미래인터넷 테스트베드를 위한 안정적인 실험자원의 공급이란?

문서 상태: 버전 0.9 최종 초안 (최종수정일: 2011-11-21)

주의: 본 이슈 분석서의 버전 0.9 는 1 차적인 내용 작성을 완료해서 잠정적으로 공개하여 보완 의견을 청취할 목적으로 2011 년 11 월에 준비되었으며, 향후 보완 과정을 거쳐서 정식 버전 1.0 을 만들 예정임.

작성자: 한상우, 차병래, 김종원 (광주과학기술원 정보통신공학부)
{ swhan, brcha, jongwon}@nm.gist.ac.kr

요약: 미래인터넷 테스트베드는 연구자들이 창의적인 네트워킹 개념들을 실험적으로 실증할 수 있도록 연결용 네트워크와 이에 연동된 장비들, 그리고 이를 지원하는 각종 실험용 소프트웨어 일체를 지칭하는 일종의 가상적인 실험실이 되어야 한다. 즉 기존의 네트워크 연결 위주의 실험에서 벗어나 새로운 서비스를 지향하는 다양한 실험자들의 요구사항들을 수용하는 것이 매우 중요하다. 이를 위해서 활용되는 컴퓨팅/네트워킹 자원들이 여러 네트워크 계층에서 프로그램이 가능해야 하며, 다수의 실험자들이 자신들의 실험들을 동시에 수행하도록 컴퓨팅/네트워킹 자원의 가상화도 필요하다. 또한 확장성을 검증하기 위해서 대규모 환경에서 지속적으로 실험하면서 검증할 수 있어야 한다. 이러한 요구를 감안하여, 본 문서에서는 미래인터넷 테스트베드에서 안정적인 실험자원의 공급을 위한 요구사항을 도출하고자 한다. 또한 슬라이스 기반으로 실험자원을 관리하는 관련된 테스트베드 구축 및 운용에 관한 연구 동향을 분석하여 안정적으로 실험을 유지하기 위해 탄력적으로 실험자원을 공급하기 위해 고려해야 하는 각종 이슈들에 대해 논의한다.

주제어: 미래인터넷 테스트베드, 컴퓨팅/네트워킹 자원, 슬라이스 기반 실험 관리, 자원의 프로그래밍 및 가상화

목 차

1. 서론

첨단 디지털 저작기술의 발전과 스마트 네트워크 컴퓨팅 장치들의 보급은 고성능 미디어 응용서비스의 적용 범위를 실생활의 모든 분야로 광범위하게 확대시키고 있다. 미래의 인터넷을 통해 제공될 응용서비스들은 다양한 그리고 복합적인 형태의 디지털 정보, 매체, 콘텐츠들을 창출하고, 이들은 사용자의 요구에 따라 상황별로 가공, 조립되어 안전하고 빠르게 제공될 것이다 [1]. 그러나 현재의 인터넷은 실감 콘텐츠의 보급으로 인한 폭발적인 트래픽 증가와 수준 높은 서비스 품질에 대응할 근본적인 해결책을 제시하지 못하고 있는 실정이다 [2]. 이런 문제를 근본적으로 해결하기 위해, 인터넷을 처음부터 재설계하자는 clean-slate 개념에 따라, 혁신적이고 창의성이 있는 새로운 미래를 위한 네트워크 구조를 수용하기 위한 미래인터넷 관련 연구들이 전 세계적으로 활발히 진행되고 있다. 이러한 연구들은 확장성, 보안성, 이동성, 이질성, 서비스 품질, 자동설정, 상황인지, 관리성, 데이터 중심, 경제성이라는 주요 요구사항들을 세우고 여기에 맞는 새로운 네트워크 구조를 그리면서 한편으로는 이를 실증할 테스트베드 프로토타입의 개발을 병행하고 있다 [3]. 예를 들면 미래인터넷을 위한 새로운 아이디어들을 다양하게 살펴보았던 FIND (Future INternet Design) [4] 프로젝트에 이어서 4 가지 가능성 있는 아키텍처를 연구하는 FIA (Future Internet Architecture) 프로젝트가 한편에 있다. 그리고 이들 프로젝트들에서 제안될 새로운 미래인터넷 아키텍처들을 대규모 사용자들을 대상으로 하여 검증해 보는 대규모의 실증적 테스트베드를 나선형 (spiral) 개발 방법론에 따라 구축하는 GENI (Global Environment for Network Innovations) [5] 활동이 다른 한편의 대표적인 사례이다. 또한 유럽의 FP7(Future Internet in Framework Programme 7)이 후원하는 FIRE (Future Internet Research and Experimentation)[6], 일본의 NWGN (New Generation Network) [7]도 테스트베드 부분에서도 비슷한 구도를 나름대로 그리면서 테스트베드에 관한 차분하고 의미가 있는 진전을 위해 노력하고 있다.

미래인터넷을 연구하기 위한 테스트베드는 연구자들이 창의적인 네트워킹 개념들을 실험적으로 실증할 수 있도록 연결용 네트워크와 이에 연동된 장비들, 그리고 이를 지원하는 각종 실험용 소프트웨어 일체를 지칭하는 일종의 가상적인 실험실이 되어야 한다. 즉 기존의 네트워크 연결 위주의 실험에서 벗어나 새로운 서비스를 지향하는 다양한

실험자들의 요구사항들을 수용하는 것이 매우 중요하다. 이를 위해서 활용되는 컴퓨팅/네트워킹 자원들이 여러 네트워크 계층에서 프로그램이 가능해야 하며, 다수의 실험자들이 자신들의 실험들을 동시에 수행하도록 컴퓨팅/네트워킹 자원의 가상화도 필요하다. 또한 확장성을 검증하기 위해서 대규모 환경에서 지속적으로 실험하면서 검증할 수 있어야 한다. 이러한 요구사항에 따라, 소프트웨어 기반 네트워킹과 클라우드 기반의 컴퓨팅을 위주로 실험자가 원하는 테스트베드의 모형을 정립하고 이들을 제어하기 위한 GENI 제어 프레임워크들(PlanetLab, ProtoGENI, ORGA, OMF 등)이 보다 구체화되고 있다. 또한 독일의 G-Lab, 일본의 CoreLab, 유럽의 OneLab 등 여러 테스트베드들에 속한 자원들을 실험자가 한번에 활용할 수 있도록 하는 자원 수준의 연동이 여러 곳에서 시도되고 있다.

장기간 지속적으로 미래인터넷 관련 실험을 수행하려면 안정적인 실험자원의 확보가 우선적으로 보장되어야 한다. 이를 위해서는 실험자가 필요한 자원을 정량적으로 기술하게 하여 그에 맞는 자원을 확보하여 안정적으로 약속된 대로 자원이 제공되고 있는지를 감시하면서 문제가 발생하면 이를 해결하여 자원을 지속적으로 공급하는 테스트베드 관리자를 위한 기술 지원이 중요하다. 이에 본 문서에서는 미래인터넷 테스트베드에서 안정적으로 실험자원을 공급하기 위해 살펴보아야 하는 이슈들을 다음과 같이 살펴본다.

- **서비스 지향 테스트베드가 과연 필요한가?** 미래인터넷 테스트베드에서 네트워킹은 실험에 직접적인 영향을 미치는 중요한 부분이지만, 미래인터넷의 유용성은 콘텐츠와 이를 활용하는 서비스들에 의해서 검증된다는 점에 유의해야 한다. 이를 위해 테스트베드는 실험자가 요구한 네트워킹/컴퓨팅 자원의 특성과 양을 충족시키는 실험 환경을 제공해야 한다. 또한 테스트베드는 실험용으로 할당된 네트워킹/컴퓨팅 자원들을 안정적으로 공급하면서, 준비된 자원들이 서비스들에 의해 효율적으로 소비될 수 있도록 자원의 공급과 서비스의 소비 관계를 계획해야 한다.
- **가상화(virtualization)는 얼마나 필요한가?** 테스트베드는 실험자에게 자원들의 하드웨어적인 특성들을 숨기는 것이 좋다. 테스트베드는 실험자들이 특정한 종류와 양으로 추상화된 실험 자원만 이해할 수 있도록 도와주는 것이 바람직하다. 또한 테스트베드는 실험들이 또 다른 실험의 진행에 (성능 상의) 영향을 미치지 않도록, 실험별로 나누어진 자원들 사이에 엄격한 단절 (isolation)을 유지해야

한다. 이를 통해 실험자에게 독립된 실험환경을 제공함으로써, 동시에 많은 실험들이 수행될 수 있도록 해야 한다. 높은 수준의 가상화를 지원하는 테스트베드일 수록 더 많은 실험들을 장기간 동안 지속적으로 수행할 수 있다. 뿐만 아니라, 테스트베드가 보유한 전체 자원에서 필요한 양만큼만 개별 실험자들에게 할당할 수 있으므로 테스트베드 자원을 효율적으로 운용할 수 있게 된다.

- **프로그램화(programmability)는 얼마나 필요한가?** 테스트베드는 할당된 자원 위에 준비된 실험 코드를 설치하여, 의도한 기능을 제공하는 실험 자원으로 만들 수 있어야 한다. 예를 들면, 테스트베드는 가상 머신들을 위해 설치할 이미지를 동적으로 내려 받고 설치할 수 있어야 한다. 또한 테스트베드는 실험자들이 원하는 기능을 구현할 수 있도록 자원이 보유한 능력을 손쉽게 프로그래밍할 수 있는 환경을 제공해야 한다. 이 환경을 통해 실험자는 더욱 복잡하고 세밀한 실험을 위해 주어진 실험자원의 능력을 최대한 활용할 수 있게 된다.

제기한 이슈들을 다음과 같은 과정을 걸쳐서 분석하고 설명해 보고자 한다. 먼저 2 절에서 실험의 요구사항들에 대해 논의해 보고, 이어서 3 절에서는 실험자원의 공급 관리를 위한 배경 기술들을 설명하고, 4 절에서 탄력적으로 실험자원을 공급하기 위한 실증적인 개념과 이를 위한 기본적인 접근 방법에 대해 설명한다. 그리고 마지막으로 5 절에서 본 문서를 맺는다.

2. 효율적인 실험 지원을 위해 필요한 기능 요구들

미래인터넷 테스트베드는 컴퓨팅/네트워킹 자원들의 집합을 넘어, 실험자들이 수행할 미래지향적인 네트워킹 실험들을 편리하게 수행할 수 있도록 여러 가지 유용한 기능들을 제공해야 한다. 첫째로 실험자들은 테스트베드를 통해 실험에 필요한 자원들을 추상화된 형태로 선택/이용할 수 있어야 한다. 각 실험자는 개방된 인터페이스들을 통해 주어진 제어 권한에 따라 실험 과정에서 소요되는 컴퓨팅과 네트워킹 자원을 원하는 형태와 양, 그리고 이 자원들의 연결 관계를 명시적으로 기술하여 테스트베드에게 요청해야 한다. 여기에 대응하여 테스트베드는 계산 자원(CPU, GPU), 저장 자원(메모리, HDD), 네트워크 자원(대역폭, 버퍼)의 능력을 가상화하여 단위 자원으로 준비하게 된다. 둘째로, 실험자는

자원들의 능력을 정량화된 수치로 이해할 수 있어야 한다. 테스트베드들은 이종 노드들로 구성되는 경우가 일반적이며, 이로 인해 실험자는 이종 노드들이 제공하는 계산, 저장, 네트워킹 능력을 일관된 기준으로 가늠하기 어렵다. 따라서 특정 기능을 공급하기 위한 소요 자원의 형태와 양을 정량화하여, 사용자가 실험에 필요한 자원을 주의 깊게 선택할 수 있도록 도와주는 것이 필요하다. 셋째로 공급하는 자원의 성능 변화를 주어진 (또는 상황에 따라 변화된) 시간 간격 마다 측정하여 실험과정에서 계산, 저장, 네트워크 자원의 양이 부족하지 않은지 점검해야 한다. 실험자가 요청한 경우, 테스트베드는 실험자에게 사용 중인 자원의 상태와 변동 추이를 제때에 알려, 실험자가 이에 선제적으로 대처할 수 있도록 도와주어야 한다. 마지막으로 테스트베드 위에서 콘텐츠와 서비스들을 쉽게 설정하고 제어하기 위한 상위 수준의 소프트웨어 중심의 실험제어 기능들이 포함되어야 한다. 이를 위해 테스트베드는 사용자 요구에 대응하여 콘텐츠의 지속적인 공급과 소비를 위해, 네트워킹/컴퓨팅 자원들을 이용하는 서비스들의 연결 관계와 제어방법을 쉽게 기술하고, 그에 맞게 효율적인 방법으로 적시에 서비스들을 합성해야 하며, 서비스 합성이 적절한 품질로 계속 유지되는지 정량적으로 측정할 수 있는 도구를 제공해야 한다.

이러한 요구사항들과 비교한다면, 현재 미래인터넷 테스트베드들은 자원(resource)의 가상화와 프로그램화에 주력하면서 실험자가 요청한 자원을 확보해줄 수 있는 편리한 도구를 제공하는데 초점을 두고 있다. 특히 실험에 필요한 자원을 요구하는 부분에서 많은 진도를 보이고 있는데, GENI 에서는 자원 요구사항을 정형화된 틀에 맞게 기술하는 RSpec 을 현 단계에서 논의하고 있으며, ORCA 와 SURFnet 에서는 테스트베드의 다양한 측면을 지식으로 표현하기 위해 RDF (resource description framework)과 OWL (web ontology language)와 같은 확장 가능한 시맨틱 웹 언어들을 이용하여 컴퓨팅/네트워킹 자원들, 네트워킹 경로들, 서비스들, 서비스 합성을 위한 제약 조건들, 개체들의 관계들을 의미론적으로 정의하고 있다. 리소스 능력을 정량화하는 부분은 미진한 진도를 보이고 있으며, 테스트베드의 성능 모니터링 부분에서는 GENI I&M (instrumentation and measurement) 논의를 중심으로 하여 Instools 과 OML 를 비롯한 여러 도구들이 활용되고 있다 [8]. 마지막으로 서비스 운용 및 제어와 관련해서는 네트워킹 프로토콜 스택을 세분화/추상화한 뒤 서비스 지향 구조에 따라 이들을 조합하고 세밀한 조절하는 SILO 등이 대표적이나, 아직 대부분의 테스트베드 구축 및 운용에는 실제로 적용되지는 않은 상태이다 [9].

3. 슬라이스 기반 실험관리를 활용하는 관련 테스트베드 동향

슬라이스는 실험을 위해 할당되고 설정된 컴퓨팅/네트워킹 자원들로 구성된 실험용 네트워크이다. 슬라이스는 서로 다른 관리 영역들안에 있으면서 여러 지역에 흩어진 자원들을 포함할 수 있다. 즉 슬라이스는 테스트베드의 운용규칙에 따라 실험자들이 원하는 실험을 하기 위해 사용하는 자원들의 네트워크를 가리키는 추상적인 개체이다. 실험자들은 그들에게 할당된 슬라이스를 통해 분산된 자원들을 원격에서 발견, 예약, 설정, 프로그램, 디버깅, 운용, 감시, 관리, 중단하는 도구들을 사용하다 [GENI-SE-SY-RQ-02.0.pdf]. 이러한 슬라이스 기반의 실험관리를 활용하는 테스트베드 설계에 관한 연구들이 진행되고 있으며, 안정적으로 슬라이스를 유지하기 위한 노력들도 병행되고 있다. 본 절에서는 몇 가지 사례를 소개하면서 슬라이스 관리에 관한 동향을 살펴본다.

3.1. Slice-based Federation Architecture

SFA(Slice-based Federation Architecture)는 슬라이스 기반의 네트워크 테스트베드들의 연동을 허용하는데 필요한 인터페이스들과 자료형들을 정의한 제어 프레임워크 아키텍처이다 [10]. SFA 는 제어 프레임워크를 통해 운용되는 객체들의 핵심적인 형태를 4 가지로 나누는데, 이는 소유자와 운영자, 테스트베드를 이용하는 연구자와 이 연구자들을 인증해주는 보증인으로 구성된다. SFA 는 실험 환경 제공을 위해 정의한 객체들의 역할을 다음과 같이 조정한다.

- 소유자들의 통제 아래에서 테스트베드 자원들을 공급하기 위해, 소유자들이 자원의 할당과 사용에 관한 정책들을 선언하도록 함.
- 운용자들이 테스트베드에 새로운 장치를 설치하고, 구식 또는 고장난 장치들을 제거하고, 시스템 소프트웨어를 설치하거나 업그레이드하면서, 테스트베드의 성능, 기능, 보안 상태를 감시하도록 함.
- 연구자들이 슬라이스를 만들고, 그들에게 자원을 할당하며, 슬라이스안에서 실험 소프트웨어들을 돌릴 수 있도록 함.
- 소유자들과 연구자들이 (슬라이스들 위에서 자원들을 관리하고 접근하는) 인가 규칙들을 명시하도록 함. 예를 들면, 인가 규칙들은 지정된 PI(Principal

Investigator)들이 그들의 조직에 있는 연구자들을 식별하고 인증하여, 주어진 실험 자원들을 이용하는 것을 허용할 수 있음.

SFA 는 자원을 컴포넌트(component)로 추상화하는데, 컴포넌트에는 컴퓨터, 라우터, 프로그램화 가능한 액세스 포인트가 대응될 수 있다. 컴포넌트는 한 장치 또는 분산된 장치들에 의해 공급될 자원들을 캡슐화하여 물리적인 자원들(예, CPU, 메모리, 디스크, 대역폭), 논리적인 자원들(예, 파일 기술자, 포트 번호), 합성된 자원들(예, 패킷 전달 경로)을 만들 수 있다. 이러한 컴포넌트는 AM(Aggregate Manager)의 통제 아래에 있는 Aggregate 의 형태로 관리된다. 한 컴포넌트는 여러 실험자들을 위한 컴포넌트 자원들로 나뉠 수 있는데, 이러한 행위를 실험자가 컴포넌트의 슬리버(sliver)를 얻는다고 말한다. 각 컴포넌트는 하드웨어 또는 소프트웨어 기반 가상화 기술을 통해 슬리버들을 다른 슬리버들로부터 격리시킨다. 또한 슬리버는 실험 코드를 로딩하고 실행할 수 있는 능력을 지닌 자원이면서 터널 (tunnel), VLAN (Virtual LAN), light-path 와 같은 통신 자원들도 될 수 있다.

SFA 는 행위자(소유자, 운용자, 연구자, 보증간)들 사이에 슬라이스에 기반을 둔 자원 접근권한 허가와 철회를 하는데 사용하는 자료형을 다음과 같이 정의한다.

- RSpec 은 Aggregate 에서 제공하는 가용 자원들을 XML 문서형태로 기술하고, 실험자가 필요한 자원의 형태와 양을 기술하는데 사용되는 형식이다. SFA 는 RSpec 에 대한 통일된 형식은 요구하지 않으며 Aggregate 에 따라 고유한 형식을 가질 수 있다.
- Ticket 은 AM 이 서명한 RSpec 으로써 사용자가 요청한 자원을 주어진 시간동안 할당하겠다는 증서이다. 발급한 티켓은 요청한 자원이 할당된 후에 회수된다.
- Credential 은 실험자가 자원 사용에 대한 합법적인 권한이 가지고 있다는 것을 입증하는 증명서로써, Credential 의 구체적인 형식은 제어 프레임워크에 따라 달리 결정된다.

3.2. GENI Plastic Slice Project

GENI Plastic Slice Project 는 실험자에게 제공되는 GENI 자원의 품질을 향상시키기 위해, 중규모(meso-scale: 8 개의 캠퍼스를 연결한 전국 규모의 인프라에서 3 개월 동안 10 개의

GENI 슬라이스들을 실행하는) 실험들을 진행할 목표를 가지고 있으며, 그러한 환경을 관리하는 경험을 얻을 예정이다 [11]. 또한 이 프로젝트는 장기간 실험을 위한 다중 슬라이스 관리와 관련된 기술적인 이슈들을 발굴하고 이런 실험들을 지속적으로 지원하기 위한 간편한 운용 절차를 제공하는 것을 목표로 한다. 구체적으로 다음과 같은 기술적인 과제들을 다룬다.

- 슬라이스안에 다중 aggregate 들로부터 공급되는 자원을 포함
- 슬라이스를 지원하는 종단간 데이터 평면들을 제어하는 실험 지원
- 슬라이스에 포함된 자원들을 실험팀 구성원들이 공유하는 실험 지원
- 수개월동안 실험들을 연속적으로 수행
- non-IP 프로토콜 실험들을 지원
- 실험안에서 IP 와 non-IP 경로들의 성능 비교 지원
- 영상/음성 어플리케이션 지원

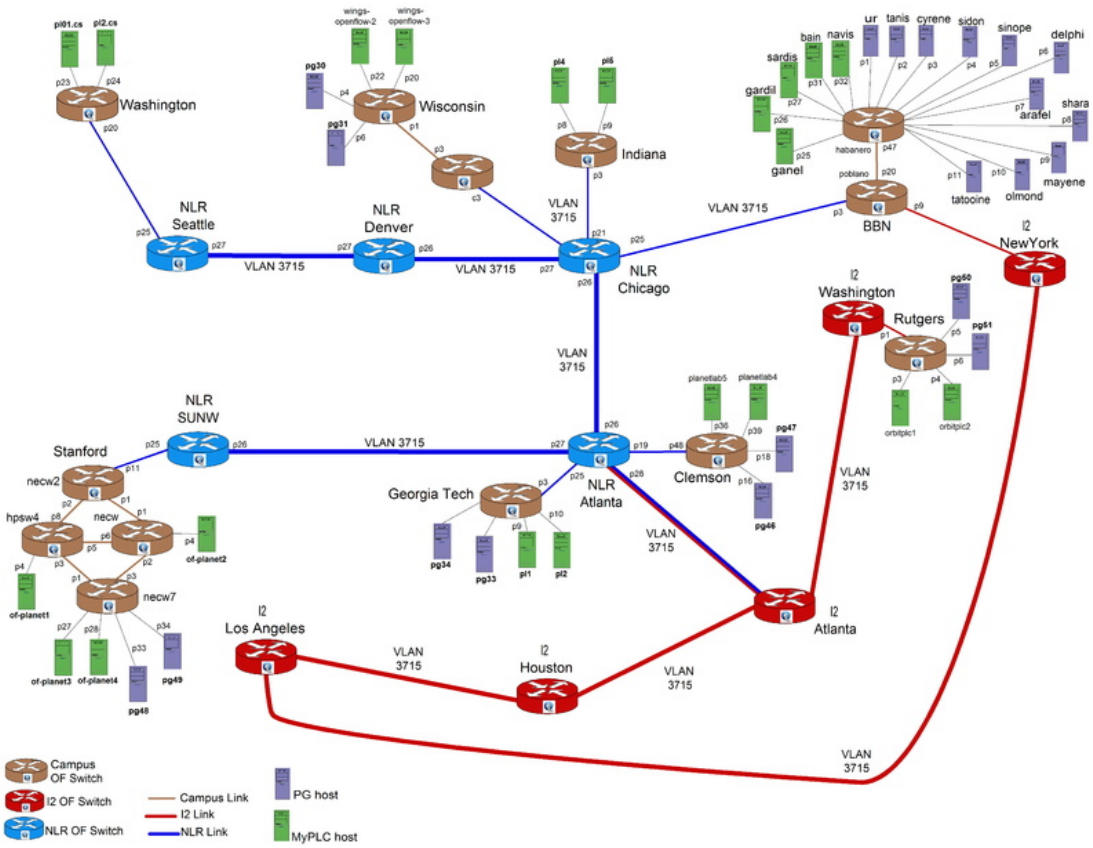

그림 1. GENI Plastic Slice Project 의 주요 VLAN 구성도 [11].

그림 1 은 GENI Plastic Slice Project 를 위해 구축된 VLAN 의 구성을 보여준다. 이 프로젝트에서는 슬라이스들을 이용하여 ping 실험, 비암호화된 TCP 스트림을 분석하는 netcat, HTTPS (Secure Hypertext Transfer Protocol)를 통해 웹 콘텐츠를 내려받는 wget 실험, 그리고 TCP 와 UDP 의 IPerf 의 5 가지 실험을 수행하였다. 향후에는 보다 복잡한 실험하기 위해, 모바일 단말과의 제어 및 데이터 연결들을 지원하고, GENI 슬라이스에서 클라우드 서비스를 이용할 수 있도록 하며, 실험에서 1000 개 이상의 호스트 이용을 지원하는 목표를 가지고 있다. 또한 실험자가 통제하는 부하분산 기법과 QoS 기능들도 추가로 지원할 예정이다.

3.3. 슬라이스 기반의 GENI Cloud

GENI Cloud 는 클라우드 컴퓨팅 자원들을 GENI 와 호환성을 가지는 테스트베드들과 연동시킴으로써, 계산 자원과 저장 자원들을 실험자들에게 공급하고 실험 코드들을 유칼립투스(Eucalyptus) 클라우드 위에서 돌릴 수 있도록 한다. GENI Cloud 는 SFA 를 준수하면서 슬라이스를 만들고, 가상머신들을 할당하고 회수하는 기능을 구현하고 있다. 첫 단계로 GENI Cloud 는 유칼립투스 Aggregate 를 PlanetLab Aggregate 와 연동하는 작업을 진행하고 있다 [12]. 여기서 유칼립투스 Aggregate 는 클라우드 컴퓨팅 자원들을 통제하고 관리한다. 세부적으로 설명하면, 실험자들이 유칼립투스의 가상머신들을 이용하여 슬라이스들을 만들고자 한다면, 실험자들은 그들이 원하는 가상머신의 능력을 Rspec 으로 기술한 다음, 슬라이스 관리자에게 전달한다. 슬라이스 관리자는 유칼립투스 AM 을 포함한 관련된 AM 들에게 슬리버를 만들어 줄 것을 요청한다. 이때 유칼립투스 AM 은 전달받은 RSpec 을 해석한 후에, RSpec 에 해당하는 가상머신 인스턴스를 만들고, 이 인스턴스를 슬라이스들에게 할당한다.

4. 탄력적이고 안정적인 실험자 지원을 위한 슬라이스의 관리

각각의 실험자에게 부여된 슬라이스는 실험에 사용되는 컴퓨팅과 네트워킹을 위한 각종 자원들의 집합에 대한 실험자의 권한을 연결하는 고리이다. 각 실험자는 슬라이스를 이용하여 확보된 자원들에 자신이 원하는 서비스 기능을 제공할 수 있는 응용 코드들을

수행하고 이에 대한 필요한 조정 및 관찰을 지속적으로 진행해야 한다. 서비스 코드들이 안정적으로 동작하기 위해서는 2 절에서 설명한 것과 같이 자원의 상태를 계속적으로 관찰해서 파악하고 있으면서 문제가 발생하면 적절히 대응하는 조정 기능이 필요하다. 이러한 맥락에서 슬라이스에는 실험을 보조하기 위한 특별한 기능들이 추가되어야 하는데, 이에 대한 상세한 내용을 본 절에서 설명한다.

4.1. 슬라이스의 구성

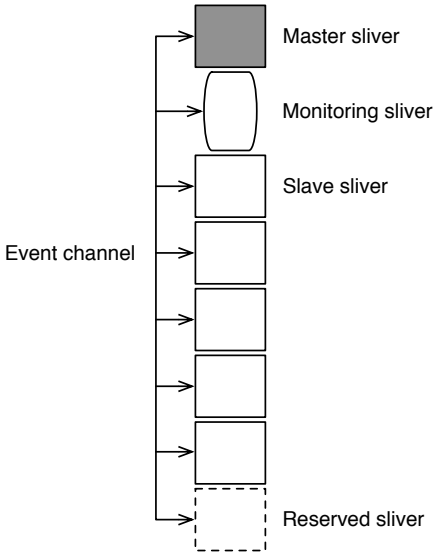

그림 2. 슬라이스의 구성.

하나의 슬라이스는 컴퓨팅과 네트워킹 자원들을 제공하는 슬리버들뿐만 아니라, 특별한 기능들을 수행하는 슬리버들도 함께 포함하며, 이는 그림 2 과 같다. 슬라이스는 마스터 슬리버 (master sliver), 슬레이브 슬리버 (slave sliver), 감시 슬리버 (monitoring sliver), 예비 슬리버(reserved sliver)로 구성된다. 또한 슬리버들 간에 정보 전달을 위한 이벤트 채널(event channel)이 포함된다. 개별적인 슬리버들의 역할은 아래와 같다.

- 마스터 슬리버: 마스터 슬리버는 모니터링 슬리버를 통해 각각의 슬레이브 슬리버의 상태을 파악하면서 문제가 발생한 슬레이브 슬리버들을 예비 슬리버로 교체한다.
- 슬레이브 슬리버: 슬라이스를 구성하는 컴퓨팅 자원과 네트워킹 자원을 제공한다.

- 감시 슬리버: 모니터링 슬리버은 슬레이브 슬리버들의 성능을 감시하면서 슬라이스가 안정적으로 유지되고 있는지 판단한다. 만약 문제가 발생하면 마스터 슬리버에게 통지한다.
- 예비 슬리버: 운용중인 슬레이브 슬리버가 비정상적으로 동작할 경우에 대비해, 예비 슬리버를 운용한다. 슬라이스 안에서 예비 슬리버는 문제가 발생한 슬레이브 슬리버를 대신할 수 있다.
- 이벤트 채널: 슬라이스 안에 있는 모든 슬리버들이 이벤트 정보를 주고받을 수 있는 통로를 제공한다. 이벤트 채널을 통해, 마스터 슬리버, 슬레이브 슬리버들, 감시 슬리버들은 제어 명령과 이벤트, 상태에 관한 정보들이 교환한다.

4.2. 슬라이스들의 연결

주어진 서비스를 실행하는데 필요한 자원들이 기존 슬라이스에 없다면, 필요한 능력을 제공할 수 있는 다중 슬라이스들을 연결하여 사용할 수 있다. 예를 들면, 컴퓨팅 자원을 공급하는 RA(들)에서 만들어진 슬라이스와 네트워킹 자원을 공급하는 RA(들)에서 만들어진 슬라이스가 합쳐질 수 있다. 이와 같은 다중 슬라이스들의 연동을 통한 서비스 운용을 하기 위해서는 슬라이스간에 연결 관계가 명확히 정리되어야 한다. 그림 3 와 같이 슬라이스들을 연결하기 위해 주 슬라이스(primary slice)와 하나 이상의 보조 슬라이스(secondary slice)들이 연결된다. 이때주어진 연동 프로토콜에 따라 다중 슬라이스의 슬리버들을에 대한 접근 권한을 부여하고, 연결시킨다. 연결되는 슬라이스들은 이벤트 채널을 통해 이벤트 정보를 주고받는다. 주 슬라이스는 연결되는 슬라이스들이 주어진 서비스를 잘 지원할 수 있도록 관리하면서, 서비스 코드의 실행을 위해 할당된 자원들로부터 문제가 발생하면 주어진 규칙(예, 예비 슬리버 운용)에 따라 해결한다. 주 슬라이스는 마스터 슬리버를 가지고 있는데, 이 마스터 슬리버는 주 슬라이스의 슬레이브 슬리버들뿐만 아니라 보조 슬라이스의 슬리버들도 함께 관리한다. 주 슬라이스와 보조 슬라이스가 가지고 있는 감시 슬리버는 슬레이브 슬리버들로부터 수집한 상태 정보를 주 슬라이스의 마스터 슬리버에게 전달한다.

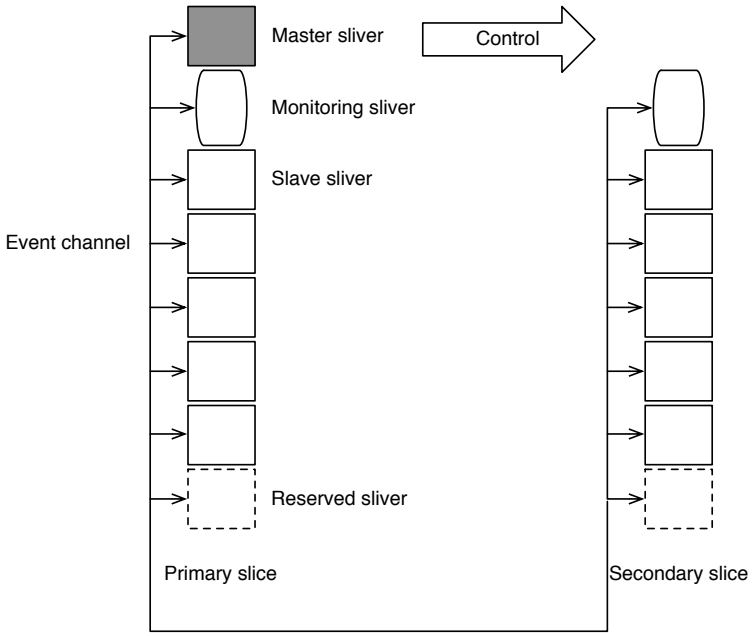

그림 3. 슬라이스의 연결.

4.3. 슬라이스의 유지

안정적인 실험자원 공급을 위해, 일부 슬리버에서 오류가 발생하면 그에 대한 대처방안으로 마스터 슬리버는 문제의 슬레이브 슬리버를 예비 슬리버로 교체한다. 슬리버 교체는 그림 4 와 같이 진행되는데, 사용중인 슬레이브 슬리버의 작업 내용을 예비 슬리버에게 온전히 이전시킨 다음, 이 예비 슬리버를 슬레이브 슬리버로 사용하면서 오류가 발생한 슬레이브 슬리버의 사용을 중단시킨다. 슬레이브 슬리버에 문제가 발생하게 되면, 감시 슬리버는 이를 감지하고 문제의 이벤트를 마스터 슬리버에게 통보한다. 마스터 슬리버는 대체할 예비 슬리버를 결정한다. 작업 내용을 안전하게 이전시키기 위해, 오류가 발생한 슬리버에서 작업 중인 모든 콘텍스트를 예비 슬리버에게 전달하고, 이 예비 슬리버를 다른 슬레이브 슬리버들과 연결시켜서 슬라이스의 형태를 지속적으로 유지한다.

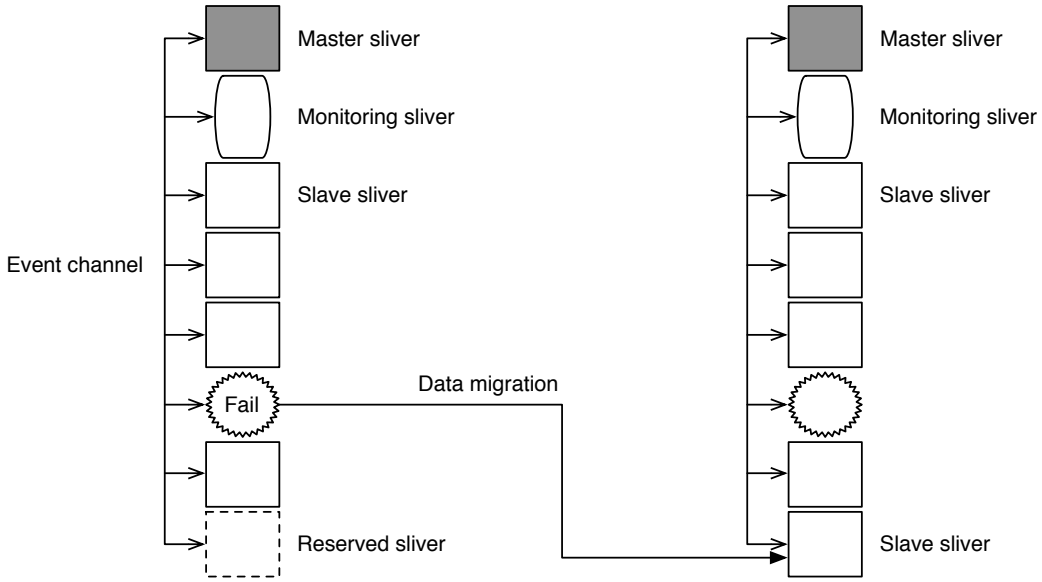

그림 4. 슬리버 오류발생시 예비 슬리버의 활용을 통한 서비스 연속성 보장.

4.4. 슬라이스 상에서의 서비스 운용

주어진 슬라이스를 이용하여 서비스를 운용하기 위한 상위 수준의 실험 제어가 필요하다. 실험 제어는 서비스가 정의하는 서비스의 단위 작업들을 이해하고, 이 단위 작업들을 주어진 종류와 양의 자원들을 공급하는 여러 슬레이브 슬리버들에게 대응시킨다음, 서비스 품질이 적정한 수준에서 유지되는지를 정량적으로 측정한다. 세부적인 과정은 다음과 같다. 세부적인 순서는 다음과 같다.

- 자원 정합 (resource matchmaking): 어떤 슬레이브 슬리버 위에 어떤 서비스 단위 작업들을 대응시킬지 결정한다. 슬리버들은 서로 다른 자원 능력(resource capability)를 가지고 있기 때문에, 서비스 단위 작업들의 요청사항과 슬리버들의 자원 능력을 조화시키는 것이 중요하다.
- 자원 할당 (resource allocation): 서비스 단위 작업들에게 슬레이브 슬리버를 할당하고 서비스 코드를 설치한다.
- 자원 감시 (resource monitoring): 슬라이스의 성능을 감시한다. 슬라이스의 모니터링 슬리버들은 관련된 슬리버들의 성능 변화를 주어진 (또는 상황에 따라 변화된) 시간 간격 마다 측정하여 실험과정에서 계산, 저장, 네트워크 자원의 양이 부족하지 않은지 점검한다.
- 자원 조절 (resource tuning): 슬라이스 또는 슬리버의 자원 종류와 양을 조절한다.

- 자원 회수 (Cleanup): 슬라이스의 사용 기한이 만료되면 서비스 코드를 제거하고 할당된 슬리버들을 회수하고 슬라이스를 종료한다.

그림 5 에서는 준비된 슬라이스 위에서 서비스를 운용하는 사례를 보여준다. 서비스의 시나리오는 다음과 같다. 미디어 서버로부터 획득한 비압축 영상을 단계적으로 DXT (DirectX Texture) 압축을 하면서 다양한 프레임율을 가진 영상 스트리밍 서비스를 제공한다. 이렇게 하는 이유는 한 호스트에서 모든 작업을 병행하기에는 컴퓨팅 비용이 많이 들기 때문이다. 여러 호스트들 위에서 운용되는 슬레이브 슬리버마다 DXT 압축을 위한 서비스 코드를 설치하고, 일부 슬레이브 슬리버에는 캐싱(caching) 코드를 내려서 영상 공급을 하면, 컴퓨팅 부하를 분산시키면서 다양한 품질의 영상 스트리밍 서비스를 할 수 있다.

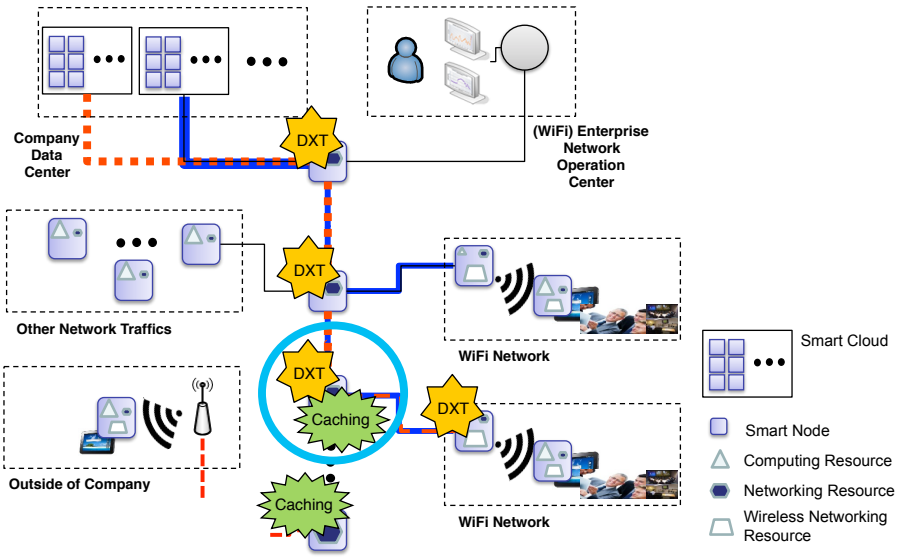

그림 5. 슬라이스 상에서의 서비스 운용 사례.

4.5. 보안 문제들

고려되어야 할 보안 이슈들을 다음과 같다. (1) 먼저 실험자를 인증을 하고, (2) 실험자에게 슬라이스에 대한 접근 권한을 인가하여 필요한 실험 자원들을 추가/삭제한다. 모든 절차는 인증이 먼저 이루어지며, 인증이 완료됨과 동시에 자원들과 서비스들에 대한 접근 권한을 획득하게 된다. 모든 자원과 서비스에 대한 활동들은 주어진 권한에 의해서 접근 제어가 이루어지게 된다. 기존에는 인증과 인가를 위해 IBAC(Identification-Based Acccess Control)을 사용하였다. IBAC 은 ID 를 인증하는 방식인데, 중앙집중식으로 운용되기

때문에 확장성을 제공하지 못한다. 이 문제를 극복하기 위해 통합된 ID 를 관리하는 방식으로 운용되는 FIdM (Federated Identity Management)이 제안되었다. 또 다른 방안으로 역할기반으로 운용되는 RBAC(Role-Based Access Control)이 제안되었다. [14] RBAC 은 소규모 시스템에 적용하기에는 적합하지만, 많은 규칙들을 적용하기에는 한계가 있으며, 다양한 콘텍스트를 지원하지 못한다. 이러한 단점을 극복하고 세밀한 접근 제어를 위해 속성기반으로 운용되는 ABAC (Attribute-Based Access Control) [15, 16] 논의되고 있다. 또한 MS 의 AZURE 에서 운용되는 CBAC (Claim-Based Access Control) [17]과 다양한 인자(factor)에 의한 정의되는 위험(Risk)에 의해 운용되는 RAdAc (Risk Adaptive Access Control) [18]도 함께 논의되고 있다. 그러나 이러한 접근 제어 기법들은 컴퓨팅과 네트워크 자원에 대한 속성의 의미를 동의하기가 어렵다는 이슈가 제기되어, 이를 해결하기 위한 방안으로 인증을 기반으로 운용되는 NBAC (autheNtication-Based Access Control)[19]과 권한에 의해 운용되는 ZBAC (authoriZation-Based Access Control)[20] 등이 제안되었다.

5. 결론

본 문서에서는 미래 인터넷 테스트베드에서 안정적인 실험자원 공급을 위한 요구사항을 도출하고, 슬라이스 기반의 실험자원 관리에 관한 동향과 이슈들을 분석하였다. 슬라이스 기반의 실험자원 공급을 위한 기본 구조와 방법론에 대해서는 정리가 되어가고 있으며, 이를 장기적으로 유지하려는 노력들이 진행되고 있다. 하지만 실험자원에 변화가 생겼을 때, 슬라이스를 안전하게 유지하면서 성능 저하를 막는 연구가 추가로 필요하며, 이 문제는 탄력적인 슬라이스 연구를 통해 해소할 수 있을 것으로 예상된다. 따라서 향후 테스트베드에 대한 연구는 자원 가용성을 높이기 위해 자원의 능력을 정밀하게 이해하면서 서비스 요구사항을 효과적으로 수행할 수 있는 방향으로 개선되기를 기대한다.

6. 참고문헌

[1] 이동훈,박주원,김종원, "실감미디어, 서비스 합성, 프로그래머블 인프라에 근간한 미래인터넷 서비스 프레임워크," 한국정보과학회 논문지, 제 28 권, 제 1 호, pp. 31-40, 2010.

[2] J. Roberts, "The clean-slate approach to future Internet design: A survey of research initiatives," Annals of Telecommunications, vol. 64, no. 5-6, pp. 271-276, 2009.

[3] 신명기, "미래인터넷 기술 및 표준화 동향," 전자통신동향분석, 제 22 권, 제 6 호, pp. 116-128, 2007 년 12 월.

[4] P. Stuckmann, R. Zimmermann, "European research on future Internet design," IEEE Wireless Communications, vol. 16, no. 5, pp. 14-22, Oct. 2009.

[5] GENI Planning Group, "GENI Design Principles," IEEE Computer , vol. 39, no. 9, pp. 102-105, Sep. 2006.

[6] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the FIRE initiative," ACM SIGCOMM Computer Communication Review, vol. 37, no. 3, pp. 89-92, 2007.

[7] T. Aoyama, "A new generation network: Beyond the Internet and NGN," IEEE Communications Magazine, vol. 47, no. 5, pp. 82-87, May 2009.

[8] P. Barford, M. Blodgett, M. Crovella, and J. Sommers, "Requirements and Specifications for the Instrumentation and Measurement Systems for GENI," Tech. Report, May 2009.

[9] R. Dutta, G.N. Rouskas, I. Baldine, A. Bragg, D. Stevenson, "The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet," in Proc. of IEEE International Conference on Communications, pp. 1899-1904, Jun. 2007.

[10] L. Peterson, R. Ricci, A. Falk, and J. Chase, "Slice-based Federation Architecture," 2010, working draft V2.0.

[11] GENI Plastic Slices, http://groups.geni.net/geni/wiki/PlasticSlices

[12] L. Peterson, R. Ricci, A. Falk, and J. Chase, "Slice-based Federation Architecture," 2010, working draft V2.0.

[13] M. Yuen, "GENI in the Cloud," M.S. Thesis, University of Victoria, 2010.

[14] D.F. Ferraiolo and D.R. Kuhn, "Role Based Access Control", in Proc. of National Computer Security Conference, Oct. 1992.

[15] M. Blaze, J. Feigenbaum, J. Ioannidis, “The KeyNote Trust-Management System Version 2”, IETF RFC 2704, Sep. 1999.

[16] A. Pimlott and O. Kiselyov, “Soutei, a Logic-Based Trust-Management System”, in Proc. of Int. Symp. on Functional and Logic Programming. Fuji-Susono, Japan, Apr. 2006.

[17] D. Baier, V. Bertocci, K. Brown, and E. Pace, and M. Woloski, "A Guide to Claims-Based Identity and Access Control," Microsoft, 2010.

[18] Alan H. Karp, Harry Haury, Michael H. Davis, “From ABAC to ZBAC: The Evolution of Access Control Models,” HP Labs, 2009.

[19] J. Li and A. H. Karp, "Access Control for the Services Oriented Architecture", in Proc. of ACM Workshop on Secure Web Services, Nov. 2007.

[20] L. Fang and D. Gannon, “XPOLA - An Extensible Capability-based Authorization Infrastructure for Grids”, in Proc. of PKI R&D Workshop: Multiple Paths to Trust, Apr. 2005