GENI 제어 프레임워크 연구 개발 동향

Technology and Trends of GENI Control Framework

남기혁 (K.H. Nam) u-인프라표준연구팀 연구원
정상진 (S.J. Jeong) u-인프라표준연구팀 선임연구원
신명기 (M.K. Shin) u-인프라표준연구팀 책임연구원
김형준 (H.J. Kim) u-인프라표준연구팀 팀장

목 차

* 본 논문은 2010-P1-12 네트워크 가상화 기술 표준개발 사업의 지원을 받아 작성되었음.

GENI 인프라스트럭처의 기본 구조와 연구 개발 현황을 GENI 표준 규격 문서와 세부 구현 결과를 중심으로 정리한다. 최근 발표된 SFA 2.0과 GENI Aggregate Manager API 1.0을 참고하여 GENI 전체 시스템 구조와 이기종 플랫폼의 상호 연동을 위한 인터페이스를 살펴보고, 이러한 규격에 대해 GENI에서 제시하는 참조 구현(GCF, Omni), ProtoGENI의 AM/CM API 및 참조 구현인 ReferenceCM와 관련 도구도 간략히 정리한다. 마지막으로 최근 활발히 진행되는 페더레이션 관련 연구 동향도 살펴보고 향후 계획을 전망해 본다.

I. 개요

GENI는 네트워크 연구를 위한 NSF 연구 프로그램으로서 네트워크 이론부터 분산 시스템, 네트워크 정책 및 사회/경제적 측면의 연구 등을 포함한 다양한 학제간 연구 성격을 띠고 있다[1],[2]. GENI는 철저한 엔지니어링 성격의 프로젝트로서 PlanetLab, ProtoGENI, VINI 등의 여러 프로젝트를 통해 도출된 결과를 토대로 GENI 인프라스트럭처의 골격을 정의해 나가고 있다. 또한 GENI는 연구 프로그램으로부터 도출된 결과물인 인프라스트럭처의 명칭이기도 하며, 이후로 본 문서에서 언급한 GENI는 인프라스트럭처를 의미한다.

GENI는 Programmability, Resource Sharing (Virtualization), Federation, Slice-Based Experiment라는 네 가지 핵심 개념[1]을 토대로 실제 시스템을 구현해나가는 나선형 개발 모델로 진행되고 있다. 따라서 최근 발표된 SFA와 AM API도 이러한 원칙과 구현 결과를 반영하여, 여러 기관이 페더레이션 된 형태로 제공되는 자원으로 슬라이스라는 추상화된 네트워크를 구성하는 메커니즘을 구현하고 있다.

본 논문에서는 GENI 인프라스트럭처의 아키텍처인 SFA를 바탕으로, 자원 공유 및 페더레이션을 위한 컨트롤 프레임워크의 핵심 구성요소인 Aggregate Manager에 대한 최신 규격을 소개하고, ProtoGENI를 비롯한 관련 연구 개발 동향도 함께 살펴본다.

II. GENI 기술 동향

1. 기술 규격

GENI는 앞에서 언급한 네 가지 핵심 개념을 구현하여 독립적으로 운영되는 이기종 플랫폼의 생태계를 구성하기 위하여, 슬라이스 기반 페더레이션 아키텍처(이하 SFA)를 정의하여 GENI 인프라스트럭처의 구성과 페더레이션 된 다양한 플랫폼으로부터 자원을 공유하고 슬라이스를 운영하는 기본 골격을 제시하고 있다. 참고로 2010년 7월 2.0 버전에 들어서 SFA의 명칭이 Slice-Based Facility Architecture에서 PlanetLab, ProtoGENI(Emulab), OpenFlow가 상호 연동되는 페더레이션 개념이 강조된 Slice(-Based) Federation Architecture로 변경됐다[3]. 현재의 PlanetLab 구현은 이러한 SFA 개념이 상당부분 반영되어 있다[4].

GENI의 결과물은 네 가지 핵심 원칙으로부터 도출된 아키텍처인 SFA, GENI를 구성하는 핵심 구성요소인 Aggregate Manager(이하 AM)와 이를 지원하기 위한 API, GENI 컨트롤 프레임워크 구현의 참조를 위한 GCF 등과 같이 top-down 방식으로 구성되어 있지만, 개요에서 언급한 바와 같이 철저한 엔지니어링 프로젝트 성격을 강조하는 GENI의 특성상 PlanetLab과 ProtoGENI 등의 세부 프로젝트를 통해 실제 구현 및 검증된 결과가 규격에 반영되는 측면이 강하다. 그 중에서도 SFA는 전체 GENI 인프라스트럭처를 위한 컨트롤 프레임워크의 핵심 골격을 정의한 것이며, 여기서 명시된 사항에 대한 최신 이슈나 동향은 ProtoGENI나 PlanetLab, ORCA와 같은 세부 프로젝트 진행을 통해 확인할 수 있다[5]-[8].

2. 핵심 개념

SFA에서는 GENI 인프라스트럭처에 대해 다음과 같은 두 개의 추상화 개념으로 구성된다.

가. 슬라이스(Slice)

GENI로부터 제공되는 컴퓨팅 및 통신 자원으로

구성된 하나의 실험망 또는 서비스 네트워크로서, 실제 물리 자원으로부터 도출된 가상 자원 또는 격리된 물리 자원인 슬리버(sliver)의 집합으로 구성된다. 슬라이스는 슬리버뿐만 아니라 다양한 사용자와 권한과 연계되며, 슬라이스 자체의 라이프사이클을 갖고 있다.

나. 컴포넌트, Aggregate

컴포넌트는 에지 컴퓨터, 라우터, 프로그래머블 액세스 포인트 등에 해당하는 GENI 인프라스트럭처의 기본 구성 요소로서, CPU나 메모리와 같은 물리 자원과 포트번호나 파일 디스크립터와 같은 논리 자원, 그리고 두 형태가 결합된 패킷 포워딩 패스와 같은 자원 등으로 구성되며, 특정한 자원이 속한 컴포넌트는 최대 한 개만 존재할 수 있다. 컴포넌트는 컴포넌트 매니저(CM)에서 관리한다. 여러 개의 컴포넌트가 하나의 그룹으로 묶은 것을 Aggregate라 하고, 마찬가지로 Aggregate Manager(AM)가 관리하며 복수 개의 컴포넌트를 관리하는 점을 제외하면 컴포넌트와 유사하다. 따라서 CM은 소속 컴포넌트가 1개인 AM이며, 본 문서에서는 AM으로 통칭한다. AM은 다른 기관 및 사용자에서 접근하기 위한 표준 인터페이스를 제공하며, GENI의 슬라이스 관리, 자원 공유, 그리고 페더레이션의 핵심 기능을 수행한다. 특히 AM에서 제공하는 자원을 가상화하거나 격리된 물리 자원 형태인 슬리버 단위로 제공하는 기능을 담당한다.

3. 역할 및 권한

SFA에서는 GENI를 사용하는 역할(principal)을 다음과 같이 세 가지로 구분한다.

- MA: AM의 정상적인 운영을 위한 권한으로서, AM에 적용된 정책에 맞게 자원이 할당되고 동작하는지를 관리한다.
- SA: 슬라이스 생성 및 사용 권한으로서, 슬라이스 등록과 슬라이스 사용자 접근 제어 등을 담당한다.
- 사용자: GENI를 통해 새로운 네트워크 모델을 실험하거나 원하는 서비스를 실행하는 연구자와 GENI 인프라스트럭처 전체 혹은 특정 AM을 관리하는 시스템 관리자, 관련 프로젝트 책임자(PI), GENI 페더레이션에 참여한 물리 자원 제공자 등이 모두 사용자에 해당한다. 또한 슬라이스를 통해 특정 실험 및 서비스를 제공할 때, 또 다른 계층의 사용자가 존재할 수 있으나, SFA 차원에서 별도의 규정을 두고 있지 않고 해당 슬라이스에서 자체적으로 해결하는 것으로 간주한다.

GENI의 접근 제어 방식도 위와 같은 분류에 따라 정해진다. SA의 경우, 슬라이스의 인스턴스를 생성하고, 특정 AM 및 자원과 바인딩하고, 슬라이스를 제어하기 위한 권한으로 세분화 될 수 있으며, 제3자 서비스에게 위임할 수도 있다. 모든 사용자는 기본적으로 슬라이스와 컴포넌트의 정보를 조회하는 권한(info privilege)을 갖는다. 또한 MA는 자신이 관리하는 AM에 대하여 관리할 수 있는 권한(operator/control privilege)을 갖는다.

SFA의 구조 특성상 사용자와 슬라이스, 개별 AM의 관계를 고려하여 다양한 각도로 접근 제어를 해야 하므로, 현재 구체적인 방법에 대한 논의가 진행되고 있으며, ABAC 프레임워크를 도입하는 방안도 제시되고 있다[3].

4. 식별 체계

SFA에서는 정확하고 안전한 시스템 구성을 위해

GENI를 구성하는 컴포넌트, 슬라이스, 서비스, 역할(MA나 SA 등)과 같은 모든 대상에 대해 GID를 정의하고 있다. GID는 단순한 식별자의 기능을 넘어 일종의 인증서로서 사용되도록 다음과 같이 세 부분으로 구성된다.

GID = (공개키, UUID, 유효기간)

먼저 공개키는 인증에 사용되며, GID로 식별된 대상은 여기에 지정된 공개키에 대응되는 개인키를 가진다. UUID는 GUID로도 불리는 RFC 4122에 정의된 포맷으로, 오브젝트에 부여된 UUID는 공개키가 바뀌어도 변경할 수 없으며 SFA 기반 전체 시스템에 고유한 식별자로 사용된다. 그러나 현재 PlanetLab과 ProtoGENI에서는 UUID를 실질적으로 사용하지 않고 있다. 마지막으로 유효기간은 말그대로 GID의 유효기간을 명시하고, 만료 전에 갱신해야 한다.

5. 데이터 타입

SFA는 RSpec, Ticket, Credential 등의 세 가지 데이터 타입을 제시하고 있다.

RSpec은 일차적으로 AM에서 제공하는 자원 현황을 표현하고 이러한 자원을 요청할 때 사용되는 포맷으로서, 자원 현황 및 요청 대상에 대한 항목과 더불어, 요청을 시작한 시각과 유효 기간을 나타내는 항목도 함께 담고 있다. RSpec은 SFA 기반 시스템 구현에 큰 비중을 차지하고 있으며, 현재도 SFA 차원에서 구체적인 포맷으로 통일되어 있지 않으며, 앞으로 심도있는 논의와 개정 작업이 예상된다. 참고로 PlanetLab과 ProtoGENI에서는 XML 기반의 포맷을, ORCA에서는 RDF 기반의 RSpec을 사용하고 있으며[8], ProtoGENI 관련 자료에 현재 구현에 사용되고 제안된 RSpec 포맷의 예를 확인할 수 있다[9].

티켓(ticket)은 AM이 서명한 RSpec으로서, 사용자가 요청한 자원을 할당해주겠다는 증서의 역할을 한다. 따라서 AM이 발급(issue)한 티켓은 요청한 자원이 할당된 후에 회수(redeem)된다. 티켓은 RSpec과 더불어 할당한 자원에 대한 권한을 갖는 대상에 대한 GID와 티켓이 고유한지를 나타내는 숫자로 구성된다. GENI는 분산 시스템 형태로 구현되기 때문에, 원격에 있는 특정 AM에 대한 요청이 반영되어 슬라이스에 자원이 할당되어 가용한 상태로 되기까지 시간이 걸릴 뿐만 아니라, 요청 시점과 할당 시점의 자원 및 AM 가용 현황이 얼마든지 변경될 수 있기 때문에, 오류없이 제대로 할당되었다는 확인 과정도 필요하며, 이러한 과정에 티켓이 중요한 역할을 갖는다.

Credential은 특정한 역할에 대한 권한을 담은 데이터로서, 사용자가 슬라이스를 생성하고 특정한 AM으로부터 자원을 할당받아 운영하려면, 이 과정에 발생하는 모든 연산에 대해 이러한 권한을 담은 credential을 제공해야 한다. 구체적인 포맷은 SFA 및 AM API 규격과 앞서 3절에서 언급한 ABAC와 같은 접근 제어 프레임워크의 종류에 따라 결정된다.

6. 인터페이스

SFA에서는 2절에서 언급한 핵심 개념인 슬라이스와 컴포넌트에 대해 상호 연동 및 구현을 위한 API 형태의 인터페이스를 정의하고 있다. SFA가 GENI 전체의 페더레이션을 중심으로 추상화 단계의 개념을 정의한 만큼, API에 대한 실제 구현은 PlanetLab이나 ProtoGENI를 통해 확인할 수 있다. ProtoGENI와 PlanetLab에서는 이러한 API를 HTTPS에서 XML-RPC로 동작하도록 구현했다[6],[7]. SFA에서 정의한 인터페이스는 슬라이스와 컴포넌트를 중심으로 다음과 같이 정의했으며, 구체적인 형태는 Ⅲ장에서 AM API와 ProtoGENI 구현에서 자세히 소개한다.

가. 슬라이스 인터페이스

슬라이스 인터페이스는 크게 슬라이스 생성 및 티켓 관련 연산(CreateSlice, GetTicket, RedeemTicket, ReleaseTicket), 자원 할당 관련 연산(SplitTicket, LoanResources, UpdateSlice), 슬라이스 제어 연산(StartSlice, StopSlice, ResetSlice, DeleteSlice), 그리고 슬라이스 정보 조회 연산(ListSlices, GetResources)으로 구분하여 각 연산의 포맷과 기능을 명시하고 있다.

나. AM 인터페이스

AM 인터페이스는 슬라이스 작업을 처리할 수 있도록 개별 AM 및 컴포넌트의 상태를 관리하기 위한 연산(SetBootState, GetBootState, Reboot) 기능을 제공한다.[1)]

Ⅲ. GENI 구현 현황

1. 개요

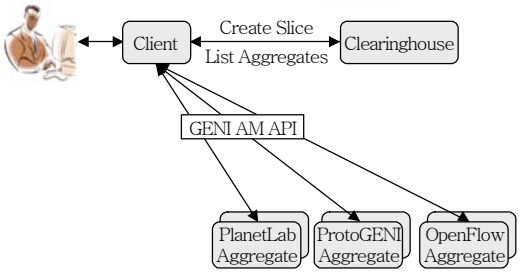
GENI에서 제시한 네 가지 원칙을 실현하기 위해 ProtoGENI와 PlanetLab 프로젝트에서는 각기 나름대로의 방식으로 컨트롤 프레임워크를 구현해 왔다. GENI에서는 SFA 기반 분산 시스템들의 생태계로서의 GENI 인프라스트럭처를 구성하도록, 각 프로젝트에서 자체적으로 정의한 인터페이스로 슬라이스 생성 및 자원 할당을 수행하도록 구성된 현재 각 프로젝트의 결과물을 하나의 표준화된 인터페이스로 규격화하여, 2010년 9월 1일, GENI Aggregate Manager API v1.0(이하 AM API)을 발표했다[2].

1) AM API 문서에서는 CM 인터페이스로 표기되어 있으나, CM은 1개의 컴포넌트를 가진 AM이므로 본 문서에서는 통일성을 위해 AM으로 표기한다.

이 문서는 GENI 설계 원칙과 기본 아키텍처, 추상화된 인터페이스가 명시된 SFA 문서보다 좀 더 시스템 구현에 가까운 API 규격을 명시했으며, 이와 더불어 AM API에 대한 참조 구현인 GCF와 이를 사용하기 위한 커맨드 라인 인터페이스로 구성된 클라이언트인 Omni도 제공하고 있다[10],[11].

2. GENI AM API

GENI의 목표 중 하나는 다양한 기관에서 운영되는 자원을 공유하여 네트워크 실험을 수행할 수 있는 환경을 제공하는 것이다. 이러한 맥락으로 여러 CH와 연동(페더레이션)되어 동작하기 위한 AM 규격을 정의했으며, 주요 내용은 현재 가장 완성도가 높으며, SFA를 잘 반영한 ProtoGENI와 PlanetLab의 결과물을 토대로 작성됐다. (그림 1)은 AM API를 적용한 GENI의 기본 시스템 구성도를 도시한 것이다. 사용자는 SFA 기반 시스템을 사용하기 위해 전용 클라이언트를 통해 특정 시스템의 CH에 접속하여 슬라이스 생성 및 실험을 수행한다. GENI 인프라스트럭처를 구성하는 다양한 자원은 여러 기관에 의해 운영되는 AM으로 관리되며, 사용자의 선택에 따라 해당 AM에 접속하여 필요한 작업을 처리한다. 가령 GENI 페더레이션을 구성하는 ProtoGENI는 AM 인터페이스를 따르며, 내부 동작은 기존의 자체 API로 처리하거나 AM 규격에 맞는 API를 직접 구현하는 형태로 상



(그림 1) AM API 기반 시스템 구성도[2]

호 연동을 지원한다.

AM API 문서에서는 SFA에서 구체적으로 명시되지 않은 인증 및 통신 방식도 명시하고 있다. 클라이언트와 AM 간의 안전한 통신을 위해 현재 널리 사용되고 있고 구현도 용이한 SSL[RFC 5246]을 인증 메커니즘으로 채택하고 있으며, Shibboleth[12]와 같은 다른 인증 메커니즘도 현재 검토하고 있다. 또한 분산된 AM 및 클라이언트 통신을 위해 HTTPS [RFC 2818] 기반 XML-RPC[13]를 적용하도록 규정하고 있다. 알려진 바와 같이 XML-RPC는 다양한 프로그래밍 언어를 지원할 뿐만 아니라, 구현도 용이하여 이미 ProtoGENI와 PlanetLab에서 사용하고 있으며, SSL도 손쉽게 지원할 수 있어서 현실적인 구현 방법으로 채택됐다.

3. 사용 시나리오 및 API 구성

가장 일반적인 사용 시나리오는 사용자가 슬라이스를 생성하고 여기에 필요한 자원을 할당하여 하나의 가상 네트워크를 구성하는 것이다. 단계별로 나열하면 다음과 같다.

(1) 클라이언트로 먼저 CH 접속하여 인증을 거치고, 슬라이스를 하나 생성한다. 이 때 생성된 슬라이스는 사용자와 권한이 연계된 구체적인 자원(슬리버) 없이 이름만 등록된 상태다.

(2) 현재 CH에 페더레이션된 AM 목록을 조회하고, 원하는 자원을 할당하도록 그 자원이 소속된 AM에 접속한다. AM API는 점진적으로 확장, 발전하기 때문에 GetVersion을 호출하여 본격적으로 연산을 수행할 API 버전을 확인한다.

(3) 특정 AM에서 제공하는 자원을 조회하도록 ListResources를 호출한다. 이때 결과는 Advertisement RSpec 포맷으로 전달된다.

(4) 원하는 자원을 선택하여 슬라이스에 할당하도록, 인자로 전달할 Request RSpec을 구성하고 CreateSliver를 호출한다. 요청의 결과로 실제로 할당된 자원 목록이 Manifest RSpec 형태로 반환된다. GENI의 분산 시스템 성격상, AM 및 하위 자원의 가용 상태는 전체 시스템 사용 현황에 따라 수시로 변경할 수 있으며, Request RSpec과 Manifest RSpec의 차이는 얼마든지 존재할 수 있다.

(5) 생성된 슬리버의 상태를 SliverStatus로 확인한다. 가령 최초 할당된 시점에는 본격적으로 사용할 수 있는 ready 상태에 놓인다.

(6) 슬리버의 사용 기간을 연장하거나 삭제하려면, 위와 마찬가지로 먼저 GetVersion을 호출한 뒤, RenewSliver나 DeleteSliver를 호출한다.

위 사용 시나리오를 통해 전반적인 실행 흐름 및 API 구성, 그리고 그 과정에 사용되는 RSpec, 인증, 식별자, 권한, credential 등을 확인할 수 있다. 먼저 GENI와 연계된 사용자나 자원, CH, AM, 슬리버, 슬라이스, 역할 등을 비롯한 모든 오브젝트는 고유한 식별자를 가지며, URN[RFC 214] 포맷을 적용하고 있다. <표 1>은 URN에 대한 예를 보여주고 있다.

CH와 AM에서 사용자를 인증하기 위해 X.509 v3 규격의 인증서로 공개키를 URN에 바인딩하며, PEM [RFC1421] 포맷으로 저장한다. 이러한 인증서는 SSL(HTTPS) 연결 과정에서 서버와 클라이언트 인증, 그리고 credential의 대상을 확인할 때 사용된다.

〈표 1〉 URN 예

종류	GENI URN
SA	urn:publicid:IDN + etri + authority + sa
유	urn:publicid:IDN + gcf:gpo + user + n
자원	urn:publicid:IDN + plc + node + switch + 1 + port + 2

Credential은 기본적으로 ProtoGENI의 방식을 따르며, PlanetLab과의 호환을 위해 몇 가지 권한이 추가됐다.

사용 시나리오의 핵심 단계인 슬리버 생성 과정에서 RSpec이 사용되는데, 현재는 AM에서 제공하는 자원의 목록을 표현하는 Advertisement RSpec과 요청할 자원의 목록을 표현하는 Request RSpec과 실제 요청된 자원을 담은 Manifest RSpec으로 구분하고 있다. 앞서 언급한 바와 같이 현재 통일된 형태의 포맷에 대해서는 활발한 논의가 진행되고 있으며, 특히 이기종 플랫폼 간의 페더레이션을 위해 RSpec의 비중이 크게 차지하고 있어서, 추후 SFA 및 AM API 문서에 추가될 가능성이 많다. 현재 ProtoGENI에서는 XML을 이용한 RSpec을 위와 같이 세 종류로 구분하여 사용하고 있다[9]. XML이 풍부한 기능과 범용성이 뛰어나지만, API 호출 과정에 실제 전달할 데이터보다 부가적인 데이터의 양이 많다는 부담이 있으며, AM API에서는 구체적인 포맷으로 XML이나 RDF으로 명시하고 있다[2].

4. GENI 참조 구현

GENI에서는 AM API와 함께 GCF와 Omni라는 참조 구현을 제공하고 있다[10],[11]. 물론 AM API에 대한 실질적인 구현 예는 GENI control framework WG의 cluster B, C, D에서 진행중인 PlanetLab과 ProtoGENI, ORCA의 웹사이트 및 코드 저장소를 통해 관련 문서와 함께 공개되어 있지만[8],[14],[15], GCF와 Omni는 이러한 프로젝트에서 추출된 공통 인터페이스를 토대로 제정된 AM API 표준의 구체적인 의미와 전체 시스템 구성을 명확히 표현한 성격이 강하다.

GCF와 Omni는 Python 2.6으로 작성됐으며, 다섯 개의 핵심 코드와 두 개의 패키지로 구성된다.

[핵심 코드]

- gen-certs.py: 인증서 생성
- gam.py: AM 참조 구현
- gch.py: 테스트용 CH
- omni.py: 클라이언트
- client.py: API 테스트용 스크립트

[라이브러리]

- geni: GPO에서 작성한 서버, Omni 관련 기능
- sfa: PlanetLab의 SFA 라이브러리 축소판

Omni 클라이언트는 사용자가 AM API를 통해 ProtoGENI, PlanetLab, GCF, OpenFlow에 접근하기 위한 커맨드라인 인터페이스를 제공한다[11]. 2010년 7월 개최된 GEC 8에서 세 프레임워크에 대한 공통 인터페이스로서 소개된 바 있으며, 좀 더 원활한 동작을 위한 보완 작업이 계속 진행중이다. 아직 CH API에 대해서는 표준화되지 않았으므로 현재 Omni는 CH와 네이티브 API로 통신하며, 플러그인 아키텍처를 통해 추후 정의된 CH API를 쉽게 반영할 수 있도록 구성했다. 또한 Omni에서는 ProtoGENI와 PlanetLab, OpenFlow AM에서 사용하는 고유의 RSpec을 하나의 언어(omnispec)로 통일하여, 사용자에게 자원을 표시하는 과정을 간소화 했다.

5. ProtoGENI 구현 동향

앞서 언급한 바와 같이 AM API는 PlanetLab과 ProtoGENI를 통해 실제 구현 검증된 결과를 토대로 제정된 표준 규격이다. 특히 API 부분에서는 ProtoGENI의 구현 사항이 상당 부분 반영되어 있다. 따라서 현재 ProtoGENI 구현에서는 AM API에서는 언급하지 않는 기능을 담고 있거나, 같은 기능의 API의

명칭에 차이가 있기도 하다. ProtoGENI는 Component Manager API라는 이름으로 규격을 사용하고 있으며, 현재 AM API와의 호환을 고려하여 재설계된 CM API Ver.2도 지원하고 있다. 두 규격은 API 명칭 수준의 차이가 존재하여[16], 현재 ProtoGENI는 어댑터 패턴[17]을 적용하여 구현하고 있다. <표 2>는 SFA와 AM API, ProtoGENI간의 API 차이점을 보여주고 있다.

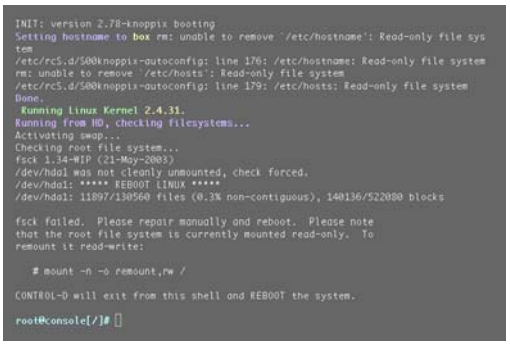
ProtoGENI에서는 GENI GCF에 해당하는 참조 구현인 Reference CM을 제공한다[15]. Reference CM은 ProtoGENI(Emulab)의 축소 버전으로서, 전반적인 컨트롤 프레임워크의 동작을 구현하고 있으며, QEMU를 이용한 가상 머신 관리자와 연동하는 기능도 간단히 구현되어 있다((그림 2) 참조). GENI의 Omni에 해당하는 오픈 소스 클라이언트는 제공하지 않지만, 테스트 스크립트를 통해 AM(CMv2) API 기능을 확인할 수 있으며, (그림 3)에서 보는 바와 같이 웹 기반의 GUI 인터페이스[18]를 통해 ProtoGENI에 연동된 Reference CM의 현황을 확인하고, 슬라이스도 WYSIWYG 형태로 구성할 수 있다.

ProtoGENI Reference CM은 Apache 기반의 XML-RPC 모듈을 연동한 형태로 구성되어 있으며, 핵심 모듈은 Perl로 작성됐다((그림 4) 참조). GCF와 달리 별도의 CH 모듈은 제공하지 않으며, 현재는 Emulab의 CH를 활용하고 있어서, Reference CM을 설치한 후 반드시 인증서를 CH에 등록해야 ProtoGENI에 연동된다.

특히 주목할 점은 별도의 가상 머신 관리자(QEMU)를 Libvirt 인터페이스로 제공하고 있어서, 몇 가지 연동 기능을 추가하면 QEMU 뿐만 아니라 Xen과 같

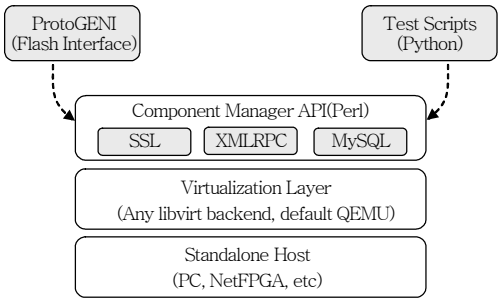
〈표 2〉 SFA와 AM, ProtoGENI의 API 차이

AM	SFA	CMv2
GetVersion	없음	없음
ListResources	GetResources	DiscoverResources, Resolve(슬라이스 지정된 경우)
CreateSliver	CreateSlice	CreateSliver
DeleteSliver	DeleteSlice	DeleteSlice
SliverStatus	없음	RenewSliver
RenewSliver	없음	RenewSlice
Shutdown	없음	Shutdown



(그림 2) QEMU 가상머신 실행 화면

(그림 3) ProtoGENI MapInterface



(그림 4) Reference CM 구조

은 다양한 가상 머신을 통해 슬리버를 구성해 볼 수 있다. 간단한 조작은 MapInterface로도 가능하지만, 구체적인 CM의 동작은 Python으로 작성된 테스트 스크립트를 사용해야 한다.

Ⅳ. 페더레이션

최근 SFA 문서 및 GEC 발표 자료에 의하면 페더레이션 관련 연구 및 개발이 활발히 진행되는 것을 확인할 수 있다. 기본적으로 GENI는 다양한 프로젝트로부터 개발된 이질적인 프레임워크를 하나의 통일된 인프라로 제공하기 위해 SFA라는 표준 구조와 AM API라는 인터페이스를 제시하고, 이 구조를 따르면서 자연스레 연동(페더레이션)되는 구조로 추진하고 있다. 이러한 관계를 바탕으로 AM과 CH API에 대한 지속적인 보완 작업을 통해 구체화하고 있다. 그러나 아직까지는 SFA에서 페더레이션을 위해 해결해야 할 기술 및 정책적 문제가 몇 가지 남아 있다. 대표적으로 다양한 플랫폼에서 제공하는 자원을 하나의 공통된 RSpec으로 표현하는 구체적인 방법은 아직 논의중에 있으며, 인증 및 권한 관련 문제를 위해 새로운 인증 메커니즘도 검토하고 있다. 또한 다양한 AM을 통해 가상 네타워크를 구성하기 위한 네트워크 스티칭(network stitching)에 대한 논의도 활발히 진행되고 있다. 단순한 SSO을 넘어서 슬라이스 기반 구조의 전반적인 실행 흐름과 다양한 권한을 가진 사용자가 이질적인 시스템을 통해 연동하는 만큼 간단히 해결될 문제는 아니다.

한편, 유럽에서도 PanLab과 GENI의 페더레이션을 SOA 기반의 구조로 해결하고자 하는 연구가 발표된 바 있으며[19], 동 저자의 2010년도 연구 결과에서는 SFA를 수용하여, 자체적으로 사용하는 자원의 종류를 GENI와 호환되는 형태로 제공하기 위한 RSpec 및 추상화 레이어(어댑터)를 추가하는 방안을 제시했다[20]. PlanetLab에서는 페더레이션 환경에서 다양한 자원을 표현할 뿐만 아니라 이러한 자원을 할당하는 정책을 반영하기 위한 메커니즘으로 sfi-ables라는 툴체인을 제안하여, 사용자가 제공하는 자원에 대한 접근 권한을 지정하여 페더레이션 환경에 적용하는 기능을 제공하고 있다[21]. 또한 FP7의 Federica 프로젝트는 GENI와 마찬가지로 유럽 전체를 연결하는 가상 인프라 기반의 미래 인터넷 테스트베드를 구축하고 있다[22].

한편 풍부한 컴퓨팅 기능을 제공하는 클라우드에 PlanetLab과 같은 GENI 인프라의 기술을 활용하여, 다양한 클라우드가 표준 프로토콜로 연계되는 Inter-Cloud 개념을 구현하는데 GENI 인프라 기술을 활용하는 프로젝트도 진행되고 있다[23].

Ⅴ. 결론

본 문서에서는 SFA와 AM API, 그리고 이에 대한 참조 구현과 AM API을 뒷받침하는 실제 구현 시스템인 ProtoGENI를 중심으로 현재까지 진행중인 GENI 프로젝트의 현황을 정리했다.

Spiral 2에서는 GENI의 기본 골력과 상위 인터페이스에 대해 어느 정도 구체화된 구현 단계에 이르렀다. 앞으로는 하위 메커니즘에 해당하는 이기종 플랫폼의 자원 명세를 위한 RSpec 부분과, 신뢰성 있는 시스템 연동을 위한 인증 및 credential 규격 부분, 그리고 네트워크 스티칭에 대해 구체화된 방법이 제시될 것으로 예상되며, 이러한 사항이 반영된 페더레이션 기반 시스템 흐름에 연동되는 클라이언트도 지속적으로 업데이트될 것으로 기대된다.

● 용 어 해 설 ●

슬리버(Sliver): 특정 AM/CM에서 제공하는 자원으로 구성된 슬라이스 구성 요소로서, 각 자원은 PC나 네트워크 장비 등이 해당 AM/CM에서 격리된 형태로 제공되거나, 가상화를 기반으로 컴퓨팅 및 통신 자원의 가상화된 복사본 형태로 제공된다.

약어 정리

ABAC	Attribute-Based Access Control
AM	Aggregate Manager
CH	Clearinghouse
CM	Component Manager
GCF	GENI Control Framework
GENI	Global Environment for Network Innovations
GID	Global Identifier
MA	Management Authority
SA	Slice Authority
SFA	Slice(-Based) Federation Architecture
SSO	Single-Sign On
UUID	Universally Unique Identifier

참 고 문 헌

[1] GENI Control Framework Requirements(DRAFT), GENI-SE-CF-RQ-01.3

[2] GENI Aggregate Manager API, GENI-SE-CF-AMAPI-01.0, Sep. 1, 2010.

[3] Larry Peterson, Robert Ricci, Aaron Falk, and Jeff Chase, Slice-Based Federation Architecture, Version 2.0, July 2010, http://groups.geni.net/geni/wiki/SliceFedArch

[4] PlanetLab Implementation of the Slice-based Facility Architecture, http://svn.planet-lab.org/attachment/wiki/WikiStart/sfa-impl.pdf

[5] http://groups.geni.net/geni/wiki

[6] http://www.protogeni.net

[7] http://www.planet-lab.org

[8] http://geni-orca.renci.org/trac/wiki

[9] ProtoGENI RSpec, http://www.protogeni.net/trac/protogeni/wiki/RSpec

[10] http://trac.gpolab.bbn.com/gcf

[11] http://trac.gpolab.bbn.com/gcf/wiki/Omni

[12] http://shibboleth.internet2.edu

[13] http://www.xmlrpc.com

[14] http://svn.planet-lab.org/wiki/MyPLC

[15] http://www.protogeni.net/trac/protogeni/wiki/ReferenceCM

[16] https://www.protogeni.net/trac/protogeni/wiki/GeniAggregateManager

[17] https://en,wikipedia.org/wiki/Adapter_pattern

[18] https://www.protogeni.net/trac/protogeni/wiki/MapInterface

[19] T. Magedanz et al., "Service-Oriented Testbed Infrastructures and Cross-Domain Federation for Future Internet Research," IM 2009, pp.101-106.

[20] Konrad Campowsky et al., "Resource Management in Large Scale Experimental Facilities," NOMS 2010, pp.930-933.

[21] Sapan Bhatia et al., "Establishing Resource Allocation Policies in Federated Systems," http://svn.planet-lab.org/attachment/wiki/WikiStart/sfatables.pdf

[22] http://www.fp7-federica.eu

[23] http://groups.geni.net/geni/wiki/GENICloud