논문 12-37B-10-02

한국통신학회논문지 '12-10 Vol.37B No.10
http://dx.doi.org/10.7840/kics.2012.37B.10.877

연구망에서 가상네트워크 통합제어플랫폼 구현 및 실험

조 일 권◆, 강 선 무°

VIMS: Design and Implementation of Virtual Network Integrated Control and Management Framework over National Research Network

Ilkwon Cho◆, Sun-Moo Kang°

요 약

네트워크 가상화 기술은 NaaS (Network as a Service) 또는 SDN (Software Defined Network)으로 불리는 서비스 지향 아키텍처를 추구하는 미래인터넷에서 중요하게 다루고 있는 연구 분야이다. 네트워크 가상화는 혁신적인 프로토콜들에 대한 상호 독립적인 시험이 가능한 네트워크 테스트베드 구축 기술로서 미래인터넷 연구에서 중요한 역할을 할 것으로 기대하고 있다. 본 논문에서는 단일 도메인의 중소규모 연구망에서 네트워크 가상화를 통해 사용자가 정의하는 토폴로지와 대역폭을 제공하여 다중 사용자의 서비스 트래픽들을 분리, 관리함을 목적으로 하는 제어프레임워크를 제안한다. 본 프레임워크(VIMS; Virtual network Integrated control and Management System)는 이기종의 가상네트워크 장비 제어평면을 수용함으로써 장비에 변경을 요하지 않고 확장할 수 있는 구조를 지닌다. KOREN (Korea advanced REsearch Network)에 적용, 실현 가능성을 확인하였으며 GENI의 제어프레임워크와 비교를 통해 본 프레임워크와의 차이점과 개선을 위한 향후 연구 방향을 도출한다.

Key Words : virtualization, control platform, Future Internet, SDN, NaaS, KOREN

ABSTRACT

Network virtualization technology is a crucial research issue of Future Internet which pursues a service-oriented architecture so-called NaaS (Network as a Service) or SDN (Software Defined Network). Network virtualization is expected to play an important role in Future Internet researches as a network testbed technology which enables innovative protocols to be experimented independently on a common testbed environment. We propose a control framework in order to provide user defined topology and bandwidth services with network virtualization and to separate and manage multiple-user traffics in a small and medium scale - single domain research network. The proposed framework (VIMS; Virtual network Integrated control and Management System) supports testbed expansions without any changes of heterogeneous virtual network support equipments through accommodation of each equipment's control plane. The framework shows a feasibility through applied to KOREN and we describe the differences and further study directions for improvement the framework comparing with GENI control framework.

Ⅰ. 서 론

현재 인터넷은 ARPANET이라는 시험 네트워크에서부터 출발하여 상용 네트워크로 발전, 사회 전반적으로 영향을 미치는 정보통신의 인프라로서의 역할을 하고 있다. 그러나 서비스가 다양해지고 트

※본 연구는 미래네트워크연구시험망구축운영(11-951-00-001) 연구사업 지원으로 수행되었습니다.
◆ 주저자 : 한국정보화진흥원 디지털인프라단, ikcho@nia.or.kr, 정회원
° 교신저자 : 한국정보화진흥원 디지털인프라단, extkang@nia.or.kr, 정회원
논문번호 : KICS2012-06-001, 접수일자 : 2012년 6월 4일, 최종논문접수일자 : 2012년 9월 13일

래픽이 기하급수적으로 늘어나는 상황에서 현 인터넷 아키텍처와 프로토콜에 대한 한계점을 인식하고 이를 근본적으로 해결하려는 논의 및 연구가 진행되고 있다[1]. 현재 논의되는 미래인터넷 아키텍처 및 프로토콜 연구에 대한 분류를 살펴보면 크게 멀티미디어 콘텐츠 전송에 적합한 네트워크 기술, 모바일 환경에 적합한 네트워크 기술, 클라우드 환경에 적합한 네트워크 기술, 보안 이슈들을 해결하는 네트워크 기술로 분류할 수 있으며, 이러한 아키텍처를 구현 실험할 수 있는 테스트베드 기술을 들 수 있다[2].

미래인터넷 구현을 위한 다양한 아키텍처와 프로토콜들의 실현가능성을 확인하기 위해서는 실험 인프라가 필요하다. 다양한 아키텍처의 동시 다발적인 실험을 가능하게 하는 미래인터넷 실험 인프라를 구현하기 위해서는 네트워크 가상화 기술이 필요하다. 네트워크 가상화 기술은 물리적 네트워크 인프라를 공유하며 사용자에게 상호 격리된 논리적 네트워크 파티션을 제공함으로써 다양한 가상네트워크가 동시에 하나의 물리 네트워크 위에서 공존하게 하는 기술이다[3]. 네트워크 가상화를 위한 연구는 가상의 네트워크 인터페이스와 링크단위에서 대역폭을 제공하는 링크 가상화 연구, 라우팅 기능 등 연구자가 원하는 프로토콜을 생성하여 동작시키는 프로그래머블 환경에 대한 연구, 가상 네트워크 간을 상호 연동하여 규모 있는 가상 네트워크를 제공하는 페더레이션에 관한 연구, 그리고 다수의 사용자가 원하는 토폴로지의 가상 네트워크를 구성할 수 있도록 네트워크 자원 들을 할당하고 모니터링 하는 제어 프레임워크에 대한 연구로 구분할 수 있다. 이러한 연구들을 수행하는 대표적인 테스트베드 기술 연구 사례로 GENI(PlanetLab, ProtoGENI, ORCA, ORBIT), OpenFlow, FIRE, JGN-X를 들 수 있다[4-7].

네트워크 제어프레임은 비단 미래인터넷 연구에서 뿐만 아니라 네트워크 트래픽 엔지니어링, 네트워크 프로비저닝, 네트워크 자동화 관점 등에서 오래된 역사를 지닌 연구 분야이다. 최근 미래인터넷 테스트베드 구현에 있어 네트워크 자원 관리와 네트워크 프로비저닝이 더욱 중요해지고 있다. 네트워크 자원 관리 및 네트워크 프로비저닝에 대한 대표적인 연구로는 UCLP(User Controlled LightPath), DRAC(Dynamic Resource Allocation Controller), ProtoGENI, PlanetLab, GEYSERS(Generalized Architecture for Dynamic Infrastructure Services) 등이 있다[8-12]. UCLP는 물리적 자원을 추상화하여 웹 서비스로 구현하여 사용자가 네트워크 물리자원의 세부 사항에 무관하게 전용의 네트워크를 구성할 수 있는 능력을 제공하는 서비스 지향의 네트워크 가상화 프레임워크이다. 주로 파장단위 전송자원 제어를 중심으로 연구가 진행되고 있다. DRAC은 전송매체와 무관하게 단대단(end-to-end) 연결의 프로비저닝을 자동화하여 제공하는 네트워크 제어 프레임워크이다. 주로 광경로 설정 제어를 자동화하는 능력을 제공한다. ProtoGENI, PlanetLab은 선행적인 미래인터넷 아키텍처와 프로토콜을 실험하기 위해 연구망을 기반으로 대규모 테스트베드를 구축하는 GENI(Global Environment for Network Innovations) 프로젝트의 한 클러스터들로서 GENI의 자원들을 제어하기 위한 제어 프레임워크를 연구, 규격화하고 있다. GEYSERS는 GMPLS(Generalized Multi-Protocol Label Switch)와 같은 기존의 NCP(Network Control Platform)를 확장하여 동적으로 가상네트워크를 구성, 제어할 수 있는 참조모델을 제시하고 있다.

국내에서도 GENI, FIRE와 같은 미래인터넷 연구자들이 프로토콜 개발과 시험을 위해 상시적으로 활용하고 관리되어지는 네트워크 테스트베드가 필요할 것으로 보인다. 연구자들은 개발한 프로토콜에 대한 시험을 위해 지협적인 수준에서 자체적인 시험환경을 구성하고 있으며 해당 연구에 종속적인 환경을 지향하고 있다. 더욱이 테스트베드 기술은 선행 연구결과물들을 적용, 구현하여 선행 연구의 동향을 확인하고 연구협력 네트워크를 구성하거나 가상네트워크 원천기술 연구 개발을 위한 활용 차원에서 접근해 왔다. 따라서 불특정 다수의 연구자들에게 개방하며 안정적인 운용 환경을 전제로 다중 사용자를 위한 상시적이고 독립적인 시험 환경 제공을 기대하기 어렵다. 개방적 환경의 가상화 테스트베드 제공을 위한 노력의 일환으로 본 논문에서는 단일 도메인의 중소 규모 연구망에서 새로운 프로토콜들의 상호 독립적인 상시 시험이 가능한 테스트베드 제공을 목적으로 사용자 정의형 토폴로지와 대역폭, 그리고 다중 사용자의 트래픽 분리를 제공하는 가상네트워크 제어프레임워크를 제안하며 이를 구현한 시스템(VIMS; Virtual network Integrated Management System)을 설명한다. 2장에서는 본 프레임워크 연구와 밀접한 관련이 있는 네트워크 제어 프레임워크 기술에 대해 기술한다. 3장에서는 제안한 프레임워크의 설계 및 구현내용을

기술하며 본 프레임워크를 적용한 KOREN의 물리적 네트워크 환경과 VIMS를 통해 가상네트워크를 생성, 네트워크 경로 상에서 트래픽의 독립적 전송을 확인한 실험 결과를 기술한다. 4장에서 가장 대표적인 미래인터넷 테스트베드인 GENI의 제어프레임워크와 설계 원칙, 설계 구조 등의 비교를 통해 본 프레임워크의 한계점과 지향점을 기술한다. 마지막으로 5장에서 본 연구가 갖는 의미와 향후 개선을 위한 연구 방향에 대한 기술로 결론을 맺는다.

Ⅱ. 관련 연구

2.1. Dynamic Circuit Network

DCN(Dynamic Circuit Network)은 사용자 정의형 네트워크 프로비저닝에 대한 대표적인 연구로서 GMPLS(Generalized Multi-Protocol Label Switch)를 근간으로 하여 패킷네트워크에서 동적으로 가상의 회선네트워크를 구성하게 하는 기술이다. 사용자가 원하는 네트워크 대역폭을 할당하고 단대단 링크 가상화를 통해 가상네트워크를 제공한다[13]. DRAGON(Dynamic Resource Allocation in GMPLS Optical networks)은 DCN을 위한 제어 프레임워크를 연구하는 프로젝트로서 단일 도메인과 다중 도메인간의 단대단 GMPLS 전송을 위한 네트워크 구조를 제시하고 있다[14]. 단일 도메인 내에서 라벨(label) 스위칭을 지원하는 LSR(Label Switch Router)로 구성되며 단대단 연결을 위한 제어는 GMPLS 라우팅과 시그널링 프로토콜을 통해 이루어진다. Label은 MPLS와 같은 packet label, SDH/SONET과 같은 TDM label, Wavelength label, Fiber label 들이 있으며, IEEE 802.1D/Q를 이용하며 제어평면에 802.1 계열의 브리지 프로토콜 대신 GMPLS 신호프로토콜을 사용한 Ethernet-GMPLS 기술의 VLAN label이 있다.

DRAGON 구조는 GMPLS 기능을 제공하지 않는 이종의 네트워크 기술과 장비에 VLSR(Virtual Label Switch Router) 개념을 도입하여 non-GMPLS 장비를 GMPLS 도메인으로 통합한다. VLSR은 GMPLS 프로토콜을 이해하고 SNMP 또는 CLI(Command Line Interface)를 통하여 non-GMPLS 장비를 제어한다. VLSR은 형성된 LSP를 통해 LSP 단위의 링크 가상화를 구현한다. GMPLS 다중 도메인 간을 연결 단대단 LSP를 형성하는 기능으로 NARB(Network Aware Resource Broker)를 정의하고 있다. NARB는 한 도메인내의 경로 계산과 도메인간의 경로를 계산하는 기능을 수행한다. 따라서 NARB는 GMPLS 다중도메인 사이의 링크 데이터베이스를 공유하고 전체 토폴로지 정보를 관리한다. 그림 1은 DCN의 VLSR 구조를 나타낸다.

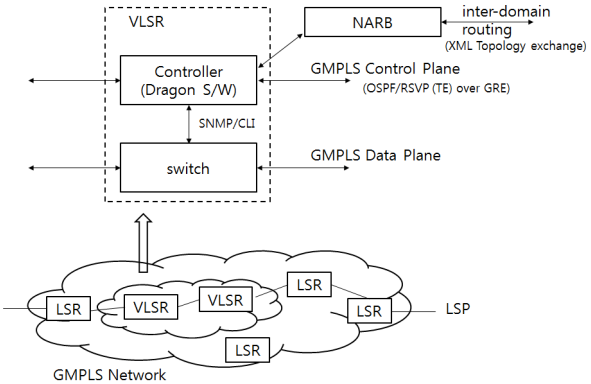


그림 1. DCN의 VLSR 구성도
Fig. 1. Architecture of DCN VLSR

2.2. OpenFlow

OpenFlow는 라우터나 스위치 등의 네트워크 장비의 포워딩 테이블(Forwarding Table)을 직접 제어할 수 있도록 개방형 인터페이스를 표준화하는 기술로 포워딩 테이블을 사용자가 프로그래밍 할 수 있기 때문에 임의의 새로운 아키텍처나 프로토콜을 적용할 수 있다. 오픈 네트워킹 파운데이션(ONF; Open Networking Foundation)에서는 소프트웨어 정의 네트워크(SDN; Software Defined Network) 구현을 위해 OpenFlow 프로토콜 규격을 개발하고 표준화하고 있다[15,16]. OpenFlow 스위치 기술은 플로우 테이블(Flow Table), 보안채널(Secure Channel), OpenFlow 프로토콜로 크게 나눌 수 있다. 플로우 테이블은 각 플로우를 어떻게 처리할지에 대한 정보를 갖고 패킷의 헤더필드와 관련된 액션을 수행한다. 보안채널은 원격의 컨트롤러와 OpenFlow 스위치 사이 간에 OpenFlow 프로토콜로 통신하기 위한 연결을 제공한다. OpenFlow 프로토콜은 플로우 테이블의 엔트리를 컨트롤러가 정의할 수 있도록 스위치와 컨트롤러 간에 정의된 표준화된 통신 인터페이스를 의미한다. OpenFlow 스위치를 제어하기 위한 컨트롤러(Controller)로 네트워크 운영체계(NOX)[17]와 다수 컨트롤러에 의해 가상의 네트워크를 구현할 수 있는 OpenFlow Flowvisor에 대한 연구가 진행되었다[18]. NOX는 OpenFlow 스위치 네트워크를 제어하기 위한 개방형 플랫폼으로

플로우 테이블을 정의하는데 필요한 각종 프로그램 인터페이스를 제공한다[19]. 프로그램인터페이스를 통해 사용자는 새로운 네트워크 아키텍처 및 프로토콜을 구현할 수 있다. Flowvisor는 다중의 컨트롤러와 OpenFlow 스위치 사이에서 프록시(Proxy)처럼 동작하는 특별한 목적의 컨트롤러로서 각각의 컨트롤러가 독립적으로 OpenFlow 네트워크를 구성, 사용할 수 있도록 네트워크 가상화 기능을 제공한다. 그림 2는 OpenFlow 스위치 구조를 나타낸다.

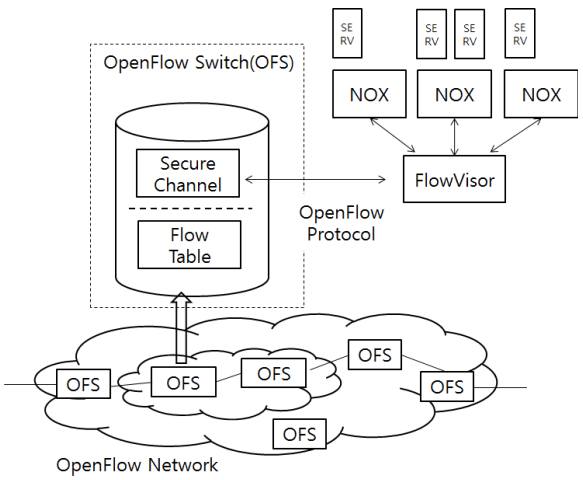


그림 2. OpenFlow스위치 및 네트워크 구성도
Fig. 2. Architecture of OpenFlow switch and network

Ⅲ. 가상네트워크 통합제어 프레임워크

3.1. 제안프레임워크

제안하는 가상네트워크 통합제어 프레임워크는 크게 두 개의 파트로 구성된다. 첫 번째 파트는 개별가상자원 제어 모듈로서 개별 스위치에 대해 가상네트워크를 제어 관리하는 개별 매니저(IM: Individual Manager) 이며 두 번째 파트는 통합가상자원 제어 모듈로서 IM을 제어 관리함으로써 가상네트워크 슬라이스를 구현하는 에그리게이션 매니저(AM: Aggregation Manger) 이다. 본 프레임워크는 웹서비스 구조로 설계되어 웹 화면을 통해 사용자가 가상네트워크를 제어 및 모니터링 하는 기능을 제공하며 이종 가상네트워크 제공 스위치와 이를 제어하는 IM을 변경 없이 수용할 수 있는 구조로 설계되어 중소 규모의 단일 도메인에서 중앙집중형의 체계적인 관리와 확장성을 제공한다.

AM의 핵심기능은 가상자원 통합 컨트롤러(IC: Integrated Controller)로서 IM에 대응하는 웹서비스 클라이언트를 구현하고 있다. IC는 해당 물리네트워크 자원의 IM에 대응하는 웹서비스 클라이언트를 수용할 수 있는 인터페이스 구조로 설계되어 있다. IC는 IM과 XML-RPC(Remote Procedure Call) 통신을 통해 제어 메시지를 교환하는 기능, XML 정보를 해석하는 기능, 사용자가 요구하는 가상네트워크 토폴로지를 생성할 수 있도록 IM을 조합하여 제어하는 기능을 갖는다. AM은 IM에 의해 제어되는 물리네트워크 자원을 확인하는 기능, 가상네트워크 자원을 예약, 수정, 삭제하고 예약된 자원을 모니터링 하는 기능, 그리고 정해진 예약 기간 만료시 일괄 회수하는 자원예약 회수 기능을 갖는다. 또한 AM은 사용자 및 관리자의 권한을 관리하는 정보설정 기능을 갖고 있다. AM은 관리되는 모든 자원에 대한 정보를 데이터베이스화하여 관리한다. 그림 3은 본 논문에서 제안하는 가상네트워크 통합제어 프레임워크를 도식화한 그림이다. IM으로 DCN의 Dragon, OpenFlow의 Flowvisor, 다수의 가상 네트워크 인터페이스(VNIC: Virtual Network Interface Card)를 제공하는 블레이드 서버(UCS: Unified Computing System)의 manager[20]를 예로 보여주고 있다.

3.2. 가상네트워크 통합제어 프레임워크

사용자가 시스템에 로그인해서 원하는 토폴로지의 가상네트워크를 생성하기까지의 사용자와 시스템(VIMS), 관리자간의 주요 작업 흐름을 그림 4와 같이 설계한다. 사용자는 ①VIMS에 로그인을 하고 ②VIMS는 사용자 권한을 체크한다. ③권한에 따라 화면이 분기되며 사용자는 현재 예약하여 사용 중인 자원을 확인하게 된다. ④사용자는 예약을 위해 네트워크 물리자원을 확인, 선정한 후, ⑤선정한 물리 자원을 대상으로 원하는 토폴로지의 가상네트워크를 구성하는 가상자원을 예약한다. ⑥VIMS는 자원예약정보를 데이터베이스에 저장하고 이를 화면에 표출한다. ⑦데이터베이스를 통해 모든 사용자의 자원예약 정보는 관리자가 관리하게 된다. 이후 ⑧-⑩까지는 예약된 자원에 가상인터페이스 ID를 할당하고 사용자 전용의 가상네트워크 생성을 명령하는 것으로 사용자 또는 관리자가 제어하게 된다. ⑪-⑭는 예약정보에 따라 AM이 IM을 통해 가상네트워크를 위한 개별 장비의 리소스를 생성, 활성화시키는 명령이 수행되고 그 정보를 데이터베이스에 저장하는 절차이다. ⑮이후 사용자 및 관리자는 자원 생성 결과를 확인할 수 있다.

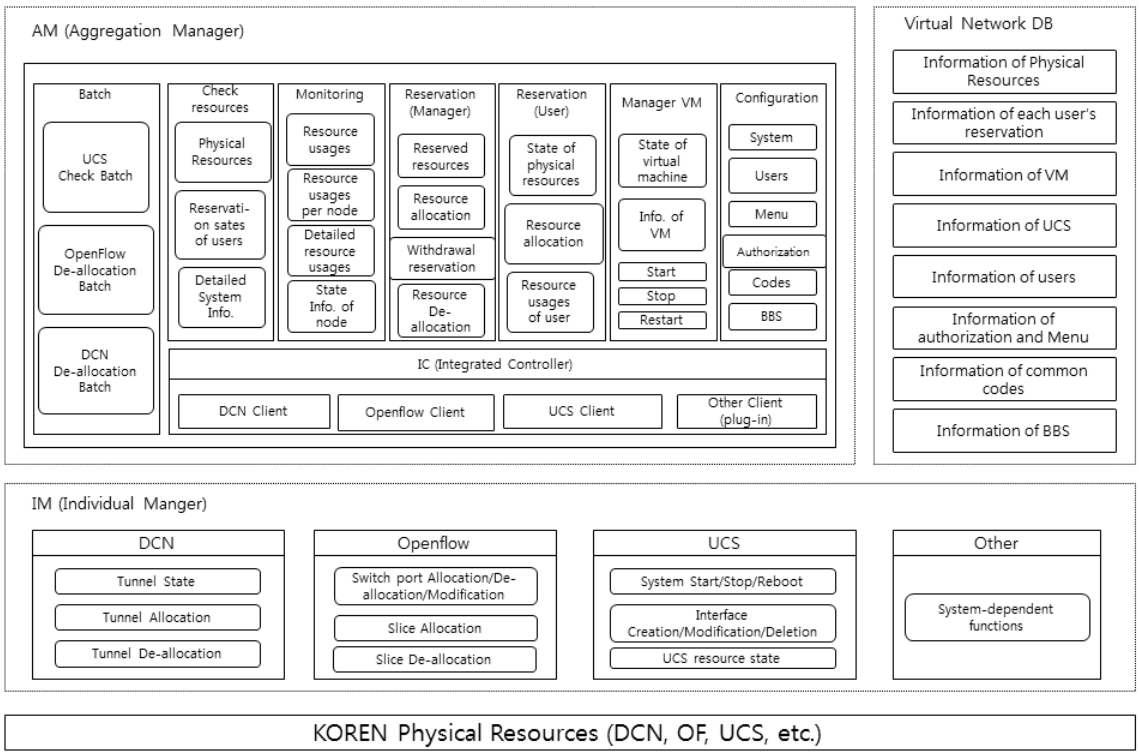


그림 3. 가상네트워크 통합제어 프레임워크
Fig. 3. Framework of virtual network integrated management system

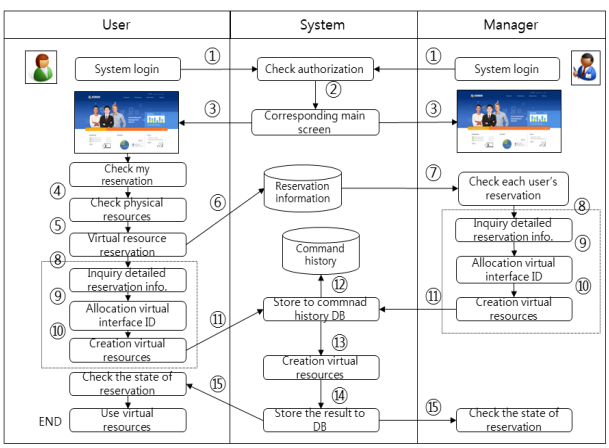


그림 4. 가상네트워크 생성 유즈케이스
Fig. 4. Virtual network creation usecase

3.3. 가상네트워크 제어 기능 설계

다음은 제안한 프레임워크를 형상화한 VIMS의 주요 기능 블록을 설명한다. VIMS의 주요 기능 블록은 그림 5에서 보이는 바와 같이 사용자 및 관리자에 대한 로그인 관리(LoginController), 회원 관리(MemberController), 메뉴 관리(MenuController), 물리 및 가상 자원에 대한 모니터링 관리(MonitoringController), 가상네트워크를 제공하는 물리노드 정보 관리(NodeManageController), 물리링크 관리(Phy_LinkController), 사용자 알림용 뉴스 게시판(NewsController), 가상네트워크 자원예약 관리(ReservationController)와 가상네트워크 토폴로지 맵 관리(ResourceMapController)로 이루어져 있다.

로그인 관리와 회원 관리는 AM 및 IM에 접근하는 사용자의 인증 및 권한을 부여하고 관리를 한다. 메뉴 관리는 자원관리와 모니터링, 그리고 가상네트워크 예약 관리로 분기하는 메뉴를 제공한다. 모니터링 관리는 자원 할당 및 사용량 등 가상네트워크 자원을 사용자에게 보여준다. 물리노드 정보 관리는 IM에 의해 제어되는 자원에 대한 정보 및 상태를 관리하며 물리링크 관리는 노드간 연결된 물리링크에 대한 정보 및 상태를 관리한다. 뉴스 게시판은 관리자가 사용자에게 공지하는 기능을 제공한다. 자원예약 관리는 사용자가 가상 네트워크를 예약할 수 있는 기능을 제공하며 토폴로지 맵 관리는 생성된 가상 네트워크 토폴로지를 맵으로 확인할 수 있도록 한다. 모니터링 관리, 물리노드 정보 관리, 물리링크 관리, 자원예약 관리, 토폴로지 맵 관리는 제안한 프레임워크의 핵심기능인 IC에 포함되는 주요 기능 블록들이다.

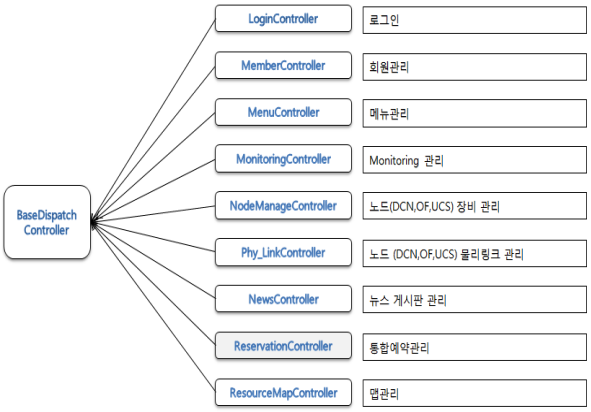


그림 5. 주요 기능 블록도
Fig. 5. Main component block diagram

3.4. 테스트베드

제안한 프레임워크를 구현, 적용한 테스트베드 환경에 대해 설명한다. KOREN은 그림 6과 같이 서울, 수원, 대전, 대구, 광주, 부산의 6개 POP을 중심으로 물리적인 네트워크를 구성하고 있다. 각 노드간에는 Ethernet over ROADM(WDM)으로 링크가 구성되어 있으며 파장당 10Gbps의 논리적 트리 토폴로지로 구성되어 있다. 이와 함께 6개 POP 간에 1Gbps의 논리적 메쉬 토폴로지도 구성되어 있다. VIMS에 의해 제어되는 가상네트워크 제공 장비는 DCN 스위치, OpenFlow 스위치, 그리고 UCS 서버로 구성했다. DCN 스위치는 전국 6개 POP에 구축되어 있으며 OpenFlow 스위치는 NetFPGA NIC을 장착한 소프트웨어 기반의 스위치가 전국 6개 POP, 그리고 하드웨어 기반의 스위치가 서울과 대전에 구축되어 있다. 그러나 안정적인 테스트베드 제공을 위해 소프트웨어 기반의 OpenFlow 스위치는 제어 대상에서 제외한다. UCS는 서울과 대전에 구축하였다.

그림 7은 Ethernet over ROADM 물리링크위에 구현된 네트워크 가상화를 위한 DCN, OpenFlow 네트워크를 보여준다. DCN 네트워크는 Ethernet Vlan을 label로 사용하는 Ethernet-GMPLS로 패스를 형성하며, OpenFlow 네트워크는 OpenFlow 1.0 규격의 Ethernet Vlan 단위로 링크를 가상화하여 패스를 형성한다. DCN, OpenFlow 모두 가용한 물리링크 용량은 1Gbps로 구성되어 있다. UCS는 가상머신 단위의 컴퓨팅 자원을 제공하며 가상머신에게 Ethernet VIC과 Vlan 단위의 링크 가상화를 제공한다. 이들 장비로 조합된 Vlan 단위의 가상화된 링크로 이루어진 가상네트워크를 구성하였다.

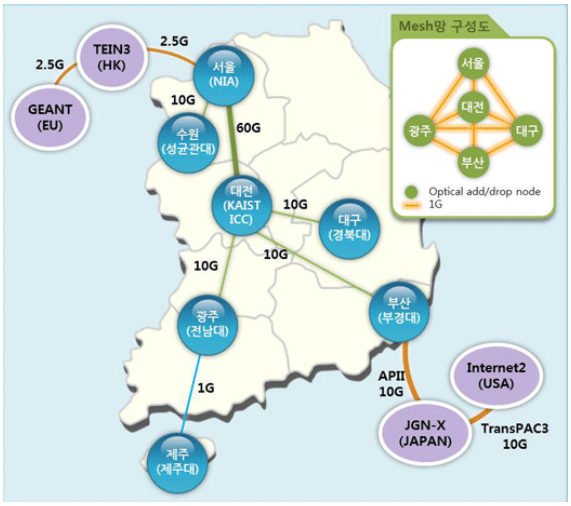


그림 6. KOREN 토폴로지 구성도
Fig. 6. KOREN Topology Map

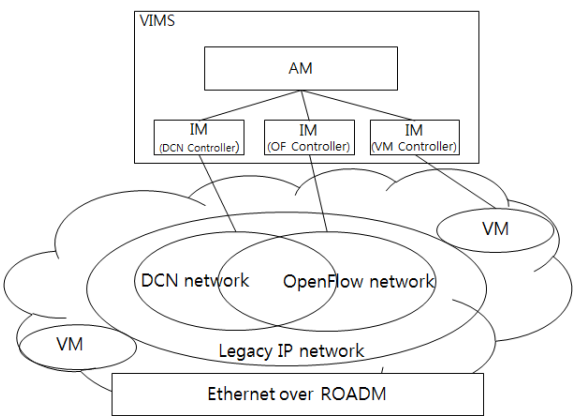


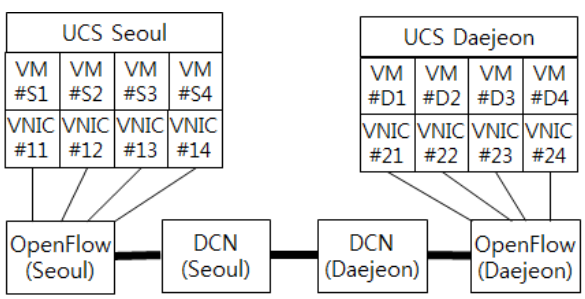
그림 7. 가상네트워크 환경 구성도
Fig. 7. Diagram of virtual network environment on KOREN

3.5. 실험 및 결과

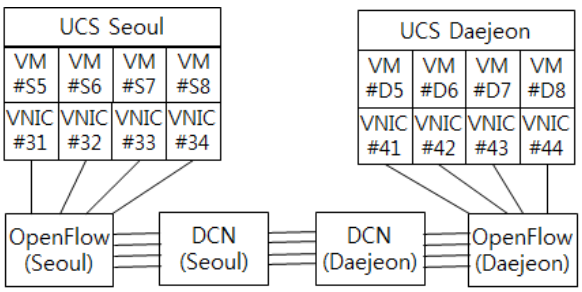
VIMS의 가상네트워크 생성 기능 확인과 생성된 가상네트워크 토폴로지 상에서 동시에 각 가상네트워크에 트래픽을 전송하여 가상네트워크간의 상호 트래픽 간섭현상을 확인하는 실험을 한다. 사용자 특정의 목적에 의해 해당 가상네트워크를 생성, 사용하기 때문에 사용자 상호간 독립적인 네트워크 서비스를 제공할 수 있어야 하며 따라서, 가상네트워크간의 상호 트래픽 간섭이 배제될 수 있어야 한다. 가상네트워크 생성 및 링크 가상화 기능을 실험하기 위해 서울 POP과 대전 POP간에 가상네트워크를 구성한다. 가상네트워크에 해당하는 토폴로지는 UCS(서울) - OpenFlow(서울) - DCN(서울) - DCN(대전) - OpenFlow(대전) - UCS(대전)를 연결하여 구성한다. VIMS가 제어하는 KOREN에서 가용한 모든 자원들을 연결함과 동시에 안정적인 네

트워크 구성을 위해 소프트웨어기반의 OpenFlow 스위치를 배제한 결과 서울-대전간의 단일 토폴로지만으로 테스트환경이 구성되었다. UCS는 가상서버와 가상네트워크를 제공하는 단말장비로서의 역할을 하며 실험에서는 트래픽을 생성 전송받는 서버와 클라이언트 기능을 수행한다. OpenFlow와 DCN은 가상네트워크를 제공하는 스위치 역할로서 UCS에서 생성하는 트래픽을 전달하는 기능을 수행한다.

상호 트래픽 간섭현상을 비교 확인하기 위해 하나의 가상네트워크에 4개의 서로 다른 사용자의 트래픽을 전송, 모니터링 하는 실험을 시나리오 A, 4개의 가상네트워크에 각각 4개의 서로 다른 사용자의 트래픽을 전송, 모니터링 하는 실험을 시나리오 B라 한다. 4개의 트래픽은 flow #1, flow #2, flow #3, flow #4로 표현하며 각 플로우는 가상네트워크의 종단점이 되는 UCS의 각 가상머신 상에서 생성하고 전송받는다. 그림 8과 표 1에 시나리오 A와 B를 정리하였다. 시나리오 A에서 하나의 가상네트워크에 400Mbps를 할당하고 flow #1 ~#4를 전송한다. 시나리오 B에서는 4개의 가상네트워크에 각각 100Mbps를 할당하고 flow #1~#4를 가상네트워크당 하나의 플로우를 각각 할당하여 전송한다.



(a) Scenario A



(b) Scenario B

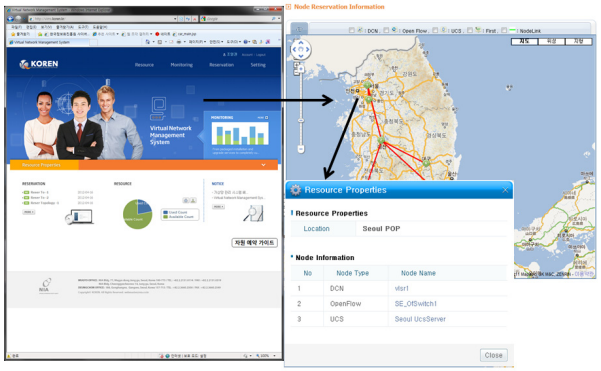
그림 8. 실험 시나리오
Fig. 8. Test scenarios

가상의 네트워크를 할당하고 사용자간 독립적인 네트워크를 제공하고자 제안한 VIMS의 기능을 설명하는 시나리오이다.

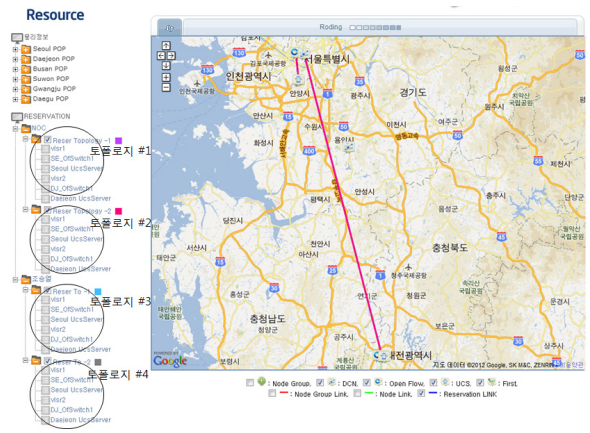
그림 9는 시나리오 B 실험을 시행하기 위해 VIMS를 통해 가상네트워크를 생성하는 모습을 보여준다. 그림 9의 (a)는 VIMS의 로그인 후 보여주는 첫 화면과 서울 POP에 있는 노드 자원의 정보를 확인하는 화면을 보여준다. 로그인 시 사용된 인증정보는 AM과 IM간의 XML-RPC 통신에 사용되어 각 IM을 제어하는데 필요한 인증정보로 활용된다. 첫 화면을 통해 노드의 자원정보 확인, 노드의 자원 사용량 모니터링, 가상네트워크 예약, 그리고 관리자를 위한 메뉴를 제공하는 것을 확인할 수 있다. 그림 9의 (b)는 서울, 대전간 100Mbps 대역폭의 가상네트워크 4개를 생성한 토폴로지 맵을 보여주고 있다. Resource 트리의 토폴로지 #1, #2, #3, #4는 각각 VN #2, VN #3, VN #4, VN #5 가상네트워크를 의미한다.

표 1. 실험 시나리오
Table 1. Test scenarios

시나리오	VN ID	flow ID	End Point #1	End Point #2	Allocated BW
A	VN #1	flow #1	VM #S1-VNIC #11	VM #D1-VNIC #21	400Mbps
		flow #2	VM #S2-VNIC #12	VM #D2-VNIC #22	
		flow #3	VM #S3-VNIC #13	VM #D3-VNIC #23	
		flow #4	VM #S4-VNIC #14	VM #D4-VNIC #24	
B	VN #2	flow #1	VM #S5-VNIC #31	VM #D5-VNIC #41	100 Mbps
	VN #3	flow #2	VM #S6-VNIC #32	VM #D6-VNIC #42	100 Mbps
	VN #4	flow #3	VM #S7-VNIC #33	VM #D7-VNIC #43	100 Mbps
	VN #5	flow #4	VM #S8-VNIC #34	VM #D8-VNIC #44	100 Mbps



(a) Main page and node retrieve



(b) Reserved resources for virtual network and reserved virtual network topology

그림 9. VIMS UI 화면
Fig. 9. User Interface screen of VIMS

End Point #2에서 네트워크의 가용성을 테스트하는 Iperf 플로우를 flow #1 ~ flow #4에 각각 할당하여 동시에 트래픽을 인가한 후 End Point #1에서 트래픽을 모니터링 한다. End Point #2의 각 VNIC은 해당하는 가상머신과 연결되어 있으며 이 가상머신에서 Iperf 플로우를 생성, VNIC을 통해 해당 가상네트워크로 트래픽을 전송한다. End Point #1에서는 해당 가상네트워크의 VNIC과 VNIC에 대응하는 가상머신에서 트래픽을 모니터링 한다. 그림 10은 시나리오 A에서의 트래픽 모니터링 결과를 나타낸다. flow #1과 flow #4는 100Mbps를 크게 상회하며 flow #2와 flow #3은 상대적으로 100Mbps를 크게 하회하고 있다. 플로우의 대역폭 간격도 매우 큰 폭의 변동을 보이고 있어 상호간의 트래픽 간섭이 심하게 나타고 있음을 확인할 수 있다. 이에 반해 그림 11의 시나리오 B는 각 플로우들이 비교적 균일하게 100Mbps 대역폭을 유지하고 있음을 확인할 수 있다. 시나리오 B 실험을 통해 상호 독립적인 가상네트워크 생성 기능을 확인한다.

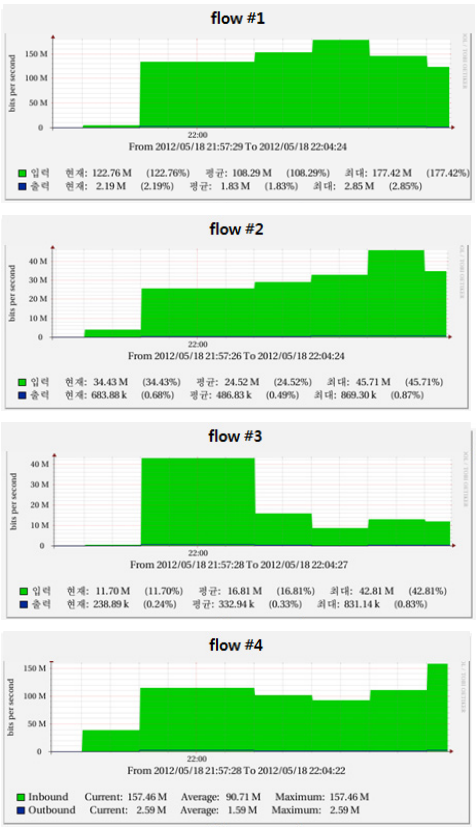


그림 10. 시나리오 A 실험결과
Fig. 10. Test result of scenario A

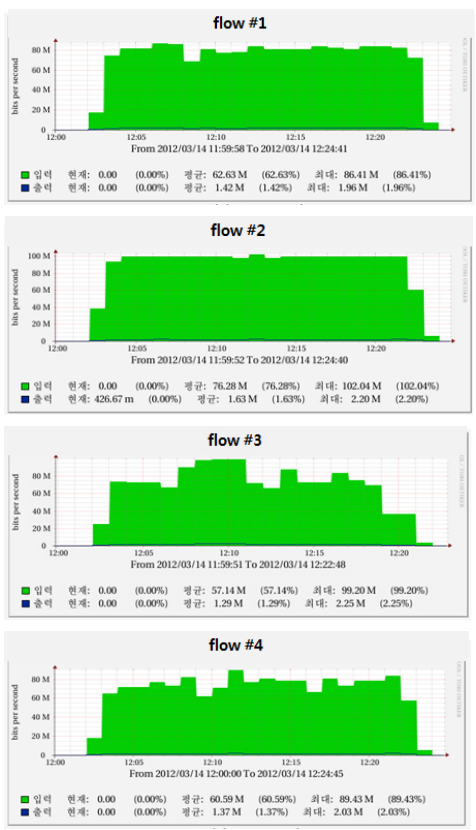


그림 11. 시나리오 B 실험결과
Fig. 11. Test result of scenario B

Ⅳ. GENI 제어 프레임워크의 비교

가장 대표적인 관련 연구인 GENI는 미래인터넷의 새로운 가능성을 확인할 목적으로 규모 있는 네트워크 시험환경을 구축하기 위한 연구프로젝트임과 동시에 연구결과가 반영되어 구축된 테스트베드를 의미한다. GENI에서는 자원공유 및 상호연동을 위해 슬라이스 기반의 네트워크 자원간 페더레이션 아키텍처와 제어 프레임워크(SFA; Slice-based Federation Architecture)를 정의하고 최소한 준수해야할 인터페이스와 데이터 타입을 규격화 하였다[21]. PlanetLab과 ProtoGENI에서는 독자적으로 구현해온 제어프레임워크가 존재하는데 하나의 표준화된 인터페이스로 규격화할 목적으로 GENI Aggregate Manager API[22]를 정의했으며 버전 1.0과 2.0을 거쳐 최근 3.0이 승인되었고 4.0 드래프트 수정작업이 진행되고 있다. 본 절에서는 GENI의 SFA와 GENI AM을 토대로 본 프레임워크와 비교를 통해 제안한 프레임워크의 한계점과 향후 지향할 점을 도출하고자 한다.

4.1. 기본 개념과 설계 원칙

GENI는 혁신적인 미래인터넷 관련 연구를 지원하기 위한 일련의 기반시설을 구축하고 이를 통해 규모 있는 실험환경을 제공한다는 개념 하에 프로그래머블 환경, 페더레이션, 실험을 위한 단대단 가상화된 슬라이스 제공이라는 설계 원칙을 갖고 진행되었다. 이에 반해 본 프레임워크는 단일 도메인, 중소 규모의 국가연구망에서 미래인터넷 연구를 위한 실험 시설을 빠른 시일 내에 제공할 목적으로 가상화된 슬라이스 제공이라는 한 가지의 설계 원칙하에 진행되었다. 실험대상 아키텍처와 프로토콜을 테스트베드에 용이하게 구현하기 위해서는 프로그래머블 환경 제공이 필요하며 규모 있는 테스트베드로 확장과 국가 간 공동연구 협력을 위해서서는 페더레이션 환경 제공이 필요하다.

4.2. 프레임워크 구조

GENI를 구성하는 요소는 슬라이스, 컴포넌트(Component), 컴포넌트 관리자(CM; Component Manager), 복수개의 컴포넌트를 관리하는 AM이 있다. CM과 AM은 컴포넌트가 단수 개인지 복수개인지에 따라 구분되며 따라서 이점을 제외하면 CM과 AM에는 차이가 없다고 볼 수 있다. 이들 요소를 고려하면 GENI는 사용자와 AM, 그리고 AM에 의해 관리되는 컴포넌트의 3계층 구조로 볼 수 있으며 사용자와 AM 사이에 정의된 표준화된 규격의 API가 중요한 역할을 하는 구조로 되어 있다. AM은 컴포넌트 -CPU, 메모리, 디스크, 대역폭과 같은 물리자원, 파일 디스크립터, 포트번호와 같은 논리자원, 그리고 패킷 포워딩 최선 패스와 같은 합성자원- 들을 제어 및 관리하는 기능을 수행한다. 그림 12는 GENI의 프레임워크 구조를 간략히 표현하고 있다[22].

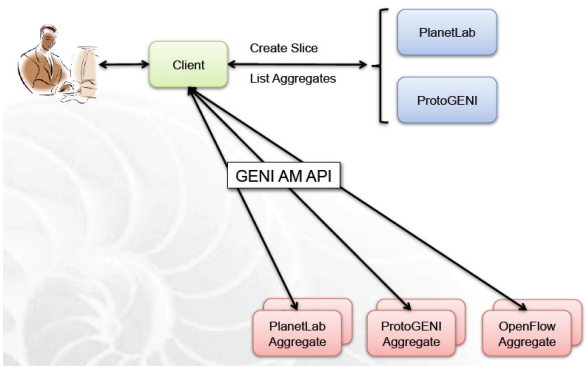


그림 12. PlanetLab과 ProtoGENI의 GENI AM API
Fig. 12. Using the GENI AM API with PlanetLab and ProtoGENI

이에 반해 본 프레임워크는 그림 3에서와 같이 사용자-AM-IM-컴포넌트의 4계층 구조로 되어 있다. 컴포넌트와 IM은 규격화 대상의 범위가 아니며 따라서 컴포넌트와 IM의 자율성을 보장한다. AM은 IM을 수용하여 확장할 수 있는 구조로 설계됐다. 이는 컴포넌트와 IM의 변경 없이 빠른 배치를 가능하게 해준다. 이에 따른 제약은 IM 기능이 없는 컴포넌트를 AM에서 직접 수용할 수 없으며 AM이 IM을 수용하기 위한 스터브(Stub)의 일종으로 해당 IM과 통신을 위한 웹서비스 클라이언트를 구현해야 한다. 웹서비스 클라이언트 구현 내용은 IM이 공개한 API를 XML-RPC를 통해 호출하고 이를 AM의 IC와 정합하는 일이다. 컴포넌트와 IM 세트의 다양성이 커질수록 본 시스템의 복잡도는 증가할 것이나, 단일 도메인에서의 다양성은 한계관리가 가능하다는 전제하에 안정성이 확인된 컴포넌트와 IM 세트의 빠른 수용과 배치에 우선을 두어 설계하였다. AM의 API를 별도로 외부에 공개하지 않았으며 API 호출방법을 정의하지 않은 상태로 사용자는 웹브라우저를 통해 HTTP 프로토콜로 AM에서 제공하는 기능을 사용하도록 한정되어 있다. 컴포넌트의 다양성에 의해 복잡성이 증가할 경우 IM의 API를

규격화하는 것을 고려할 수 있으나 IM과 컴포넌트 개발자와 연구협력이 전제되어야 할 것으로 보인다.

4.3. 식별체계

GENI에서는 기반시설의 소유자(Owners), 기반시설의 운용자(Operators), 기반시설을 사용하는 연구자(Researchers), 그리고 다른 개체를 확인해주는 식별 앵커(Identity Anchors)의 네 개의 개체를 정의하고 있다. 이들 개체와 컴포넌트, 슬라이스, 서비스 등에 대해 인증서 기반의 글로벌 식별자(GID; Global Identifiers)를 부여하고 이 식별자를 통해서 상호 인증 및 통신을 제공한다. GENI는 이해관계가 다양한 관계자들이 참여하는 대규모 테스트베드로서 구성 요소들과 이해관계자들 사이의 관계를 고려하여 식별자 및 접근제어를 설계하고 있다. 본 프레임워크에서는 개체로서는 운용자와 사용자만을 구분하고 있으며 이들에게 식별자를 부여하고 접근제어를 하고 있다. 컴포넌트와 슬라이스, 서비스 등에 고유한 식별자를 부여하고 있으며 식별자 정보들은 데이터베이스 시스템을 통해 관리하고 있다.

4.4. 데이터타입

GENI에서는 자원을 기술하는 표현인 리소스규격(RSpec), 요청한 자원의 할당을 승인하는 티켓(Ticket), 권한을 인지하는 크레덴셜(credential)의 세 가지 데이터타입을 규정하고 있다. RSpec의 경우 PlanetLab과 ProtoGENI 모두 XML기반의 포맷을 사용하고 있으며, 본 프레임워크에서도 XML기반의 포맷을 사용하고 있다. SFA에서의 티켓은 AM이 RSpec에 서명을 한 것으로 자원 예약을 확인하는 역할을 한다. 대규모의 분산된 시설에 대해 많은 사용자의 예약이 진행되며 예약과 실제 사용까지 많은 시간이 소요될 것을 전제하기 때문에 티켓을 사용한다. 또한 슬라이스가 생성되고 소멸될 때까지의 슬라이스 전 생명주기(Life cycle) 동안 크레덴셜에 기반을 둔 사용자 인증이 개별 기능 수행 시 마다 이루어진다. 본 프레임워크는 예약과 자원할당을 거의 동시점에 처리하는 프로세스로 설계가 되어 별도의 티켓을 발부, 관리하지 않고 슬라이스ID 식별자로 처리한다. 크레덴셜의 경우 사용자 인증 정보를 바탕으로 X.509 인증서를 생성을 하는 기능은 구현되어 있으나 시스템 운용에 적용하지 않고 있다.

4.5. 유즈케이스

GENI에서의 생명주기는 (1)클라이언트로 클리어링하우스(CH)에 접속 인증을 하고 빈 슬라이스를 생성한다. (2)AM의 목록들을 조회하고 원하는 AM에 접속한다. (3)AM에서 제공하는 자원을 조회하고 원하는 자원 할당을 요청한다. (4)예약된 슬리버는 슬라이스에 할당이 된다. 설계 특성상 예약과 자원 할당 시간 사이에 가용자원의 상태 변화가 있을 수 있으며 따라서 요청과 할당된 결과에는 차이가 있을 수 있다. 본 프레임워크는 3.2절에 유즈케이스를 설명하고 있다. 차이점은 다음과 같다. 별도의 CH를 운용하고 있지 않고 AM에서 인증을 처리한다. AM은 본 시스템이 유일하여 목록을 제공하지 않는다. 자원 요청내용과 할당결과에는 차이가 있지 않다. 다만 요청한 만큼 할당을 할 수 없을 경우 슬라이스가 생성되지 않으며 빈 슬라이스는 존재할 수 없으며 요청한 모든 슬리버가 할당이 되어야 슬라이스가 생성된다는 점이다.

4.6. 페더레이션

GENI의 SFA는 페더레이션을 전제로 하고 있다. AM API를 통해 AM 상호간, 클라이언트와 AM간, CH와 AM간을 페더레이션 하여 대규모 분산 구조의 테스트베드를 구현하고 있다. 본 프레임워크는 AM이 IM을 수용하면서 중앙관리가 가능한 수준의 중소규모 테스트베드 구현을 기본 설계원칙으로 하고 있다. AM의 API와 접근 방법을 별도로 공개하지 않은 상태이다. 규모 있는 테스트베드로 확장하기 위해서는 많은 컴포넌트와 IM을 확보하고 GENI, FIRE 등과 같은 대규모 테스트베드와 페더레이션이 필요하다. PlanetLab과 ProtoGENI가 독자적인 규격으로 구현을 진행한 상태에서 SFA를 준수하여 상호 페더레이션이 가능하도록 구현했던 것과 같이 본 프레임워크에서도 API를 수정하거나 정합하는 것을 전제로 GENI와의 페더레이션은 가능하다. 이 경우 본 프레임워크의 API가 GENI 규격으로 공개됨과 동시에 네트워크 프로비저닝에 대한 프로그래머블 환경을 제공하게 된다. 더욱이 최근 OpenFlow의 GENI 수용과 같이 IM 들이 GENI의 API를 준수할 경우 본 프레임워크는 별도의 스터브를 구현할 필요 없이 KOREN 도메인 내에서 GENI 규격의 AM들을 하나로 묶은 AM으로 구현될 수 있다.

Ⅴ. 결 론 및 향후 연구방향

본 논문에서는 단일 도메인의 중소 규모 연구망

에서 새로운 프로토콜들의 상호 독립적인 상시 시험이 가능한 테스트베드 제공을 목적으로 하는 사용자 정의형 토폴로지와 대역폭, 그리고 다중 사용자의 트래픽 분리를 제공하는 가상네트워크 제어프레임워크를 제안하였다. KOREN(Korea advanced REsearch Network) 연구망에 적용한 실험 결과를 제시하여 실제 네트워크상에서 실현가능성을 확인하였다. 컴포넌트 배치의 제약으로 인해 다양한 토폴로지를 물리적으로 구현하지 못한 상황에서 서울-대전 두 POP간의 단일 토폴로지에서 수개의 동일한 가상의 네트워크 토폴로지를 구현하고 독립적인 트래픽 전송을 확인하였다. 또한 대표적인 가상네트워크 제어프레임워크인 GENI의 프레임워크와 비교를 통해 기본 개념과 설계 원칙, 프레임워크 구조, 식별체계, 데이터타입, 유즈케이스, 페더레이션 측면에서 본 프레임워크가 갖는 차이점과 향후 지향점을 확인하였다.

본 연구는 하나의 물리적 네트워크 테스트베드가 네트워크 가상화와 가상화 통합 제어를 통해 개별적인 연구만을 위한 각각의 맞춤형 네트워크를 제공할 수 있는 기능의 실현 가능성을 확인함과 컴포넌트 배치 제약으로 단순한 토폴로지에 국한되긴 하나 국내 연구망에 적용, 운용하여 상시적 테스트베드를 제공하는 인프라를 구현했다는 점에서 의미를 둔다. 향후에는 테스트베드 규모의 확장을 위해 타 테스트베드와의 페더레이션 기능 구현과 새로운 아키텍처와 프로토콜 그리고 이를 응용한 서비스들을 개발하고 동작시킬 수 있는 표준화된 인터페이스의 런타임 개발 환경을 제공하는 연구가 필요할 것으로 보인다. 이를 통해 규모 있는 네트워크 테스트베드에서 네트워크 프로비저닝뿐만 아니라 개발한 프로토콜 및 서비스 등을 로딩할 수 있는 프로그래머블 실험환경을 제공할 수 있을 것으로 기대한다.

참 고 문 헌

[1] A. Feldmann, "Internet clean-slate design : what and why?," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59-64, July 2007.

[2] Jianli Pan, Subharthi Paul, and Raj Jain, "A survey of the research on future internet architectures," *IEEE Communication Magazine*, vol. 49, no. 7, pp. 26-36, July 2011.

[3] Chowdhury, N.M.M.K, and Boutab R, "Network virtualization: state of the art and research challenges," *IEEE Communication Magazine*, vol. 47, no. 7, pp. 20-26, July 2009.

[4] Global Environment for Network Innovations (GENI) Project, July 2 2012, http://www.geni.net/.

[5] OpenFlow, July 2 2012, http://www.openflowswitch.org/.

[6] FIRE: Future Internet Research and Experimentation, July 2 2012, http://cordis.europa.eu/fp7/ict/fire/.

[7] New Generation Network Testbed JGN-X, July 2 2012, http://www.jgn.nict.go.jp/english/index.html.

[8] Junyent, G., Figuerola, S., Lopez, A., and Savoie, M., "UCLPv2: a network virtualization framework built on web services," *IEEE Communication Magazine*, vol. 46, no. 3, pp. 126-134, Mar. 2008.

[9] OpenDRAC (the Open Dynamic Resource Allocation Controller), July 2 2012. https://www.opendrac.org/.

[10] ProtoGENI, July 2 2012, http://www.protogeni.net/trac/protogeni/.

[11] PlanetLab (An open platform for developing, deploying, and accessing planetary-scale services), July 2 2012, http://www.planet-lab.org/.

[12] GEYSERS (Generalized Architecture for Dynamic Infrastructure Services), July 2 2012, http://www.geysers.eu/.

[13] Chin P. Guok, David W. Robertson, Evangelos Chaniotakis, Mary R. Thompson, William Johnston, and Brian Tierney, "A user driven dynamic circuit network implementation," in *Proc. GLOBECOM workshops 2008*, pp. 1-5, New Orleans, US, Dec. 2008.

[14] Tom Lehman, Jerry Sobieski, Bijan Jabbari, "DRAGON: A framework for service provisioning in heterogeneous grid networks," *IEEE Communication Magazine*, vol. 44, no. 3, pp. 84-90, Mar. 2006.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, "Open flow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74, Apr. 2008.

[16] Open Networking Foundation, July 2 2012, https://www.opennetworking.org/.

[17] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communications Review*, vol. 38, no. 3, pp. 105-110, July 2008.

[18] R. Sherwood, G. Gibb, K.K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, *FlowVisor: A Network Virtualization Layer (OPENFLOW-TR-2009-1)*, Retrieved July 2, 2012, from http://www.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf.

[19] Sungho Shin, Namgon Kim, JongWon Kim, "Design and Implementation of an OpenFlow-based Flow Control and Monitoring Tool for Programmable Networking Experiments," *Journal of KIISE : Information Networking*, vol. 38, no.1, pp. 11-21, Feb. 2011. (in Korean)

[20] *Cisco UCS Manager Architecture(2012)*, Retrieved July 2, 2012, from http://www.cisco.com/en/US/prod/collateral/ps10265/ps10281/white_paper_c11-525344.pdf.

[21] Larry Peterson, Robert Ricci, Aaron Falk, Jeff Chase, *Slice-Based Federation Architecture Version 2.0(2010)*, Retrieved July 2 2012, from http://groups.geni.net/geni/wiki/SliceFedArch/SFA2.0.pdf.

[22] Tom Mitchell, "GENI aggregate manager API," *Geni Engineering Conference 2010 (8th GEC)*, San Diego, US, July 2010.

조 일 권 (Ilkwon Cho)

1996년 한양대학교 전자공학졸업
1998년 한양대학교 전자공학 석사
2010년 큐슈대학교 정보공학 공학박사
정보통신기술사, 정보시스템감리사
1998～ 2001년 LG전선 통신연구소 주임연구원
2001～2003년 (주)링네트, (주)포위즈 선임연구원
2003～현재 한국정보화진흥원 책임연구원, KOREN팀장
<관심분야> SDN, Mobility, Femtocell, QoS 등

강 선 무 (Sun-Moo Kang)

1983년 충남대학교 전자공학과 졸업
1987년 스톡홀름 왕립공대 전송이론과 석사
1998년 충남대학교 전자공학과 공학박사
1983～2000년 한국전자통신연구원 팀장, 책임연구원
1984～1987년 스웨덴 L.M.Ericsson 객원연구원
2000～2004년 (주)네오텔레콤 부사장
2004～2006년 한국무선국사업관리단(현 한국방송통신전파진흥원) 연구위원
2006～현재 한국정보화진흥원 디지털인프라 단장
<관심분야> SOA, EA, IP-USN, 미래인터넷, 차세대네트워크, 클라우드 컴퓨팅 등