ISBN 978-89-6211-471-3 98560

네트워크 가상화 서비스를 위한 요구사항서

일자: 2009년 11월 30일
부서: 슈퍼컴퓨팅본부 기반기술개발실
제출자: 문정훈
jhmoon@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

1. 서론
2. 관련 연구 동향 및 필요성
3. 국내외 네트워크 가상화 테스트베드
4. 가상화 테스트베드 사용자 및 요구사항
5. OpenFlow 기반 서비스프레임워크 설계 요구
6. 결론
7. 참고문헌

1. 서론

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

현재의 네트워크 환경에서는 연구자가 실제 대규모 네트워크의 운용과 여러 가지 실험을 할 수 있어야 되지만 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 사용자에게 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(Virtual LAN Internetwork), VPN(Virtual Private Network), VLAN(Virtual LAN) 등이 있다. 물리적으로 서로 다른 하드웨어들을 하나의 공통된 형태의 가상 장치들로 만들거나 하나의 물리적 하드웨어 상에 여러 개의 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.

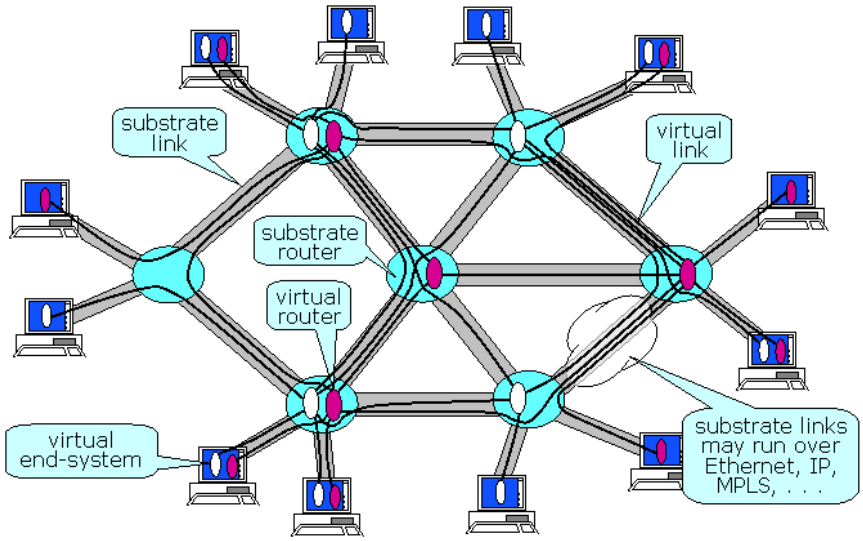

Figure 1 Network Virtualization

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

이러한 배경 속에서 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 리엔지니어링 시 실제로 어떻게 작동하는가를 연구하는 Clean-Slate 계획의 일부분이며, OpenFlow기술은 스위치 상의 Flow table, Controller, OpenFlow protocol의 3개의 주요 부분으로 구성되어 있다.

2. 관련 연구 동향 및 필요성

미래인터넷 및 네트워크 가상화의 대표적인 연구 동향으로 미국의 GENI, 유럽의 FIRE, 일본의 AKARI, 한국의 FIF 등을 살펴본다.

2.1 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 수행중에 있다. 1단계로서 개별 아키텍쳐 컴포넌트들에 대한 다양한 연구를 시작으로 2단계에서는 1단계의 아키텍쳐 컴포넌트들이 통합된 전체 아키텍쳐 연구로 진행되고, 3단계로서 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 아키텍처들을 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,E 구현하고 완성도에 따라서 최선을 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)

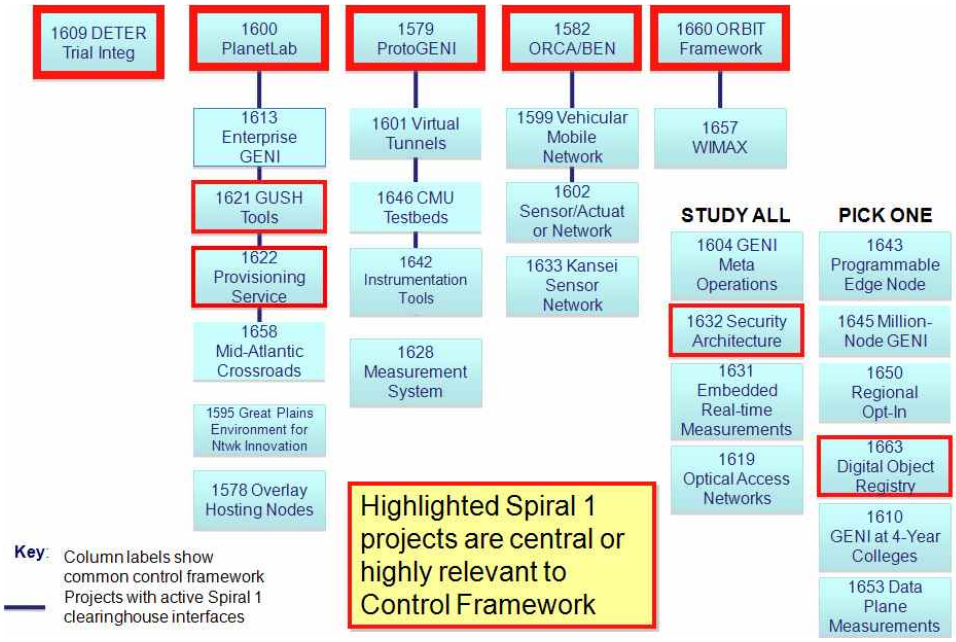

Figure 2 GENI Spiral 1 Integration : 5 Control Framework Clusters

GENI에서 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 아키텍처는 세 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 아키텍처 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 대역폭등과 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 슬리버들이 모음이다. 사용자는 슬라이스상에서 실험을 수행한다.
애그리게이트는 GENI에서 서로 연관된 컴포넌트들을 모아 놓은 하나의 집합을 나타내는 GENI의 객체이다. 컴포넌트들의 집합을 관리하고 자원할당을 코디네이션 한다.

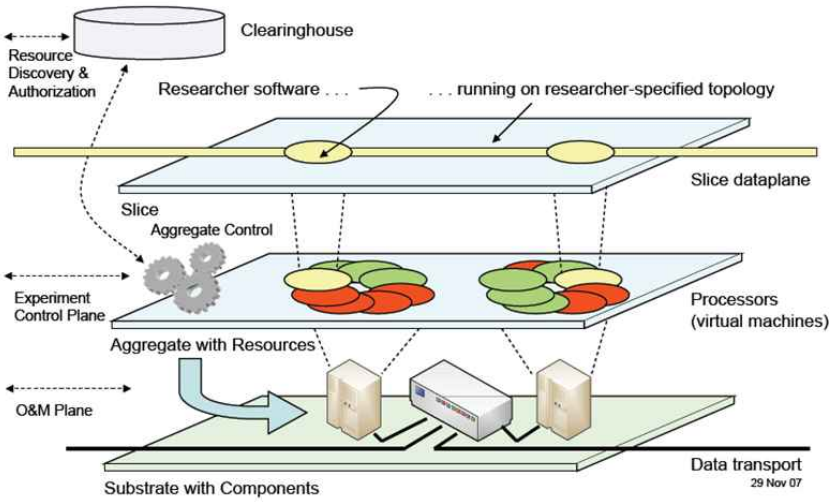

Figure 3 Substrate, Aggregates, Slices of GENI

2.2 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 아키텍쳐를 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.

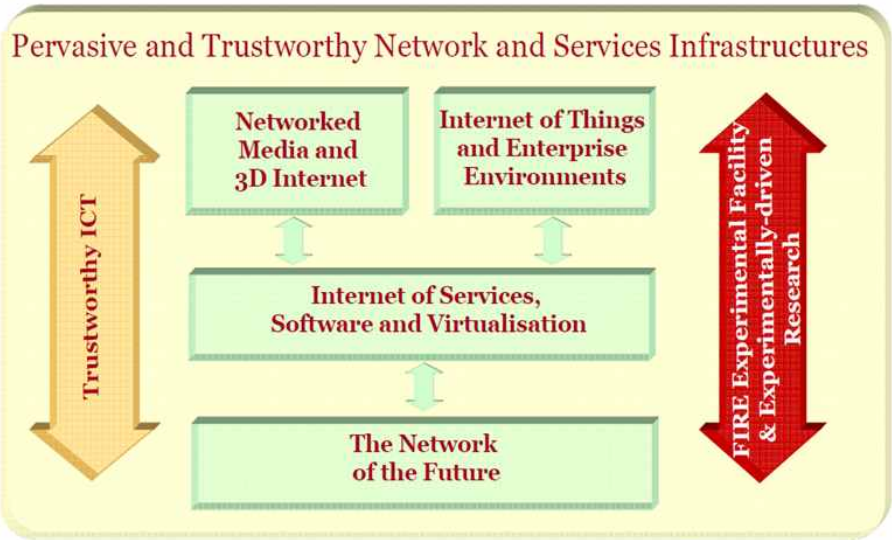

Figure 4 Future Internet of FP7 ICT by EU

- 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 기대한다.

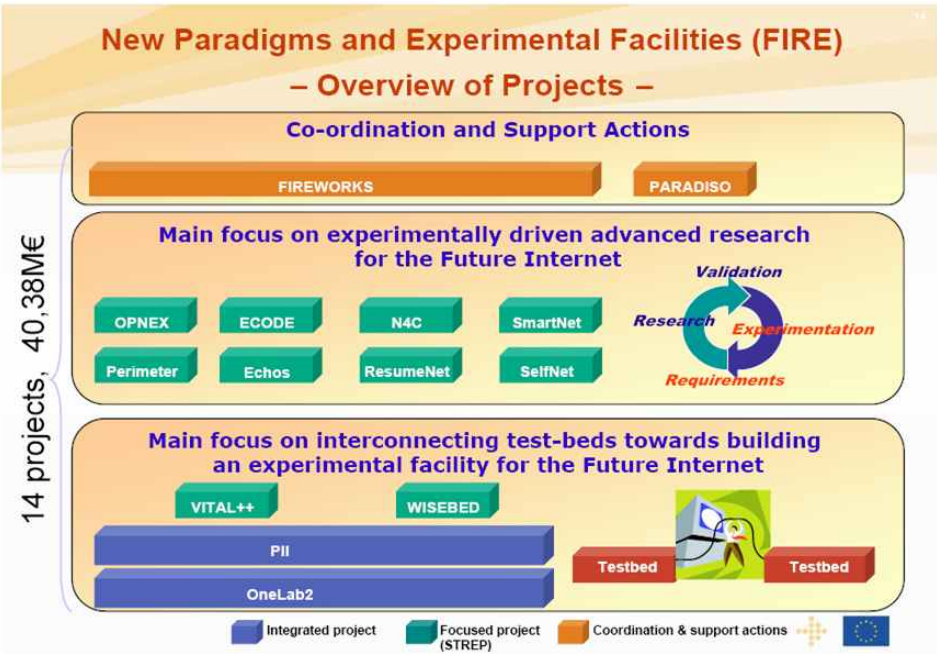

Figure 5 Overview of FIRE Projects

2.3 일본

일본의 경우 미래인터넷에 대한 연구는 현재 인터넷에 기반한 차세대 네트워크에 대한 연구(ITU-T 기반)가 진행되고 있다. 이것은 현재의 인터넷에 기반한 것인데 NGN에서 NXGN(Next Generation Network)로의 진화를 의미하고, 일본의 미래 네트워크는 신세대 네트워크로 구분하여서 2007년부터 NWGN (New Generation Network)에 대한 연구가 시작되었다. 일본의 미래인터넷의 특징은 NXGN과 NWGN의 동시 진행인데 NWGN의 경우 미래인터넷으로 구분되며, 새로운 패러다임 기반의 장기적 과제 형식이다, 국책 연구소 중심이 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

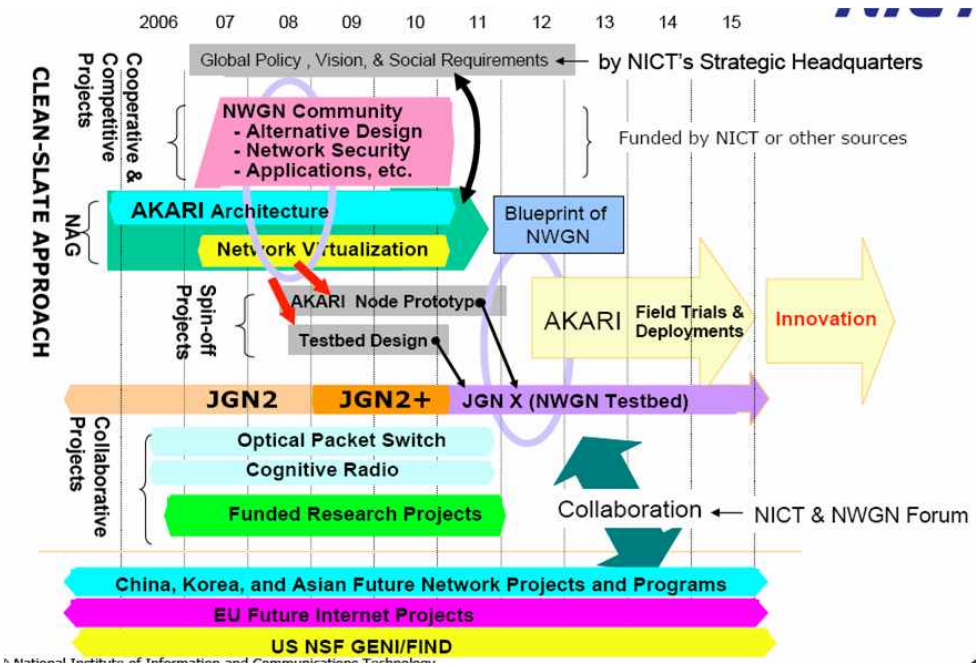

Figure 6 AKARI & NWGN R&D Plan

2.4 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 그

룹을 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, 무선기술연구 분야에서는 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.

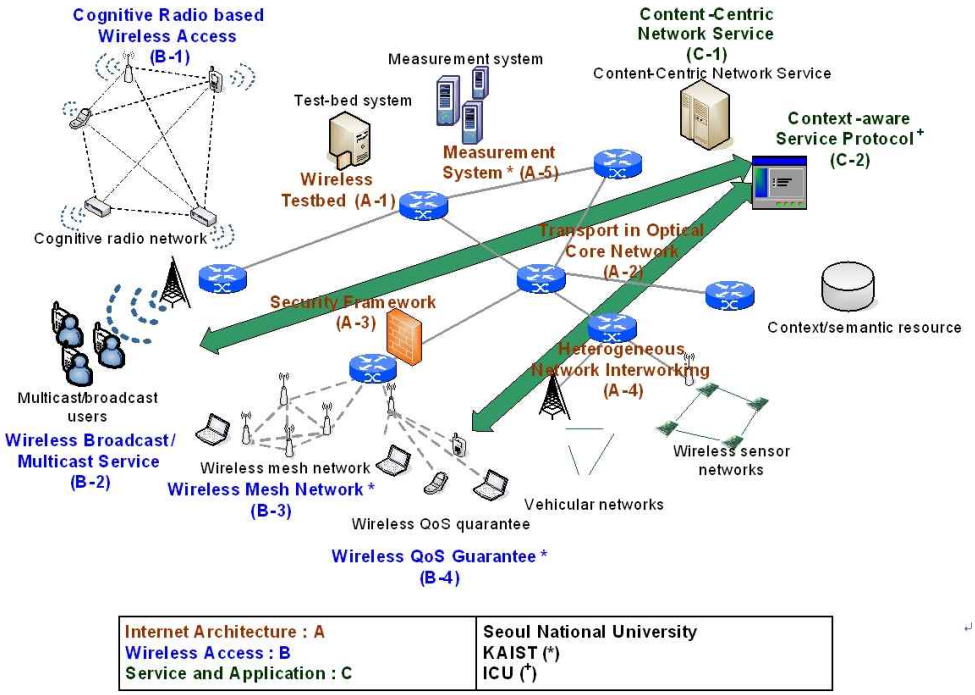

Figure7 Research topics and areas of FIF

2.5 가상화 기술 연구의 필요성

가상화 기술을 통해 네트워크 연구자들을 대상으로 기존 실험실 위주의 연구 환경을 실제 네트워크 환경으로 확대하고, 스위치, 라우터 등의 네트워크 장비에 대해서도 연구자의 환경으로 구성 또는 충분한 제어 권한을 통하여 효과적인 연구를 수행할 수 있게 한다. 로컬 환경의 한계를 벗어나 백본 환경에서의 실제적인 테스트를 통해 연구 성과 및 효과를 증대시킬 수 있게 된다. 따라서 이러한 획기적인 네트워크 테스트베드를 제공할 수 있는 네트워크 가상화 핵심 기술과 가상화 환경 하에서 효과적인 연구 수행을 위해 트래픽 모니터링,

가시화, 아카이브 등과 같은 연구자를 위한 서비스와 가상화 네트워크 관리자를 위한 서비스 기술의 개발이 필요하다.

3. 국내외 네트워크 가상화 테스트베드

전 세계적으로 미래인터넷에 대한 투자와 연구가 지속되고 있는 가운데 OpenFlow기반의 네트워크 가상화 연구와 프로젝트들은 유럽연합에서 추진 중인 FIRE 프로젝트의 일부인 FEDERICA 프로젝트에서 가상화 인프라로서 OpenFlow를 활용할 계획에 있으며, OpenFlow를 개발한 미국의 스탠포드 대학에서 다양한 솔루션과 데모들이 진행 중에 있다.

3.1 FEDERIA 프로젝트

2008년에 유럽연합 FP7의 FIRE 프로젝트의 일부로서 연구자들에게 e-Infrastructure를 가상화 기반으로 미래인터넷을 위한 네트워크/컴퓨터 연구자에게 기본적인 툴로 제공하는 프레임워크 프로그램으로서 유럽의 광대역 e-Infrastructure를 구축한다. 그리고 FEDERICA에서 사용되는 장치들은 모든 연구자들에게 할당된 'slice set'에 대해서 완전한 제어권을 제공하고, 분산 실험되는 모든 계층의 통신 제공을 목표로 한다. 다음의 그림은 FEDERICA의 e-Infrastructure의 기본 구성도이다.

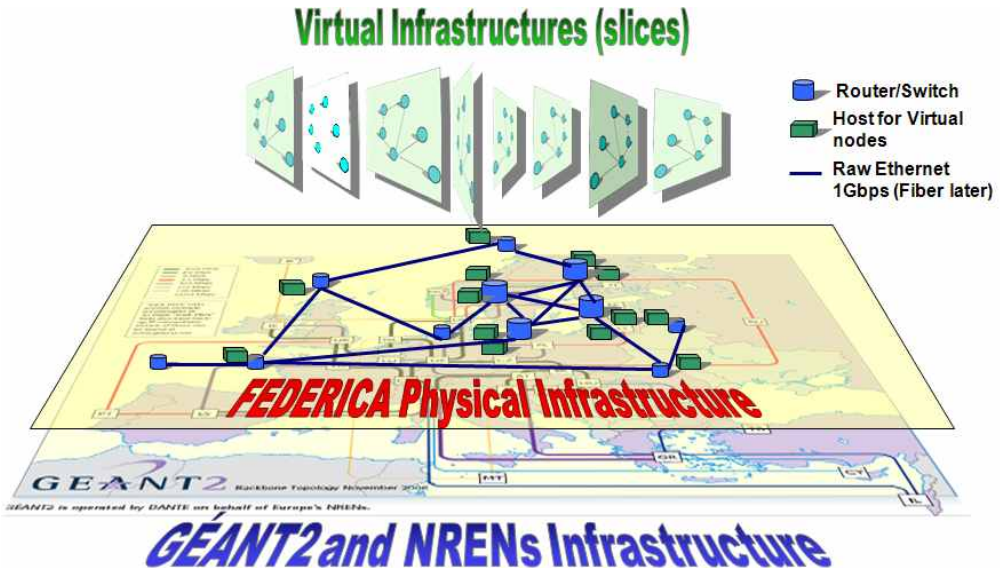

Figure 8 e-Infrastructure of FEDERICA

FEDERICA 프로젝트에서도 위의 그림과 같이 물리적인 네트워크인 GEANT2 & NREN 위에서 가상의 FEDERICA 인프라를 구축하고 있다. 가상화의 개념을 통해서 물리적인 장비와 네트워크들 'slice'해서 사용하고 있는데 다음과 같은 특징으로서 OpenFlow를 slicing의 한 방법으로 계획 중에 있다.

- General and Flexible
- Slicing on a 'per-flow' basis
- Flow-based substrate and slicing
- Management based on centralized control

FEDERICA의 향후 계획 다음과 같다.

- 기본적인 설계와 Key component를 중심으로 구현
- 리눅스 기반의 작은 규모의 환경에서 실험적 구축
- 큰 규모의 실험환경으로 업그레이드하면서 FEDERICA에 적용 및 OpenFlow-enabled 장비의 도입

따라서 KREONET 환경하에서도 FEDERICA의 접근방법과 같이 OpenFlow 환경의 기본적인 구축과 실험을 시작으로 대규모 테스트베드의 구축, 그리고 KREONET/GLORIAD로 적용하여 국내외 가상화 연구를 위한 테스트베드로 제공할 수 있다.

3.2 스탠포드 대학 중심의 FlowVisor 프로젝트

기존의 OpenFlow swtich환경에서 Controller의 성능은 한 번에 하나의 Controller만 작동하는 형태였다. 다수 사용자가 동시에 OpenFlow swtich환경의 네트워크를 사용할 수 있도록 개발 된 것이 FlowVisor이다. 또 FlowVisor는 NOX의 S/W 구조의 복잡성과 성능의 문제등으로 인해 일종의 Controller로서 개발되었다.

FlowVisor는 네트워크 자원들을 flow 기반으로 slicing해서 사용하는 것으로서 다수의 Controller가 동시에 작업을 할 수 있게 되었다. 또한 FlowVsior가 개발되면서 본격적으로 OpenFlow switch기반의 네트워크 환경도 flow 기반의 slicing을 통한 가상화 환경의 구현이 가능하게 되었다. 따라서 FlowVisor의 목적은 다수의 연구자가 동시에 OpenFlow switch환경에 접근하여 자신들의 환경에 따른 연구 및 실험을 가능케 한다. 아래의 그림은 FlowVisor의 기본 구조도이다.

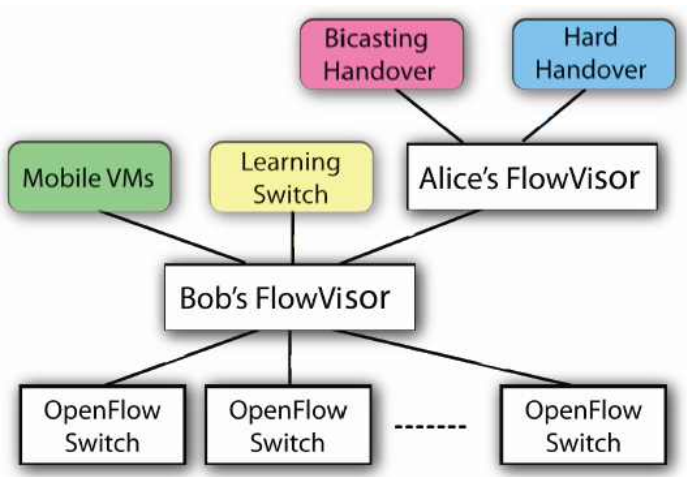

Figure 9 Multiple Controllers based on FlowVisor

위의 그림에서 Bob's FlowVisor를 통해 하단에 OpenFlow Switch 네트워크가 관리가 되고 Bob's FlowVisor는 Alice's FlowVisor, Mobile VM Controller, Learning Switch Controller를 지원하고 있다. Alice's FlowVisor는 Bicasting Handover Controller와 Hard Handover Controller를 지원하면서 Bob's FlowVisor를 통해 OpenFlow switch 네트워크를 활용하고 있다.

FlowVisor를 활용한 대표적인 기술들로는 Plug-n-Serve, OpenPipes, OpenRoads, Aggregation 등이 있다. 이러한 각각의 기술들은 OpenFlow switch 기반 하에서 하나의 Controller상에서 구현되는 것들로서 FlowVisor통하여 동시에 하나의 OpenFlow switch 네트워크상에서 작동이 가능하게 된다.

· Plug-n-Serve : Flow 기반 네트워크 상의 Load-balancing 기능
· OpenPipes : OpenFlow switch 기반으로 구성된 네트워크 상에서 H/W설계를 line rate으로 할 수 있는 기능 (경제적으로 우수)
· OpenRoad : WiFi-WiMAX사이에서 loss가 없는 Handover 기능
· Aggregation : 트래픽 관리를 위해 동적으로 flow를 모으는 기능

KREONET환경에서도 OpenFlow swtich환경을 FlowVisor를 통해 구현함으로써 네트워크 가상화 환경으로서 다수의 사용자에게 OpenFlow switch 환경의 테스트베드를 제공할 수 있다.

3.3 FIF에서의 테스트베드

한국의 미래인터넷포럼인 FIF에서는 다양한 분야에서 미래인터넷과 관련한 연구를 진행 중에 있다. 그중에서 OpenFlow관련 연구와 미래인터넷 플랫폼 개발 연구에 대한 진행은 다음과 같다.

- NetFPGA기반 OpenFlow switch 구성

NetFPGA 기반으로 OpenFlow Switch를 구성하여 KOREN 망에 테스트를 진행하였고, KOREN/KREONET의 연동을 위해 테스트를 진행 중에 있다. 이러한 경우 멀티 도메인 문제가 생기게 되는데 아직까지 Controller의 Federation문제에 대한 솔루션은 없는 상태이다. 따라서 멀티 도메인 환경에서의 테스트는 기본적으로 터널링 기술을 통하여 단일 도메인 환경으로 만들어서 테스트하게 된다. FIF에서의 테스트 과정에서는 터널링 기술로서 Mac-in-IP 기술을 통해 인접한 도메인으로 '브리징'을 하는 방식을 사용하고 있다. 다음은 Mac-in-IP를 사용하여 KOREN/KREONET 환경의 테스트베드 구축도이다.

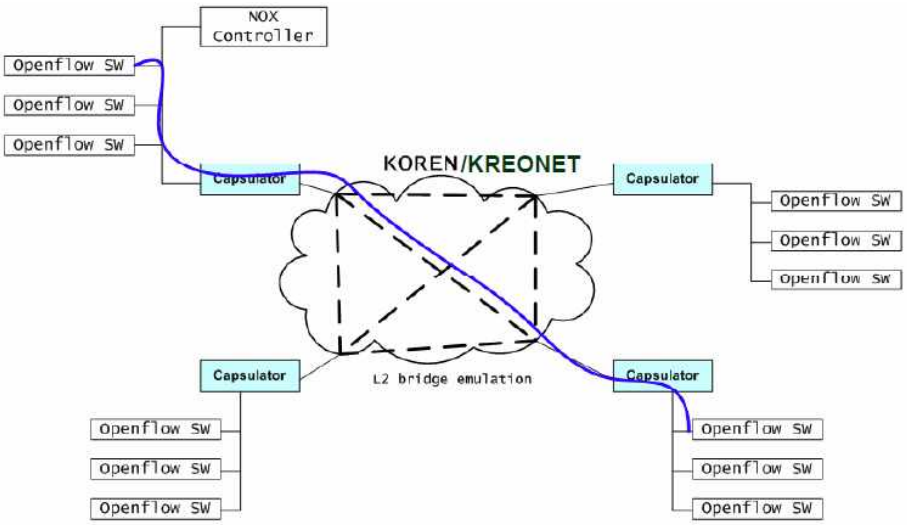

Figure 10 An Implementation of OpenFlow KOREN/KREONET testbed

- FIRST 프로젝트

ETRI에서 진행중인 프로젝트로서 Future Internet Research for Sustainable Testbed로서 Virtualized Programmable Platform의 개발이다. FIRST 프로젝트 개발의 주요 요구사항은 다음과 같다.

· Multiple/Heterogeneous/Simultaneous Experiments Support
· Scalability (# of Experiments)
· Threat-Zero
· Diverse Wireless Subnet Support
· GENI-compatible
· Domain Federation
· Inter-working with Today's Network (e.g., IPv6, BcN, IP-USN, etc. - ISP Peering)

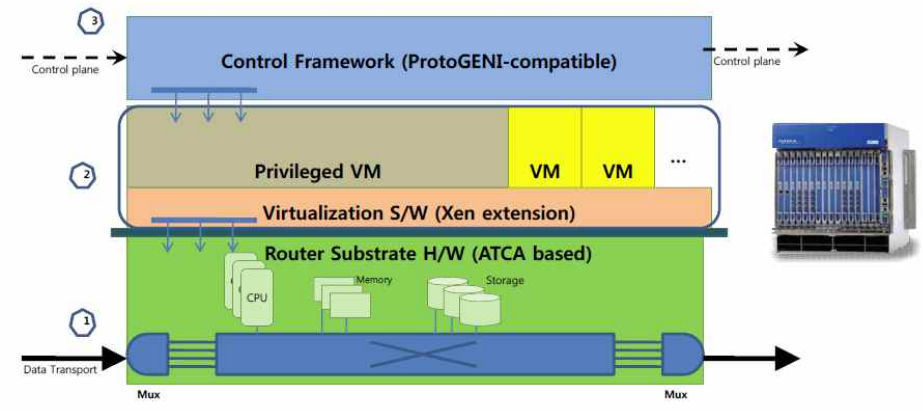

Figure 11 FIRST Platform

FIRST 프로젝트는 일종의 플랫폼 개발 프로젝트로서 미래인터넷의 다양한 환경과 변화에 대해서 기존 네트워크 환경의 대부분의 기술들과 함께 통합형으로 방향을 잡고 개발 중에 있다. 다음의 그림은 KREOENT/GLORIAD 지원을 통한 GENI-FIRST 연결도이다.

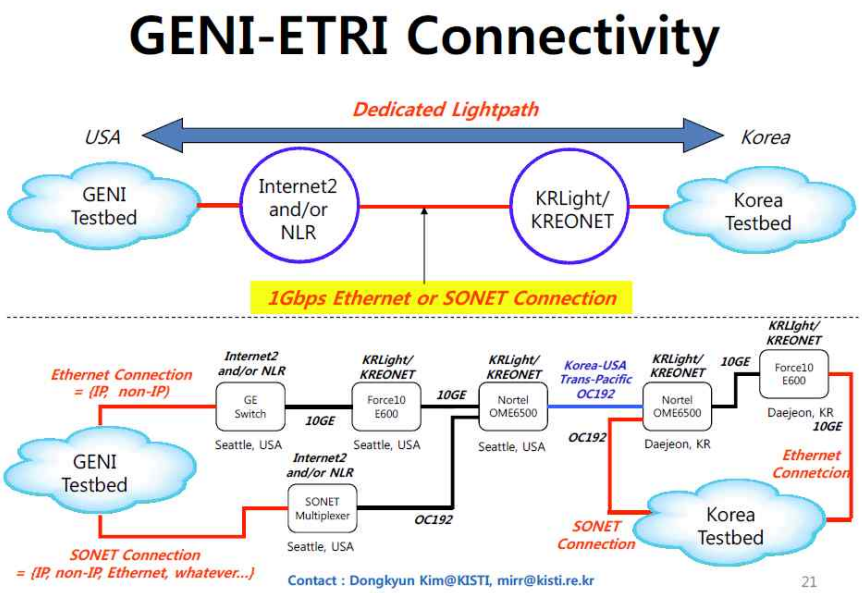

Figure 12 GENI-ETRI Connectivity

4. 가상화 테스트베드 사용자 및 요구사항

네트워크 가상화 기술의 사용자는 네트워크 연구자로서 실험망 환경이 아닌 실제 네트워크 상에서의 테스트베드에 대한 요구가 증가하고 있다. 이러한 요구는 새로운 네트워크 기술들이 실험실의 제한적인 환경하에서의 제약으로 인해 효과적인 기술임에도 불구하고 기존 망 환경에 적용이 되지 않을 뿐만 아니라 실제 환경에서의 검증도 되지 않고 있기 때문이다.

- 네트워크 가상화 환경 사용자 : 네트워크 연구자
  - 실제 네트워크 환경에서 연구가 필요한 네트워크 연구자
  - 실제 운영중인 네트워크 장비의 활용/제어가 필요한 네트워크 연구자
  - 실제 네트워크와 네트워크 장비의 임의 수정 및 재구성이 필요한 네트워크 연구자

따라서 실제 환경하에서의 실제 운영측면의 테스트는 기존 네트워크 환경의 높은 진입장벽으로 인해 지금까지는 시뮬레이션 또는 실험망 규모의 환경에서 테스트가 진행되었지만, 가상화 기술을 통하여 백본 네트워크 환경을 포함한 대륙간 네트워크 실험도 가능하게 되었다. 이러한 환경을 가능케 하는 기술들로서 지금까지 제안되고 진행되고 있는 프로젝트들이 위에서 언급한 PlanetLab, VINI 등과 같은 기술들이며, 특히 백본 네트워크에 대한 권한이 있는 KREONET 환경에서는 OpenFlow Switch환경이 효과적인 가상화 기술이다.

네트워크 연구자들의 실제 네트워크 환경에서의 테스트를 위해서 기존 네트워크에 가상화 기법의 접목이 필요하며, 가상화 기술과 함께 네트워크 연구자들의 가상화 환경 하에서 효과적인 연구 수행을 위해 트래픽 모니터링, 가시화, 아카이브 등과 같은 연구자를 위한 서비스와 가상화 네트워크 관리자를 위한 서비스 기술의 개발이 필요하다.

- 연구망 기반의 테스트베드 환경

KREONET은 현재 Optical+packet hybrid 형태의 네트워크 구성을 가지고 있으며, 14개 지역 기가 팝과 2개의 국제망을 가지고 있다. 또한 국제적으로 22개의 R&E 네트워크(NREN in Europe, US, Asia etc)와 Peer되어 있으며, 58개의 1/10G 링크로 HEP, Medical, Aerospace, Future Internet등을 지원하고 있다. 특히 대전의 대덕 연구단지를 중심으로 구성된 SuperSIReN은 대전 지역 첨단 과학기술 연구를 위한 연구소, 대학간의 10G 광네트워크로 구성된 백본이며, 또한 GRLORIAD를 통한 국제 링크가 10G 기반으로 운영되고 있다. GRLORIAD는 현재 11개국 (미국, 러시아, 한국, 캐나다, 네덜란드, 중국, 스칸디나비아 반도의 5개국)이 참여하고 있는 Global Ring Network for Advanced Applications Development로서 참여국들 간의 첨단 과학기술 연구 인프라로서 활용되고 있다. 아래의 그림은 KREONET과 GLORIAD 네트워크의 기본 개념도이다.

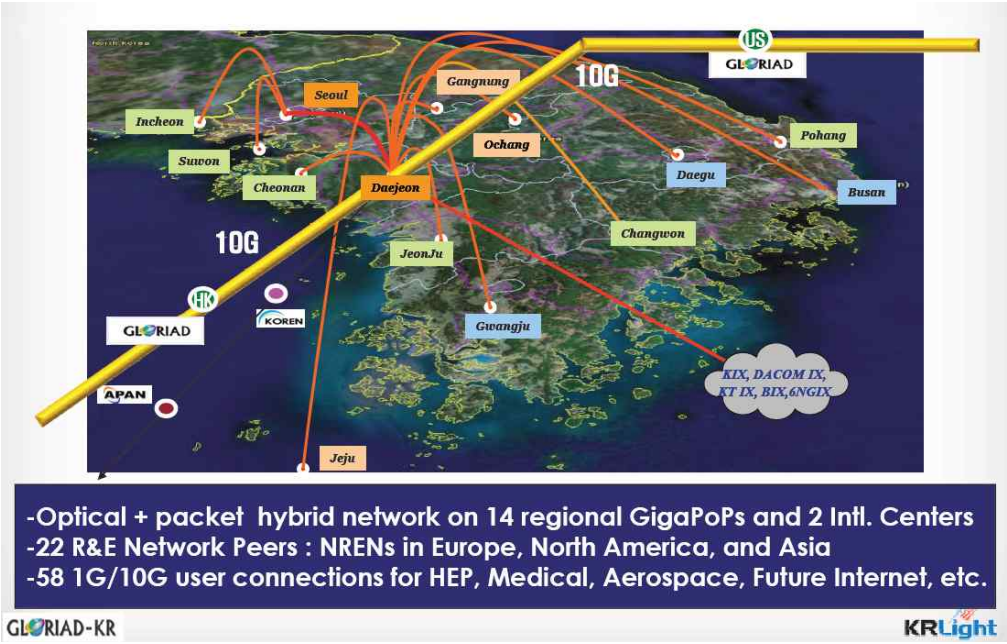

Figure 13 KREONET & GLORIAD

KREONET 기반의 네트워크 가상화 기술의 적용은 국내의 실제 백본 연구망을 활용할 수 있으며, 국내 주요 지역으로 확대 적용이 가능하다. 또한 국제망은 GLORIAD와 연계를 통해 미래인터넷연구가 활성화 되어 있는 미국, 유럽과의 협업연구가 가능하며, 가상화 기반의 네트워크 테스트베드의 제공이 국제적인 네트워크 환경 하에서 지원이 가능하다.

KREONET/GLORIAD 기반의 가상화 서비스의 로드맵을 다음과 같이 추진할 수 있다.

1) 로컬 환경에서의 OpenFlow 기반의 가상화 환경 구축
2) SuperSIReN기반의 실제 운영망에서의 가상화 환경 구축
3) 국내 백본에서의 가상화 환경 구축
4) 국제 망 기반의 가상화 환경 구축

ETRI에서 진행 중인 프로젝트로서 FIRST는 Future Internet Research for Sustainable Testbed로서 Virtualized Programmable Platform의 개발로서 KREONET/GLORIAD를 통해 미국의 GENI 테스트베드와 연계 되어있다. 이러한 환경 구성의 실제적인 사례를 통하여 미래인터넷을 위한 네트워크 가상화 서비스를 KREONET/GLROIAD 기반에서 지원을 할 수 있다.

5. OpenFlow 기반 서비스프레임워크 설계 요구

일반적으로 네트워크 가상화 기법은 접근 방법에 따라 그림에서처럼 분류할 수 있다. 각각은 “H/W programmable”, “S/W programmable”, “Virtualizing off-the-shelf nodes"에 해당된다. 각각의 접근 방법은 performance, cost, flexibility 측면에서 trade-off 관계에 있으며 각각의 일반적인 특성은 아래와 같다.

5.1 네트워크 가상화 기법 분류

- “H/W programmable” 방법

H/W 기반의 네트워크 가상화 접근 방법은 performance/flexibility 측면에서 우수성을 갖으나, cost 측면에서 나머지 접근 방법에 비해 경쟁력이 떨어진다. 이러한 접근 방법의 예로는 NP(Network Processor), ATCA(Advanced Telecommunications Computing Architecture), OpenFlow 기술을 이용한 방법이 있다.

- “S/W programmable” 방법

S/W 기반의 네트워크 가상화 접근 방법은 cost/flexibility 측면에서 우수성을 갖으나, performance 측면에서 약점을 갖고 있다. 이러한 접근 방법의 예로는 Planetlab, VINI, 넓은 의미에서 Overlay 기술들을 이용한 방법이 있다.

- “Virtualizing off-the-shelf nodes” 방법

위의 두 가지 방법이 아닌 기존에 이미 설치된 네트워크 장비를 대상으로 네트워크 가상화를 시도하는 기법이다. 직관적으로 예상할 수 있듯이, cost/performance 측면에서 강점을 갖을 수 있으나, 이미 제공하는 “vlan”과 같은 기존 네트워크 장비에서 제공하는 명령어들을 기반으로 가상화를 실현하는 방법이므로 flexibility 측면에서 약점을 갖는다. Vlan을 기반으로 한 Ethernet 스위치의 가상화, UCLP(User Controlled LightPath) 프로젝트에서 SONET/SDH 채널들을 나누어 주는 방법 등이 이러한 범주에 속할 수 있다.

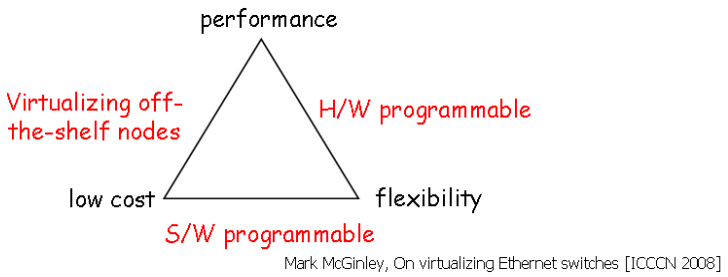

Figure 14 Trade-off in each virtualization approach

위의 기법들 중 “S/W programmable" 접근 방법의 경우, Planetlab은 네트워크 3 계층 이상에서의 새로운 기술의 테스트를 제공하고 있는 글로벌 연구 네트워크로서, 슬라이스 기반의 가상화 또는 자원 할당 등의 테스트를 진행하고 있고, VINI(Virtual Network Infrastructure)는 Planetlab의 경험을 네트워크 2 계층으로 옮겨서 네트워크 가상화를 구현해 오고 있다. 하지만, 이러한 방법들은 실제 네트워크 장비를 대상으로 연구되고 있지 않으므로 제한적인 방법론에 머무르고 있다. 그 외에, 기존 네트워크 장비들을 대상으로 가상

화를 추구하는 기법은 네트워크 장비에서 제공하는 명령어 및 동작에 의존적일 수 있으므로 제한된 범위에서의 네트워크 가상화를 가능하게 하는 수준이며, 파급 효과는 그리 크지 않을 것이다.

NP, ATCA, OpenFlow 기술의 H/W 기반 접근 방법은 패킷 또는 플로우의 처리에 있어서 우수한 성능을 제공하고, 새로운 아이디어를 “scratch" 단계부터 설계하여 적용할 수 있는 특성을 갖는다. 이들 중, NP, ATCA 접근 방법은 가장 고성능의 환경을 제공할 수 있으나, low-level의 프로그래밍, H/W 구조에 대한 풍부한 지식 요구 등 진입 장벽이 높은 것으로 알려져 있다.

반면, OpenFlow를 이용한 접근 방법은 NP, ATCA를 이용한 방법과 달리 네트워크 노드의 기능을 제어 평면과 데이터 평면으로 분리하여 새로운 idea의 구현을 용이하게 한 모듈 구조로서, 데이터 평면은 Flow table에 기반한 포워딩을 수행하고, 시그널링 및 라우팅 기능은 ”Controller“로 불리는 제어 평면에서 분리하여 수행한다. 또한 OpenFlow 기술은 공개된 소스 및 개발환경, API를 제공함으로써, 네트워크 응용 특성에 따른 시나리오를 Controller에 ”Programmable“하게 반영하여 적용할 수 있다.

Table 1 Comparison of hardware-based virtualization approaches

플랫폼	OpenFlow 기반	NP, ATCA 기반
장점	- 제어/데이터 평면 분리를 통한 새로운 idea 적용 용이 - 향후, SONET/SDH 기반의 플로우 정의를 통한 Layer 1까지 확장 가능 - Opensource 기반의 커뮤니티 활성화	- 가장 고성능을 제공하며, Scalability 비교 우위
단점	- 제한된 flow 단위 function만 수행 - Flow별 QoS 제공 기능 부족	- NP, ATCA 등의 H/W 진입 장벽 - low-level programming - High cost
비고	- OpenFlow 연구그룹 커뮤니티 - 국내 FI 커뮤니티	- Jon Turner’s SPP (Supercharging PlanetLab Platform)

5.2 OpenFlow 스위치로 구성된 플로우 기반 네트워크

그림에서처럼 일반 Ethernet 스위치로 구성된 망은 각각의 네트워크 노드들이 forwarding table을 유지하고 있는 반면, OpenFlow 스위치로 구성된 망의 경우 각각의 네트워크 노드들은 forwarding table 대신, NOX와 같은 Controller로부터 생성된 flow entry들을 flow table의 형태로 유지하게 된다. 이렇게 제어 평면과 데이터 평면을 물리적으로 분리함으로써 기존 상용 스위치가 갖는 openness 및 flexibility 측면에서의 경직성에서 벗어나, 각각의 네트워크 응용에 따른 application-driven 네트워크를 구성할 수 있는 환경을 제공할 수 있다. 참고로, 일반 스위치에서 스위치별로 유지하는 forwarding table은 OpenFlow 스위치로 구성된 망에서는 Controller에서 통합하여 유지하고 있는 것으로 생각할 수 있다.

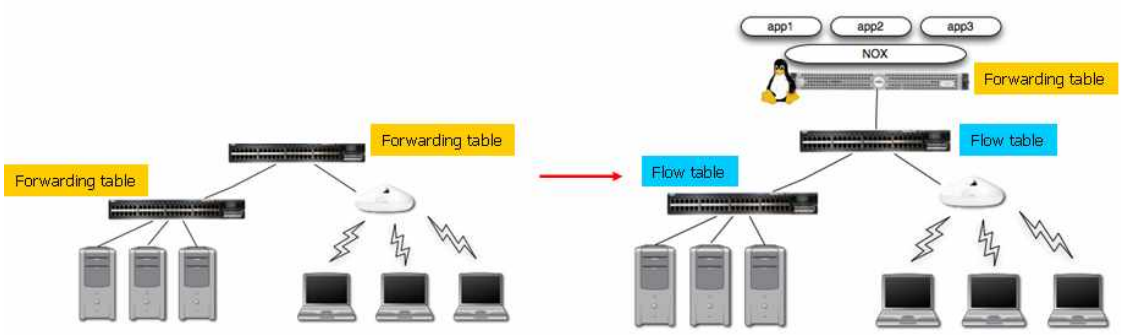

Figure 15 Normal switch vs OpenFlow switch

예를 들어, 그림과 같은 간단한 실험 환경에서 종단 Host A에서 B로 ARP request를 보낸다고 가정하면 OpenFlow 스위치와 Controller 및 호스트 사이에서는 아래와 같은 순서로 OpenFlow protocol 메시지들이 전송되어 각각의 OpenFlow 스위치의 Flow table에 Flow entry로 추가된다. 주목할 것은 Flow entry는 "Flow mod" 메시지의 OFPFC_ADD 명령에 의해 생성되어 추가되는데, “Packet In/Out” 메시지를 통한 일종의 “MAC learning”이 끝난 후에, Controller에 Forwarding table이 형성된 후에야 비로서 "Flow mod" 메시지를 통한 Flow entry 추가가 가능한 것을 볼 수 있다. 실제로, "Packet Out" 메시지를 수신한 OF 스위치는 해당 패킷을 “flooding"시키는 action을 취하고 있고, "Flow Mod" 메시지를 수신한 경우에는 해당 패킷을 특정 물리 포트로 포워딩한다.

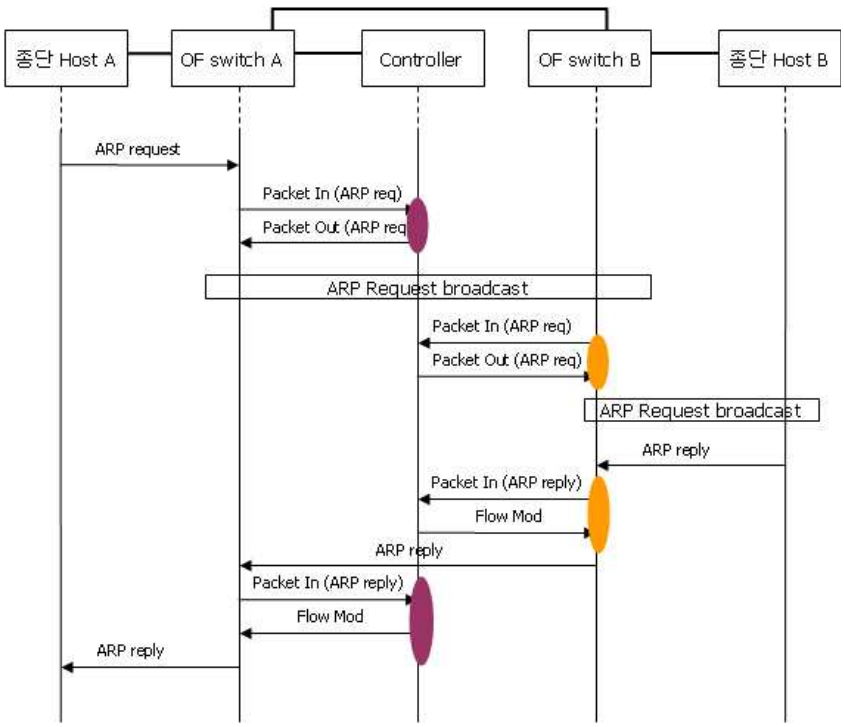

Figure 16 Sequence diagram for adding flow entries

다음 그림에서와 같이 실제로 "Packet Out" 및 “Flow mod" 메시지 구조를 보면 공통적으로 ofp_action_header를 포함하고 있고, 이러한 ofp_action_header에서 정의된 action type에 따라 flooding 또는 특정 포트로의 forwarding을 선택하여 지정할 수 있게 된다. 중요한 것은 이러한 결정이 각각의 OpenFlow switch에서 결정되는 것이 아니라, Controller를 통해 OpenFlow protocol을 통해 지시된다는 것이다. 즉, 이러한 네트워크 노드 측면의 가상화 관점에서 Control plane과 Data plane의 분리를 통해 시나리오 기반의 신규 아이디어 적용 및 네트워크 관리의 용이성까지 제공할 수 있는 장점을 갖는다.

```
/* Send packet (controller -> datapath). */
struct ofp_packet_out {
    struct ofp_header header;
    uint32_t buffer_id;           /* ID assigned by datapath (-1 if none). */
    uint16_t in_port;             /* Packet's input port (OFPP_NONE if none). */
    uint16_t actions_len;         /* Size of action array in bytes. */
    struct ofp_action_header actions[0]; /* Actions. */
    /* uint8_t data[0]; */        /* Packet data.  The length is inferred
                                     from the length field in the header.
                                     (Only meaningful if buffer_id == -1.) */
};
OFP_ASSERT(sizeof(struct ofp_packet_out) == 16);
```

Figure 17 Packet out message

```
/* Flow setup and teardown (controller -> datapath). */
struct ofp_flow_mod {
    struct ofp_header header;
    struct ofp_match match;       /* Fields to match */

    /* Flow actions. */
    uint16_t command;             /* One of OFPFC_*. */
    uint16_t idle_timeout;        /* Idle time before discarding (seconds). */
    uint16_t hard_timeout;        /* Max time before discarding (seconds). */
    uint16_t priority;            /* Priority level of flow entry. */
    uint32_t buffer_id;           /* Buffered packet to apply to (or -1).
                                     Not meaningful for OFPFC_DELETE*. */
    uint16_t out_port;            /* For OFPFC_DELETE* commands, require
                                     matching entries to include this as an
                                     output port.  A value of OFPP_NONE
                                     indicates no restriction. */
    uint8_t pad[2];               /* Align to 32-bits. */
    uint32_t reserved;            /* Reserved for future use. */
    struct ofp_action_header actions[0]; /* The action length is inferred
                                            from the length field in the
                                            header. */
};
OFP_ASSERT(sizeof(struct ofp_flow_mod) == 64);
```

Figure 18 Flow mod message

5.3 FlowVisor 기반의 네트워크 자원 슬라이싱

위와 같은 OpenFlow 기술이 갖는 장점에도 불구하고, 각각의 시나리오를 갖은 네트워크 연구자들이 OpenFlow switch들로 구성된 망을 동시에 병렬적으로 사용할 수 없는 단점이 존재한다. 즉, 서버나 스토리지 가상화 방법에서 일반적으로 말하는 슬라이싱 기법을 제공할 수 없다는 것이다. 이미 널리 사용되는 Vlan tagging 기법을 이용하여 단일 시나리오 기반의 다수의 슬라이스들을 생성하는 방법은 OpenFlow 스위치 및 vlan tagging을 제공하는 Controller 만을 이용하여 가능하지만, 이러한 제한된 접근 방법은 OpenFlow protocol에서 정의하는 10-tuples을 기반으로 한 다수 시나리오마다의 슬라이스를 제공하는 측면에서 한계를 나타낸다.

이러한 단점을 해결하기 위해 Controller의 일종인 FlowVisor가 연구되어 오고 있다. 즉, 다음 그림에서처럼 컴퓨팅 자원을 가상화하는 것과 유사하게 FlowVisor를 통해 네트워크 상의 여러 자원들을 가상화시키는 것이다. 여기서 OpenFlow protocol은 H/W 자원을 추상화시키는 수단으로 볼 수 있고, FlowVisor는 컴퓨팅 자원의 가상화 수단인 Xen, Qemu 등과 같은 역할을 수행한다고 할 수 있다. 결국, FlowVisor 기술을 통해서 OpenFlow로 구성

된 물리적 인프라의 “sharing”과 시나리오 기반으로 분리된 논리적 인프라의 “isolation"이 동시에 가능하게 된다.

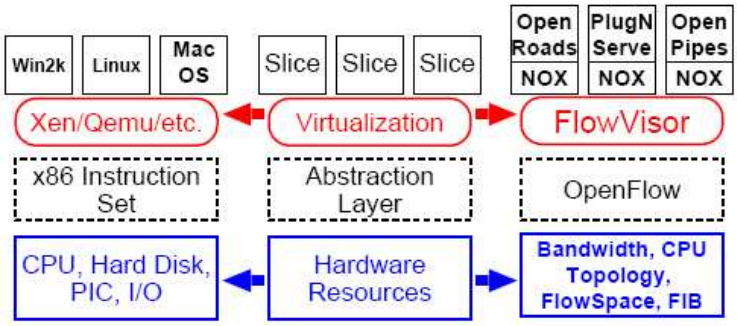

Figure 19 OpenFlow network virtualization using FlowVisor

FlowVisor는 물리 네트워크를 제어하기 위해 OpenFlow 프로토콜을 이용한다. 다음 그림에서처럼 FlowVisor는 다수의 OpenFlow Controller들을 수용할 수 있고, 슬라이스 당 하나의 Controller를 매핑시켜 Controller가 자신 소유의 슬라이스만을 관찰하며 제어할 수 있게 한다. 동시에 슬라이스들 간의 isolation을 유지한다.
그림에서 볼 수 있듯이, FlowVisor는 Controller들로 부터의 OpenFlow 메시지들을 받아들여 각각의 Controller마다 정의된 슬라이싱을 위한 policy를 참고하여, 허가된 자원을 할당한 후 OpenFlow 망의 서브셋으로 구성된 슬라이스를 제어할 수 있게 한다. 반대로, OpenFlow 스위치로부터의 메시지들도 유사한 방식으로 해당 슬라이스의 policy를 만족시키는 Controller에게 전송되게 된다. Controller의 관점에서 FlowVisor는 OpenFlow 스위치로 구성된 네트워크로 보이고, OpenFlow 스위치 관점에서는 Controller로 보임으로써, 투명성을 유지할 수 있다.

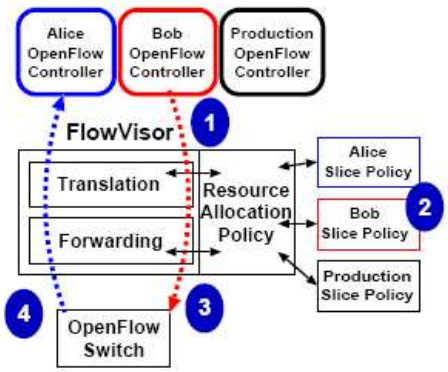

Figure 20 FlowVisor architecture

5.4 서비스 모니터링에 대한 요구사항

현재, 연구망에서는 그림과 같은 OpenFlow 스위치로 구성된 망 자원의 슬라이싱이 가능한 테스트베드를 기반으로 한 네트워크 가상화 서비스를 계획하고 있으며, 자원 슬라이싱의 방법으로 FlowVisor의 도입을 고려 중이다. 현재 FlowVisor는 알파 버전이 발표된 상태이고, FlowVisor 중심의 네트워크 가상화 서비스를 위해서 다수의 Controller들을 동시에 수용하는 통합적인 서비스 프레임워크가 필요한 실정이다.

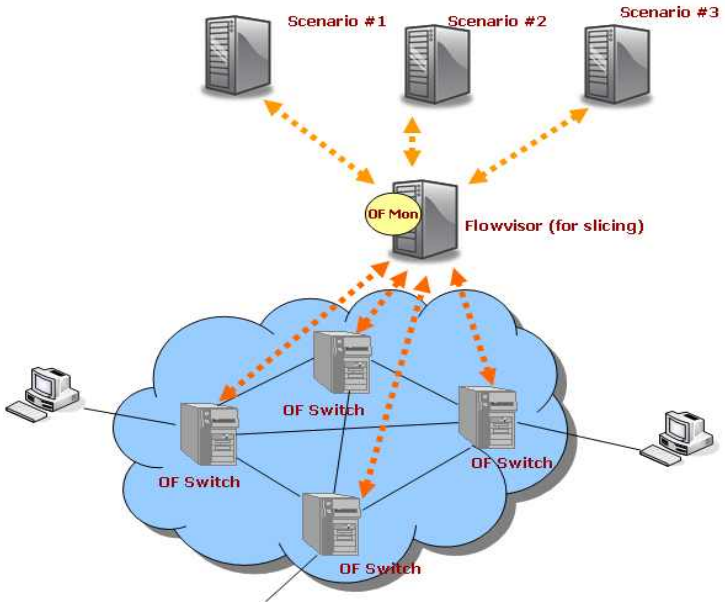

Figure 21 Monitoring for slices of OpenFlow based network resources

이러한 서비스 프레임워크에서 핵심적으로 필요한 요소들은 아래와 같이 나열할 수 있다.

- 다수 네트워크 연구자들의 시나리오 기반으로 작성된 Controller들의 등록 기능
- 슬라이스 모니터링 기능
- 물리 네트워크 토폴로지 및 링크 모니터링 등

위의 기능들 중 일부의 구현은 현재 OpenFlow procotol에서 제공하는 메시지들 중에서 필요한 메시지들을 선택하여 이용할 수 있을 것이다. 대략적인 큰 범주에서 다음 표와 같이 분류가 가능할 것이다.

Table 2 OpenFlow messages for monitoring

메시지 종류	서브 타입	의미
Controller-to-Switch	Features	스위치에서 지원되는 Capabilities (포트, 인터페이스에 대한 물리적 정보)
	Read-State	스위치의 flow-tables, 포트, flow entries들에 대한 statistics 정보
Asynchronous	Flow Expiration	flow table로 부터의 flow entry 삭제 관련
	Port-status	port 설정 상태의 변화 시
Symmetric	Echo	Controller와 switch 간의 지연, 대역폭, 연결 관련

6. 결론

현재, 미국, 유럽, 일본 등에서는 미래인터넷을 위한 대규모의 연구 프로젝트들이 진행 중에 있다. 이러한 미래인터넷의 핵심 기술들 중의 하나인 네트워크 가상화 기술을 통해 하나의 물리 네트워크 상에 다수의 가상으로 독립된 네트워크가 공존함으로써, 복수 개의 네트워크 아키텍쳐와 서비스가 동일한 인프라 상에서 동작할 수 있다.

특히, 로컬 네트워크 실험 환경에서의 한계를 벗어나 KREONET과 같은 운영 중인 백본 환경에서 네트워크 가상화 기법을 도입함으로써 실질적인 네트워크 연구 및 테스트를 수행할 수 있을 것으로 기대된다. 또한 존재하는 여러 네트워크 가상화 기술들에 비해 OpenFlow/FlowVisor 기반의 네트워크 가상화 환경을 구성함으로써 아래와 같은 장점을 얻을 수 있다.

- Open control의 개념으로 실제 사용자에 실제 네트워크 자원들에 대한 제어 권한을 부여함으로써 획기적인 테스트환경을 제공할 수 있다.
- KREONET 백본을 운영하는 KISTI의 입장에서 실제 백본 기반의 네트워크 자원의 제공이 가능함으로써 보다 실제에 가까운 가상화 환경의 제공이 가능하다.
- Controller의 효과적인 운영 및 구성으로 사용자뿐만 아니라 네트워크 운영자 입장에서도 네트워크 및 자원의 효과적인 관리가 가능하다.
- 일반 트래픽과 실험 트래픽의 확실한 분리로 상호간 트래픽에 영향을 주지 않는 가운데 실제 환경에서의 실험이 가능함으로 실험 검증의 수준이 높아진다.

이러한 장점들을 반영하고, 효율적인 네트워크 가상화 테스트베드 환경을 제공하기 위해서는 OpenFlow/FlowVisor 기반의 네트워크 자원 슬라이싱 테스트베드 및 물리네트워크 토폴로지/링크/슬라이스 모니터링 기능 등을 지원하는 통합적인 서비스 프레임워크가 필요할 것으로 사료된다.

7. 참고문헌

[1] GENI project, http://www.geni.net
[2] FIRE, http://www.future-internet.eu
[3] FIF, http://fif.kr
[4] AKARI, http://akari-project.nict.go.jp
[5] 김대영, 문수복, 박성용, 변성혁, 이순석, 신명기, 정일영, “네트워크 가상화에 대한 고찰”, 정보과학회지 제 26권 제 10호 통권 제 233호, 2008년 10월
[6] 권순종, “미래인터넷을 위한 네트워크 가상화 기술”, 정보과학회지 제 26권 제 10호, 2008년 10월
[7] M. McGinley, T. Li, and M. Veeraraghavan, "On Virtualizing Ethernet Switches," in Proc. of IEEE ICCCN'08, Aug. 3~7, 2008, St. Thomas, U.S. Virgin Islands
[8] A virtual network infrastructure. http://www.vini-veritas.net
[9] The OpenFlow Switch Consortium. http://www.openflowswitch.org
[10] OpenFlow Switch Specification 0.8.9, http://www.openflowswitch.org
[11] Saurav Das, Guru Parulkar and Nick McKeown, Simple Unified Control for Packet and Circuit Networks, Whitepaper
[12] NOX: An OpenFlow Controller. http://noxrepo.org/wp/
[13] Larry Peterson, John Wroclawski, "Overview of the GENI architecture", GDD-06-11, January 5, 2007.
[14] FlowVisor. http://www.openflowswitch.org/wk/index.php/FlowVisor
[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, 38(2):69-74, April 2008
[16] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, NOX: Towards an operating system for networks, ACM SIGCOMM Computer Communication Review, July 2008
[17] Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Te-Yuan Huang, Michael Chan, Nikhil Handigol, and Nick McKeown, OpenRoads: Empowering Research in Mobile Networks, ACM SIGCOMM Poster, Aug 2009
[18] Nikhil Handigol, Srini Seetharaman, Mario Flajslik, Nick McKeown, and Ramesh Johari, Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow, ACM SIGCOMM Demo, Aug 2009
[19] Glen Gibb, David Underhill, Adam Covington, Tatsuya Yabe, and Nick McKeown, OpenPipes: Prototyping high-speed networking systems, ACM SIGCOMM Demo, Aug 2009
[20] Rob Sherwood, Michael Chan,Glen Gibb, Nikhil Handigol,Te-Yuan Huang,Peyman Kazemian, Masayoshi Kobayashi, David Underhill, Kok-Kiong Yap, Guido Appenzeller, and Nick McKeown, Carving Research Slices Out of Your Production Networks with OpenFlow, ACM SIGCOMM Demo, Aug 2009
[21] Rob Sherwood, Glen Gibby, Kok-Kiong Yapy, Guido Appenzellery, Martin Casado, Nick McKeowny, and Guru Parulkary, FlowVisor: A Network Virtualization Layer, Technical Report