 Optional action이 추가된다.

· NORMAL : 기존의 스위치에 의해 정의된 전통적인 방법으로 패킷을 forwarding 시킨다. (Layer2, VLAN, Layer3 프로세싱) Type0 스위치는 VLAN field를 체크하여 패킷을 포워드 할지 NORMAL 프로세싱 해야 할지 결정한다.
· FLOOD : Incoming 인터페스를 제외한 최소 spanning tree 방향으로 패킷을 FLOOD 시킨다.

- Required Action : Drop

매칭이 된 패킷들에 대해서 flow entry에 구체적인 지시가 없는 경우의 모든 패킷은 'drop'된다.

- Optional Action : Modify-Field

OpenFlow의 유용성이 아주 커질 때를 대비해서 실제 네트워크와의 효과적인 통합을 위해서 Field를 수정할 수 있는 기능이다. 예를 들면 VLAN modification action을 지원하여 효과적으로 기존 네트워크와 통합을 할 수 있다.

3-1-2. Secure channel

Secure channel은 각각의 OpenFlow Switch와 Controller간에 인터페이스 역할을 한다. 이 인터페이스를 통해 Controller는 각각의 OpenFlow Switch를 관리하고 제어를 하게 되는데 스위치로부터 'events'를 받거나, 스위치로 'packets'를 보내고, 각각의 OpenFlow Switch가 가지고 있는 flow table을 제어한다. Secure channel의 모든 메시지들은 OpenFlow protocol에 의해 캡슐화 되어 전송된다.

- OpenFlow Protocol : Open/Standard 형식의 스위치와 Controller간의 통신을 담당

한다. 3가지 타입의 메세시지 형태인 cotroller-to-switch, asynchronous, symmetric으로 구성된다. 각각은 복수의 sub-type을 가진다.

- Controller-to-switch message : Controller에 의해서 초기화되고 직접적으로 스위치를 관리하거나 스위치 상태를 파악하는데 사용된다. 스위치로부터 응답을 요구할 수도 있고, 요구하지 않을 수도 있다.
  · Features : SSL 세션으로 만들어지며, Controller는 스위치에 'feature requirement message'를 보낸다. 스위치는 반드시 메시지 요구에 맞게 스위치에 의해서 제공되는 성능 및 기타의 정보를 회신한다.
  · Configuration : Controller는 스위치에 'Configuration parameters'를 셋팅하거나 질의 할 수 있다. 그리고 스위치는 질의에 대해서만 Controller에 회신한다.
  · Modify-State : 스위치의 관리 상태를 위해 Controller에서 보내는 메시지로서, 첫 번째 목적은 스위치의 flow table에 있는 flow에 대한 add/delete와 같은 수정이다.
  · Read-State : 스위치의 flow table 정보, ports 정보, 각각의 flow entry들의 정보를 모으기 위해서 Controller에 의해서 사용되는 메시지이다.
  · Send-Packet : 스위치에서 패킷들에 대해서 특정 포트 또는 다음 프레세tm로 보내기 위해 Controller에서 사용하는 메시지이다. 이 메시지를 통해 스위치들에 들어오는 패킷들이 다음의 스위치, 또는 라우터로 이동을 하게 된다.

- Asynchronous message : 스위치에 의해서 초기화되고, 네트워크 events에 따라 Controller를 업데이트하고, 스위치 상태의 변화를 업데이트하는데 사용된다. 이 메시지는 symmetric 메시지와는 반대로 스위치에서 Controller로 특별한 요구없이 보내는 메시지로서 스위치의 변화와 네트워크상의 변화 등에 대한 정보를 Controller에 보낸다. 여기에는 4개의 asynchronous message 타입이 있다.
  · Packet-in : flow entry에서 매칭되지 않는 모든 패킷들에 대해서 packet-in event를 Controller에 보낸다. (또, 패킷이 매칭은 되지만 'send to Controller" action과 함께인 경우) 이때 스위치가 충분한 메모리를 가지고 버퍼에 패킷을 보관할 수 있다면 Controller로 보내지는 packet-in event는 해당 패킷을 일부만 보내게 된다.(기본 128byte만 보내게 된다) 이 경우 Controller에 의해서 해당 패킷에 대해 forwarding 해야 하는 경우 버퍼 ID를 활용하여 패킷의 나머지부분까지 forwarding한다. 만약 스위치가 충분한 메모리를 가지지 못한다면 패킷을 일부가 아닌 전부를 보낸다.
  · Flow Expiration : 스위치의 flow의 유효기간, 패킷과 바이트 수를 포함한 flow expiration event를 보낸다. Flow expiration은 컨트롤러에 의해 활성화된 configuration message를 통해서만 설정할 수 있다.
  · Port-status : 스위치는 port configuration state가 변경되었을 때처럼 port-status 메시지를 Controller에 보낼 수 있다. 여기에는 포트 변경 정보 등이 포함된다.
  · Error : 스위치들은 Error 메시지를 통해서 Controller에 여러 가지 문제들에 대해서 알릴 수 있다.

- Symmetric message : 스위치 또는 Controller에 의해서 초기화되고, 특별한 요구 없이 보내지는 메시지로서 'Hello', 'Echo', 등의 여기에 해당된다.

· Hello : 스위치와 Controller간 처음 starup할 때 서로 간에 보내는 메시지이다.
· Echo : Echo request/reply 메시지는 스위치 또는 Controller에서 서로 간에 보낼 수 있는 메시지로서 반드시 회신을 해야 한다. 이 echo 메시지는 스위치와 Controller사이에 latency, bandwidth, and/or liveness 정보를 표시할 때 사용한다.
· Vendor : 이 메시지는 표준적인 방법으로 OpenFlow Switches에 추가적인 기능들을 제공할 때 사용되며, 향후 OpenFlow의 발전에 따라 추가적인 사항들이 발생할 때 사용된다.

3-1-3 Controller

Controller는 flow table의 flow entry들을 추가하거나 제거하는 기능을 하는 외부의 독립된 시스템이다. Controller는 단순한 형태의 Static Controller와 복잡한 형태의 Controller가 있다.

- Static Controller
  · PC상에서 간단한 application이 작동으로 실험중인 장비들 간의 정해진 시간동안 static flow를 생성하게 된다. 이때 생성되는 static flow는 보통 네트워크상에서 VLAN과 흡사한 역할을 한다. 결과적으로 상용 트래픽에 대해서 실험 트래픽을 독립적으로 구성하게 된다. 이러한 관점에서 OpenFlow는 VLAN의 한 기술로 볼 수 있다.

- Sophisticated Controller
  · 실험의 진행에 따라 flow를 동적으로 추가 삭제할 수 있다. (Dynamically Add/Remove flow)
  · 다수의 사용자 지원
  · 사용자간 서로 다른 계정의 사용
  · 서로 다른 flow-set상에서 다수의 독립적인 실험 환경 제공
  · Controller상에서 Policy Table의 사용 가능

하나의 Controller로 여러 개의 OpenFlow Switch를 관리하게 되며, 스위치들에 들어오는 패킷들은 기본적으로 각 OpenFlow Switch내의 flow table을 참조하여 정의된 action에 따라 라우팅하고, flow table내에서 매칭되는 entry가 없을 경우 Secure channel을 거쳐 Controller로 보내지게 되어 Controller에서 정의되지 않은 패킷에 대한 정의를 통해 해당 패킷을 라우팅 한다. 기본적으로 OpenFlow Switch와 Controller가 초기화 될 때 Controller를 통해 특정 사용자의 flow table이 전송되며, 전송된 flow table을 통해 해당 패킷들에 대한 처리를 하게 된다.

OpenFlow 프로그램에서는 기본적인 Controller 기능을 통해 단순한 형태의 Controller기능을 수행하고, NOX라는 전용 Controller 프로그램을 통해서 좀 더 복잡한 Controller의 기능을 수행한다.

- NOX

Network Operating System의 기본 개념을 수용한 Controller로서 네트워크 제어 플랫폼으로 정의한다. 연구자들은 원하는 네트워크 시나리오를 application으로 작성하여 NOX상에서 application을 적용함으로써 OpenFlow Switch를 control하여 실험을 할 수 있다. 다음의 그림은 NOX 기반의 OpenFlow 네트워크의 그림이다.

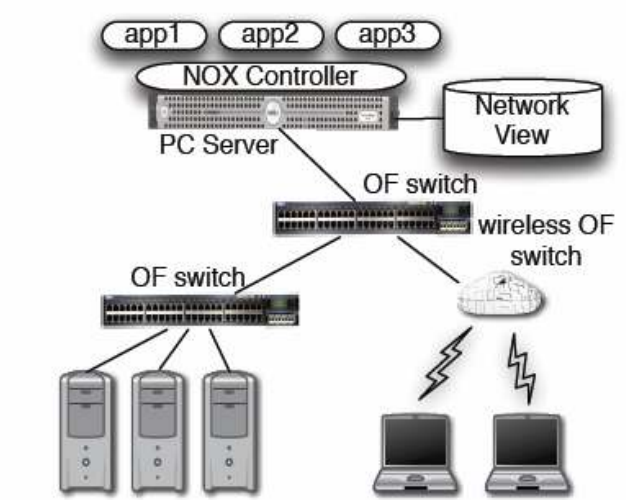


Figure16 Component of a NOX-based network

OpenFlow swtich를 활용하여 다음과 같은 활용을 할 수 있다.

- Network Management and Access Control
- VLANs
- Mobile wireless VoIP clients
- A non-IP network
- Processing packets rather than flows

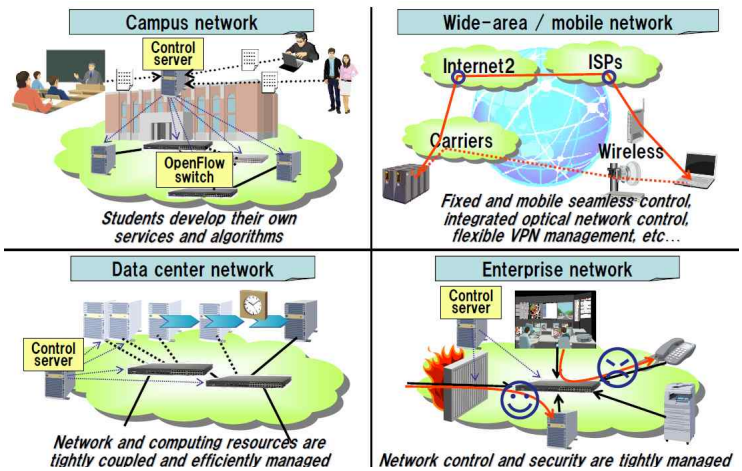


Figure17 OpenFlow Use cases

OpenFlow Switch의 기본 동작의 기본 동작 흐름은 다음의 그림과 같다.

- Controller의 User's code에 의해서 각 OpenFlow Switch로 flow table이 셋팅
- User's의 한쪽 End-host에서 트래픽을 발생
- 첫 번째 스위치에서 flow table을 참조하여 매칭을 확인
- 매칭이 된 패킷은 flow table entry의 action에 따라 라우팅
- 매칭이 되지 않는 패킷의 경우 OpenFlow Switch의 Secure channel로 보냄
- Secure channel과 Controller사이에서 SSL기반의 OpenFlow protocol로 통신
- Controller로 보내진 패킷은 action 정의를 통해 다음 프로세싱이 결정
  (스위치로 새로운 flow entry의 추가 또는 삭제를 통해서 진행한다.)
- 다시 OpenFlow 스위치로 돌려보내지고 Controller의 결정에 따라 라우팅 됨

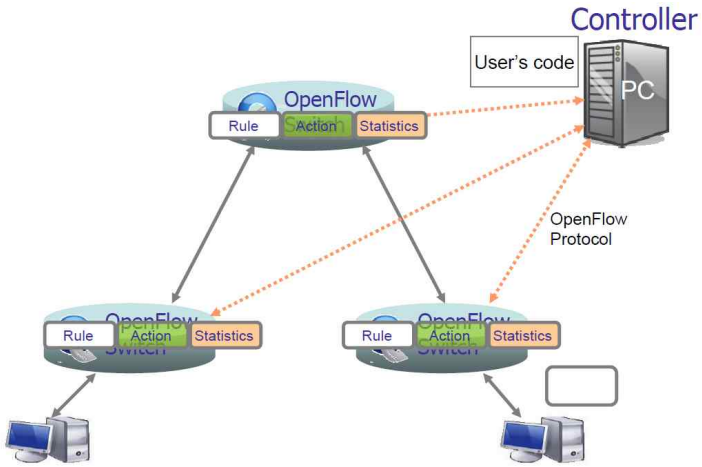


Figure18 OpenFlow behavior

3-2. OpenFlow Switch와 NOX 환경 구축

3-2-1. OpenFlow Switch install 과정 소개

OpenFlow Switch 소스 파일은 공개되어 있는 상태이고, linux 서버에 설치하여 동작 과정을 테스트 해 볼 수 있다. 현재, Ubuntu, Debian, CentOS, Fedora 등의 버전을 지원하고 있으며, 본 문서에서는 Ubuntu를 서버에 설치하는 과정을 소개한다.

- Ubuntu 설치

Ubuntu 8.10 Server ISO를 다운로드 받아 서버에 설치한다.

- OpenFlow 설치

Stanford Git repository로부터 OpenFlow를 설치한다. (참고, OpenFlow Switch Specification ver 0.8.9)

```
sudo apt-get install git-core automake m4 pkg-config libtool
git clone http://OpenFlowswitch.org/OpenFlow
cd OpenFlow
./boot.sh
```

OpenFlow를 컴파일하고 커널 모듈들을 build하기 위해 필요한 패키지들을 다운로드한다.

```
sudo apt-get install gcc linux-headers-`uname -r`
```

OpenFlow user-space와 kernel-space 스위치를 build한다.

```
./configure --with-l26=/lib/modules/`uname -r`/build
make
sudo make install
```

- Wireshark Dissector 설치

OpenFlow Wireshark Dissector는 OpenFlow 배포판에 포함된다. 설치를 위해서는 Wireshark 패키지 외에 추가적인 라이브러리를 요구한다.

Wireshark과 glib의 설치:

```
sudo apt-get install wireshark libgtk2.0-dev
```

Make 및 설치:

```
cd utilities/wireshark_dissectors/OpenFlow
make
sudo make install
```

- Regression Suite 설치

OpenFlow Regression Suit 또한 OpenFlow 배포판에 포함된다. 설치를 위해서는 몇 가지 추가적인 패키지들과 환경 변수들의 설정이 필요하다.

Test suite을 위해 필요한 패키지 설치:

```
sudo apt-get install liberror-perl libio-interface-perl liblist-moreutils-perl libpcap0.8-dev iproute psmisc libnet-rawip-perl

wget http://www.cpan.org/authors/id/J/JV/JV/Getopt-Long-2.38.tar.gz
tar xvf Getopt-Long-2.38.tar.gz
cd Getopt-Long-2.38
perl Makefile.PL
make all test
sudo make install

cd regress
sudo scripts/install_perlmods_apt.pl
```

Avahi-daemon(테스트 도중 메시지들을 보냄으로써 실패된 테스트 야기)의 비활성화:

```
sudo apt-get remove avahi-daemon
 or
sudo apt-get install sysv-rc-conf
sudo sysv-rc-conf avahi-daemon off
```

IPv6(테스트 도중 메시지들을 보냄으로써 실패된 테스트 야기)의 비활성화:
stsctl.conf의 업데이트 실시
% sudo vi /etc/sysctl.conf

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

blacklist.conf의 업데이트 실시
% sudo vi /etc/modprobe.d/blacklist.conf

```
blacklist net-pf-10
blacklist ipv6
```

시스템의 재부팅을 실시
% sudo shutdown -r now

재부팅 된 후, 추가적인 setup 시를 위해, OF_ROOT 환경 변수를 업데이트
```
cd ~/
cp <OpenFlow-dir>/regress/scripts/env_vars .
vim env_vars
```

- 설치의 검증

기본적인 설치 및 설정이 끝난 후, OpenFlow 스위치 내에서 설치 유효성을 검증한다. 우선, root로 로긴하여 환경 변수들을 로딩한다.

```
su
source ~/env_vars
```

가상 ethernet pairs(가상 인터페이스)의 로딩:
veth_setup.pl

Ethernet pairs의 검증(vethe0...7의 생성 유무):
/sbin/ifconfig | more

Wireshark의 실행:
wireshark &

Wireshark의 윗 부분 Filter 바에서 tcp 플래그 설정(단위 테스트의 분별을 위해):
of || tcp.flags.reset == 1

패킷의 기록 시작:
Capture->Interfaces로 가서 lo(loopback) 인터페이스를 선택하고, 아래와 같이 kernel-space 테스트를 시작한다.

of_kmod_veth_test.pl

Test suite이 동작하는 동안에 OpenFlow Switch와 테스트 용도로 포함된 Controller 사이에서 OpenFlow 패킷이 캡쳐되는 것을 Wireshark을 통해 볼 수 있다. TCP reset 패킷은 각 단위 테스트들을 분리해 주는 역할을 한다. Wireshark에서 각 패킷의 OpenFlow 부분을 클릭할 수 있고, 각 필드를 살펴볼 수 있다. 경우에 따라 OpenFlow 패킷 메시지가 보이지 않을 경우가 있는데, 이러한 경우 실제 패킷의 Payload부분에서 0x97로 시작하는 부분을 한 번 수동으로 디코딩하면 원하는 OpenFlow 필드들을 살펴볼 수 있다.

3-2-2. NOX install 과정 소개

NOX 소스 파일은 공개되어 있는 상태이고, linux 서버에 설치하여 동작 과정을 테스트 해 볼 수 있다. NOX는 다양한 리눅스 버전들을 제공하지만, Debian's Lenny 배포판에 가장 최적화되어 있고, 본 문서에서는 Debian's Lenny 배포판(ver 5.0.1)을 기준으로 설치 과정을 소개한다.

NOX는 다음의 소프트웨어 패키지에 의존적이다. Debian에서는 apt 패키지 형태로 모두 이용 가능하다.

g++ 4.2 or greater
Boost C++ libraries, v1.34.1 or greater (http://www.boost.org)
Xerces C++ parser, v2.7.0 or greater (http://xerces.apache.org/xerces-c)

Twisted Python 지원을 위해 다음의 추가적인 패키지들이 필요하다.

SWIG v1.3.0 or greater (http://www.swig.org)
Python2.5 or greater (http://python.org)
Twisted Python (http://twistedmatrix.com)

사용자 인터페이스를 위해 다음의 패키지들이 필요하다.

Mako Templates (http://www.makotemplates.org/)
Simple JSON (http://www.undefined.org/python/)
NOX에서 제공하는 모든 어플리케이션을 build하기 위해서 다음의 추가적인 패키지들이 필요하다.

apt-get install libsqlite3-dev python-simplejson

위의 과정들을 통해 리눅스 설치 및 NOX에서 요구되는 패키지들을 설치하였다. 이후에 진행되는 NOX(ver 0.5.0)의 설치 과정들을 아래와 같이 단계별로 실행한다.

```
git clone git://noxrepo.org/noxcore
cd noxcore/
./boot.sh
mkdir build/
cd build/
../configure --with-python=yes
make
make check
```

3-2-3. 기본 테스트

다음 그림에서는 OpenFlow Switch들과 NOX, 종단 호스트로 구성된 OpenFlow 테스트베드의 구성도를 보여준다. OpenFlow 버전은 0.8.9를 이용하였고, Controller는 0.5.0 core를 이용하였다. 스위치, 종단 호스트, Controller 모두 10.x.x.x 네트워크에 함께 연결되었다.

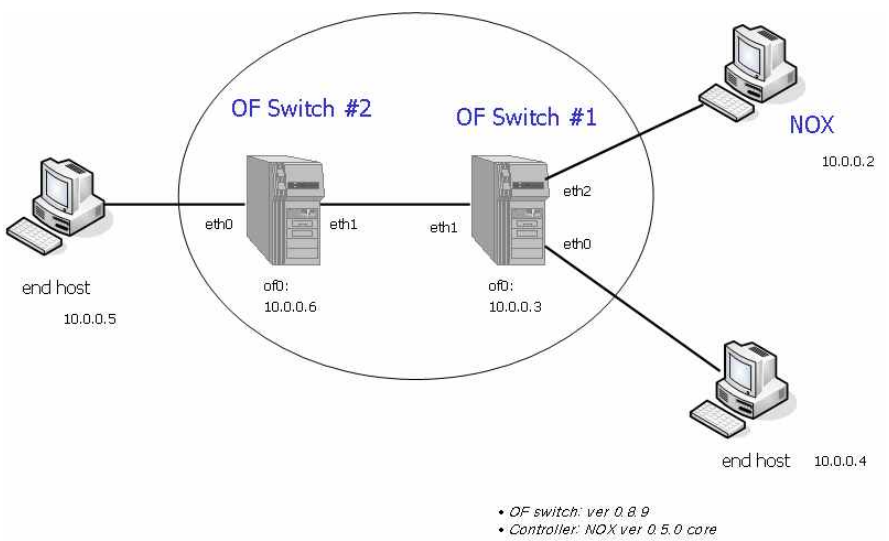


Figure19 OpenFlow Switch and NOX Testbed

3-2-4. 망 설정 단계

위의 테스트베드를 완성하기 위해서는 OpenFlow 스위치 2대와 Controller에 각각 다음과 같은 설정들이 요구된다.

OpenFlow Switch #1에서는 다음과 같은 설정들이 필요하다.

OpenFlow 커널 모듈을 로딩하기 위해:
sudo insmod datapath/linux-2.6/OpenFlow_mod.ko

Id가 0인 OpenFlow Switch를 생성하기 위해:
sudo utilities/dpctl adddp nl:0

OpenFlow Switch상에 존재하는 인터페이스를 더하기 위해:
sudo utilities/dpctl addif nl:0 eth0
sudo utilities/dpctl addif nl:0 eth1
sudo utilities/dpctl addif nl:0 eth2

OpenFlow Switch에서 필요한 일종의 관리 ip를 생성하기 위해:
sudo ifconfig of0 10.0.0.3 netmask 255.255.255.0

Datapath와 Controller인 NOX와 연결하기 위해 switch상의 Secure channel를 구동:
secchan/secchan nl:0 tcp:10.0.0.2:2525

OpenFlow Switch #2에서는 switch #1에서와 유사하게 다음과 같은 설정들이 필요하다.

sudo insmod datapath/linux-2.6/OpenFlow_mod.ko
sudo utilities/dpctl adddp nl:1
sudo utilities/dpctl addif nl:1 eth0
sudo utilities/dpctl addif nl:1 eth1
sudo ifconfig of0 10.0.0.6 netmask 255.255.255.0

secchan/secchan nl:1 tcp:10.0.0.2:2525

OpenFlow Switch에서는 아래와 같은 유용한 명령어들을 제공한다.

utilities/dpctl show nl:0 // nl:0 datapath를 보여준다.
utilities/dpctl dump-flows nl:0 // nl:0 상에 생성된 flow들의 목록들을 보여준다.
utilities/dpctl delif nl:0 eth0 // nl:0 상의 eth0 인터페이스를 삭제한다.
utilities/dpctl deldp nl:0 // nl:0 datapath를 삭제한다.

아래 그림은 switch #1에서 설정된 nl:0 datapath를 “utilities/dpctl show nl:0”를 캡쳐하여 보여준다. 각각 등록된 인터페이스들의 목록과 속성들을 차례로 보여준다.

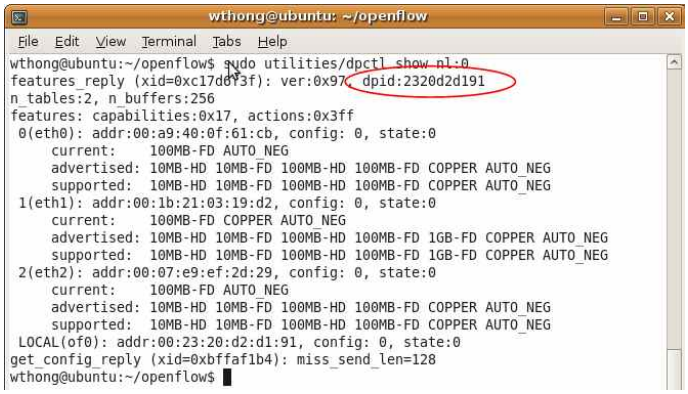


Figure20 Configuration of OpenFlow Switch

OpenFlow Switch 설정에 비해, NOX는 아래와 같이 서버 프로세스를 띄우는 명령어로 설정이 끝난다. 여기서 "pyswitch", "packetdump"는 NOX에서 구동시킬 수 있는 컴포넌트들의 예가 된다.

./nox_core -i ptcp:2525 pyswitch packetdump

다음 그림은 NOX 측의 서버 측 캡쳐 화면과 switch #1에서 Secure channel을 통해 연결된 결과 화면을 보여준다. 위에서 기 생성된 datapath id가 동일하게 올라오는 것을 확인할 수 있다.

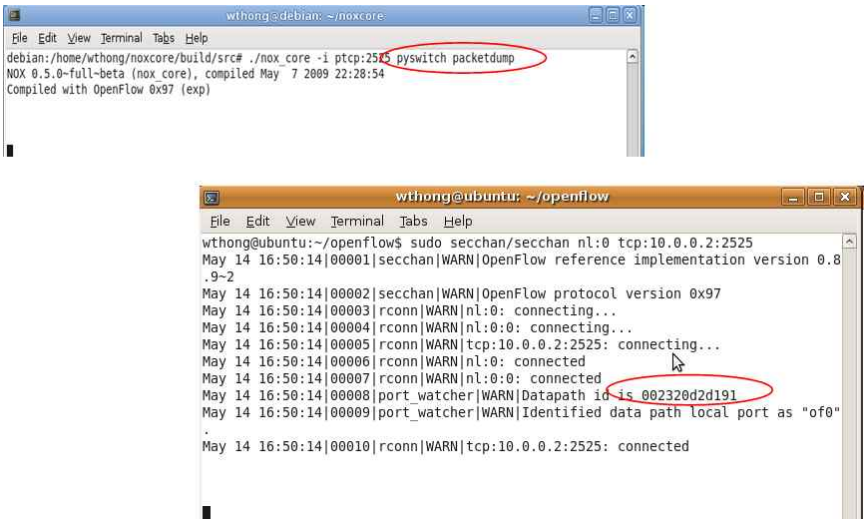


Figure21 Connection of OpenFlow Switch and NOX

- Wireshock을 통한 동작 과정 소개

위에서의 기본적인 테스트베드 설정이 끝난 후, 종단 호스트들 간의 ping 테스트를 수행하면서 OpenFlow Switch #1과 #2에서 생성되는 flow 목록들을 Wire shock의 패킷 캡쳐를 통해 확인할 수 있다. 다음 그림에서는 ping의 결과와 OpenFlow 프로토콜 메시지들 중에

서 "Flow mod"메시지를 캡쳐하여 OpenFlow Switch 상에 추가되는 목록에서 각각의 필드를 확인할 수 있다.

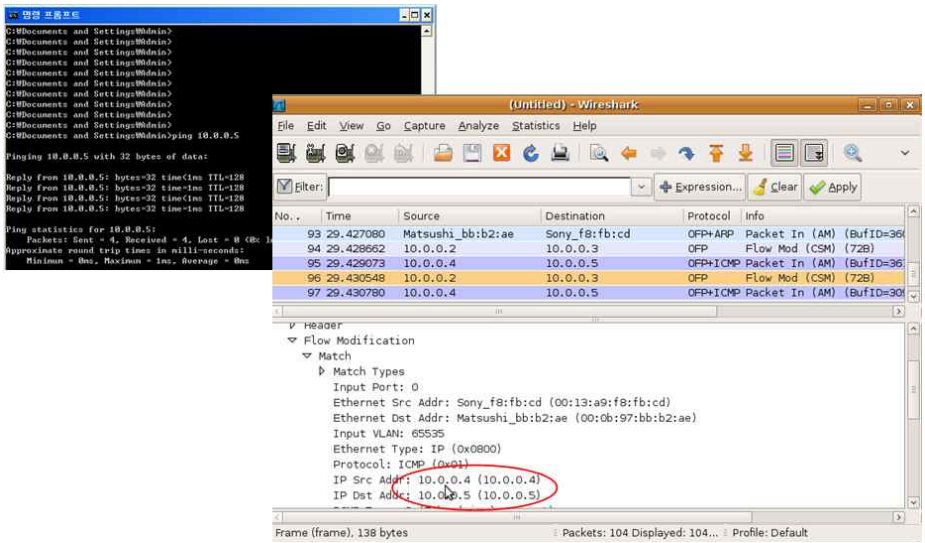


Figure22 Ping test for each hosts

아래 그림은 OpenFlow Switch #1에서 flow 테이블로 유지하고 있는 flow들의 목록들을 덤프하여 보여준다. 붉은 색 박스 부분에서 볼 수 있듯이, ping 테스트를 수행한 직후에는 OpenFlow Switch #1에서 각각의 종단(ip 주소: 10.0.0.4, 10.0.0.5)으로 ping을 전송하기 위한 icmp 패킷들이 짧은 시간 동안(duration=17s) 유지되는 것을 확인할 수 있다.

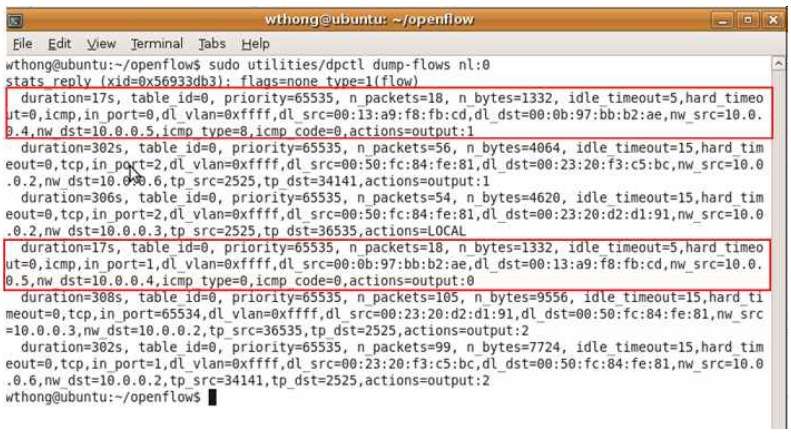


Figure23 OpenFlow Switch flows

3-2-5. 예제 및 시연

- VLAN Tagging

OpenFlow는 기존 망에서 VLAN이 제공하는 것처럼 사용자들에게 분리된 망을 제공할 수 있다. OpenFlow Switch들로 구성된 망에서의 VLAN tagging을 제공하는 Controller는 다음 그림처럼 나타낼 수 있다. 기본적으로 OpenFlow 스위치들의 역할을 Ingress, Egress,

by-pass 등의 역할로 나누어, 각 OpenFlow Switch상에서 유지하고 있는 flow 테이블내의 목록들을 기반으로 vlan tagging관련 목록의 경우 action을 설정하면 된다. 예를 들어, Ingress switch는 약속된 vlan id를 설정하는 action을 정의하고, by-pass switch에서는 해당 vlan id의 경우 특정 포트로의 forwarding을 수행한다. 그리고, Egress switch에서는 해당 vlan id를 떼어내는 action을 정의하면 VLAN 기반의 가상화된 망의 제공이 가능하다.

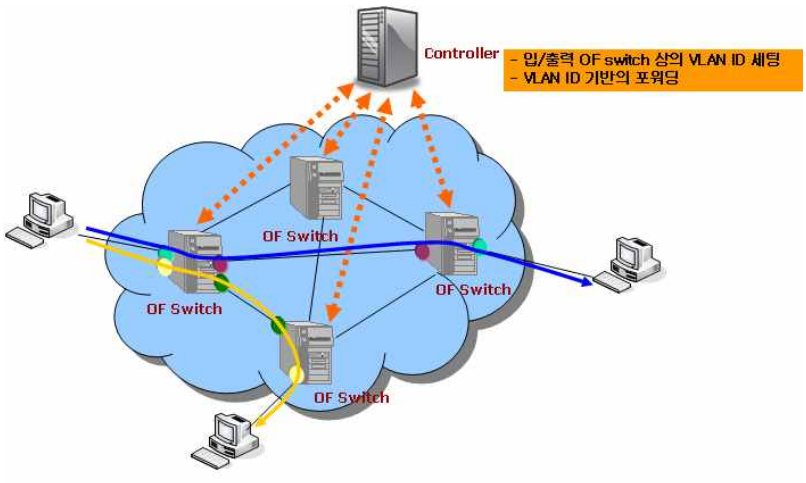


Figure24 Sample of VLAN tagging Controller

위에서의 시나리오는 아래 그림에서와 같은 사용자 기반의 vlan tagging을 통해 확장이 가능하다. 미리 설정된 user-to-Vlan 매핑 테이블 등의 사전 약속을 통해 동적으로 테이블을 유지하면서 특정 사용자로 부터의 패킷의 경우에는 vlan tag를 더하고, 삭제하는 action을 flow 목록상에 반영하여 유지한다. 이러한 Flow 목록들은 NOX를 통해 설정될 수 있도록 컴포넌트를 작성하여 반영할 수 있을 것이다.

```
# On user authentication, statically setup VLAN tagging
# rules at the user's first hop switch
def setup_user_vlan(dp, user, port, host):
    vlanid = user_to_vlan_function(user)
    # For packets from the user, add a VLAN tag
    attr_out[IN_PORT] = port
    attr_out[DL_SRC] = nox.reverse_resolve(host).mac
    action_out = [(nox.OUTPUT, (0, nox.FLOOD)),
                  (nox.ADD_VLAN, (vlanid))]
    install_datapath_flow(dp, attr_out, action_out)
    # For packets to the user with the VLAN tag, remove it
    attr_in[DL_DST] = nox.reverse_resolve(host).mac
    attr_in[DL_VLAN] = vlanid
    action_in = [(nox.OUTPUT, (0, nox.FLOOD)),
                 (nox.DEL_VLAN)]
    install_datapath_flow(dp, attr_in, action_in)
nox.register_for_user_authentication(setup_user_vlan)
```

Figure25 VLAN tagging for NOX component

- Congestion control

아래의 그림은 OpenFlow 네트워크를 통해서 Congestion control에 대한 예이다. Controller를 통해서 OpenFlow Switch A상에 트래픽의 증가로 지연시간이 길어질 때,

A-B 경로상의 OpenFlow Switch의 제어를 통해 경로상의 대역폭 확대 또는 QoS등의 기술 등의 적용으로 Congestion Control 기능을 수행 할 수 있다.

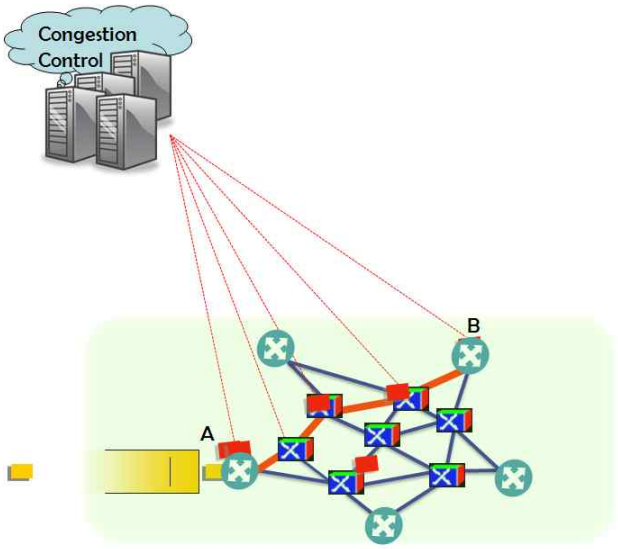


Figure26 Congestion control

- Traffic Engineering

OpenFlow Switch망 기반의 네트워크 환경에서 Traffic engineering 개념의 적용이 가능하다. 아래의 그림을 통해서 볼 때, 기본 라우팅 경로는 A-B-C로 경로가 정해져 있다고 가정하면, Controller를 통해 새로운 경로 A-B로 설정하여 라우팅을 할 수 있다.

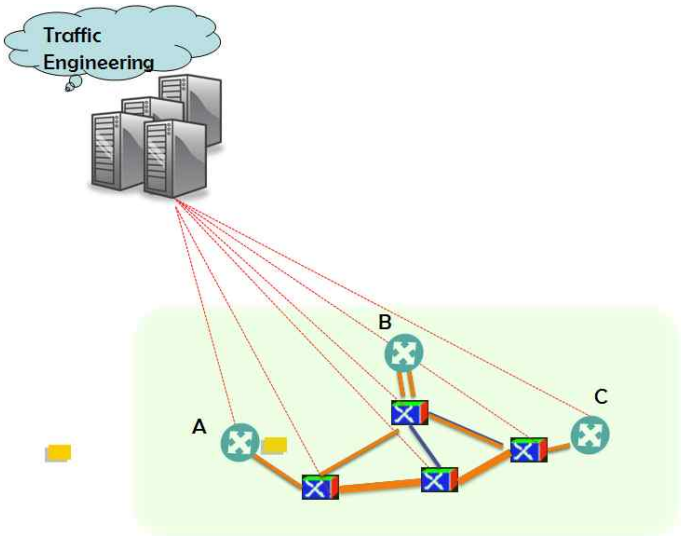


Figure27 Traffic Engineering

- 기타

최근, OpenFlow 기반의 다양한 서브 프로젝트들이 진행되고 있고, ACM Sigcomm2009 데모 세션에서는 관련 발표들이 있었다. 그들 중에 “Plug-n-Serve"는 기존의 네트워크 환경과는 달리 플로우 기반의 "Unstructured" 네트워크 환경에서 혼잡 문제를 해결하기 위한 접근 방법을 제안하였다. 기존 "Structured" 네트워크 환경에서 적용되는 Oblivious 방법(즉, 참여한 서버 상에 균일하게 request를 분배), Stateful 방법(least-loaded 서버에 request를 전송)하는 방법과 차별된 LOBUS(LOad-Balancing over UnStructured networks) 기법을 제안하였다. 그리고, 다음 그림에서와 같은 플로우 기반의 네트워크 환경에서 관련 GUI를 통해 로드 밸런싱을 시연하였다. 향후, 플로우 기반의 네트워크 환경을 적용할 때, 기존 Packet 기반의 네트워크 환경과는 차별화된 로드 밸런싱 기법을 감안하여야 하며, 관리 대상인 CPU 및 네트워크 상태를 모니터링하여 중앙제어기에 해당되는 플로우 매니저를 통해 동적인 반영을 할 수 있도록 하였다.

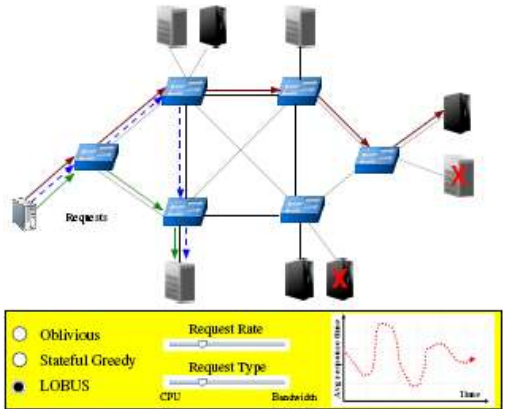


Figure28 Web Traffic load balacing based on OpenFlow network

3-2-6. OpenFlow enabled 제품 소개 및 벤더 동향

주요 네트워크 장비 벤더들은 OpenFlow enabled된 스위치들을 개발 중이거나 개발을 계획하고 있다. Juniper는 Junos SDK에 OpenFlow protocol을 탑재하고, MX-시리즈들을 대상으로 OpenFlow 기술을 반영하고 있다. HP에서는 ProCurve 5400-시리즈들을 대상으로 진행하고 있고, NEC에서는 IP8800 시리즈 장비들을 대상으로 관련 일들을 진행하고 있다. 그 외에, Cisco에서도 Catalyst 6509 상에서 OpenFlow 기술의 수용을 도입 중이다. 최근에는 주요 벤더들 외에, Toroki(http://www.toroki.com/) 라는 업체에서 OpenFlow 기술이 제공되는 LightSwitch 4810을 발표하였다.

Figure29 OpenFlow-enabled Switches

결론

세계적으로 미래인터넷의 기술 및 표준 선점을 위해서 미국, 유럽, 일본에서는 독자적으로 미래인터넷을 위한 거대 규모의 프로젝트들이 진행 중에 있다. 최근 들어 그린 IT와의 연계성이 강조되면서 그린 네트워크로 전환의 시기에 있다.

그린 네트워크로의 전환은 '고성능 네트워크를 활용하고 방송통신 자원을 효율적으로 사용하게 함으로써 타 분야의 CO_2 감축을 추구한다'는 뜻이다. 유럽통신네트워크운영자협회(ETNO)는 이미 2006년 재택근무, 영상회의 등 고성능 네트워크를 활용한 방송통신서비스 활성화로 에너지 절감 및 CO_2 감축이 가능하다고 전망했다. EU 근로자의 10%가 재택근무를 하면 연간 무려 2217만톤의 CO_2 를 감축할 수 있다고 보고되고 있다.

그린IT 동향분석 레포트에 따르면 미국, 특히 통신 영역에서 이 같은 고성능 네트워크를 구축하려는 시도는 GENI, FIND 등 미래인터넷(future internet) 관련 프로젝트로 구체화하고 있다. 특히 단순히 현재 인터넷을 개선하는 차원을 넘어 관련 신규 장비시장을 개척하고 표준화를 선점하여 인터넷 종주국의 입지를 강화하는 목적을 갖고 있다.

이러한 미래인터넷에 대한 세계적인 동향 가운데 국가연구망인 KREONET의 미래인터넷으로의 진화는 IT 분야 국가 경쟁력제고와 신규시장 개척, IT 강국으로서의 입지를 지속적으로 이어나갈 수 있는 기회이다.

따라서 미래인터넷을 위한 실제적 테스트베로서의 네트워크 가상화 기술 연구와 환경 구축은 KREONET의 효과적인 진화 방안일 뿐만아니라 미래인터넷을 준비하는 기회이다.

네트워크 가상화 기술 중 특별이 OpenFlow 기반의 네트워크 구성을 통해서 네트워크 가상화 환경 구성의 장점으로서는

1) Open control의 개념으로 실제 사용자에 실제 네트워크 자원들에 대한 제어 권한을 부여함으로써 획기적인 테스트환경을 제공할 수 있다.
2) KREONET 백본을 운영하는 KISTI의 입장에서 기존 가상화 솔루션들이 Virtual Slice들을 제공함으로써 자원의 한계성이 있는데 반해 실제 백본 기반의 네트워크 자원의 제공이 가능함으로써 보다 실제에 가까운 가상환경의 제공이 가능하다.
3) Controller의 효과적인 운영 및 구성으로 사용자 뿐만 아니라 네트워크 운영자 입장에서도 네트워크 및 자원의 효과적인 관리가 가능하다.
4) 일반 트래픽과 실험 트래픽의 확실한 분리로 상호간 트래픽에 영향을 주지 않는 가운데 실제 환경에서의 실험이 가능함으로 실험 검증의 수준이 높아진다.

이와 같이 OpenFlow기반의 가상화 기법을 활용하여 기존망의 형태를 유지하면서 효과적인 미래인터넷 테스트를 위한 네트워크 가상화 환경을 구축 제공할 수 있다.

OpenFlow 기반의 네트워크 가상화 테스트베드의 1차적인 사용자로서는 네트워크 연구자들이며, 기존 NS-2, Emulab, Simulation 등의 환경에서의 실험보다 OpenFlow 기반 하에서 실제 네트워크 환경하에서 독립적인 실험을 할 수 있는 장점이 있다. 네트워크 연구자 뿐만 아니라 독립적인 네트워크의 요구가 있는 분야의 경우 OpenFlow 기반의 독립망을 구성하여 제공할 수 있으며, 네트워크의 추가 증설 비용 및 관리 비용을 줄일 수 있는 경제적인 효과가 있다.

참고문헌

[1] Larry Peterson, John Wroclawski, "Overview of the GENI architecture", GDD-06-11, January 5, 2007.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner. OpenFlow: Enabling innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69-74, April 2008.

[3] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards and operating system for networks. In *ACM SIGCOMM Computer Communication Review*, July 2008

[4] N.M. Mosharaf Kabir Chowdhury, R. Boutaba. A Survey of Network Virtualization, Waterloo Univ. Technical Report: CS-2008-25, October 15, 2008

[5] OpenFlow Switch Consortium. http://www.openflowswitch.org/

[6] The OpenFlow Switch Specification version 0.8.9. December 2.2008.

[7] Global Environment for Network Innovations, Web site http://geni,org

[8] Federica Project, Web site http://www.fp7-federica.eu/

[9] Future Internet Forum, Web site http://fif.kr/

[10] Fire Works Project, Web site http://www.ict-fireworks.eu/

[11] AKARI Project, Web site http://akari-project.nict.go.jp/eng/index2.htm