미래인터넷 테스트베드의 Semi-federated Slice Control을 위한 Plastic Slice의 S/W 모델

S/W model of Plastic Slice for Semi-federated Slice Control of Future Internet Testbed

차병래*, 김종원*

Byung-Rae Cha*, Jong-Won Kim*

요 약

통신 및 컴퓨팅 환경의 급격한 변화 및 다양한 사용자 요구사항의 증대로 인해 현재의 인터넷이 갖는 근본적인 문제를 해결하기 위한 노력으로 미래인터넷 연구가 국내외로 활발히 진행되고 있다. 본 연구에서는 미래인터넷의 연구 주제인 Federation Job Control에 대한 Plastic Slice의 소프트웨어 모델의 기초 개념과 아이디어를 제안한다.

Abstract

In rapid changes in communications and computing environment, due to the increasing variety of user requirements, and an effort to solve the fundamental problems with the current Internet, Future Internet researches actively have been performed at domestic and abroad. In this paper, we proposed the idea and basic concept of Plastic slice software mode about federation job control in study topic of future internet.

Key words : Plastic Slice S/W Model, Federated Slice Control, Future Internet

I. 서 론

인터넷 개념이 처음 제안된 이후, 지난 40여년 가까이 인터넷은 현재 사용되는 글로벌 네트워크의 중추적인 역할을 담당하고 있다. 그러나 2000년대에 들어서면서부터 통신 환경 및 컴퓨팅 환경의 급격한 변화 및 다양한 사용자 요구사항의 증대로 인해 현재의 인터넷이 갖는 근본적인 문제에 대해 심각한 고민을 하기 시작하였으며, 그 연장선 중 하나로 최근 Clean Slate 설계 방법에 기반을 둔 미래인터넷 연구가 국내외로 활발히 진행되고 있다. 2005년부터 시작된 Clean Slate 기반의 이러한 연구 노력들은 미래인터넷(Future Internet) 연구들 로 총칭되며, 15년 이후의 새로운 인터넷 설계를 목표로, 미국 NSF가 후원하는 FIND [1], GENI [2]를 대표로 하여, 유럽 EU의 FP7 [3], EIFFEL [4], 일본의 차세대 네트워크(NWGN) 프로젝트 [5], 한국의 미래 인터넷 포럼 [6] 등에서 미래인터넷 관련 연구들이 활발히 진행되고 있다.

미래인터넷의 연구는 새로운 인터넷을 위한 설계 목표 및 요구사항을 다시 기술하고, 정의하는 작업으로부터 시작한다. 그림 1은 현재까지 논의되고 있는 미래인터넷을 위한 요구사항을 10가지로 정리하여

* 광주과학기술원 정보통신공학부
· 제1저자 (First Author) : 차병래
· 투고일자 : 2011년 10월 2일
· 심사(수정)일자 : 2011년 10월 7일 (수정일자 : 2012년 10월 23일)
· 게재일자 : 2012년 10월 30일

도식화한 것이다[7]. 미래 인터넷에 대한 확장성은 현 인터넷이 가지고 있는 가장 큰 문제 점 중의 하나로 미래인터넷을 설계할 때 가장 중요하게 요구되는 설계 목표 중 하나이며, 보안성/견고성, 이동성, 이질성, 서비스 품질, 재설정/자동설정, 상황인지, 관리성, 데이터-중심, 경제성 등의 요구사항이 많다.

Clean Slate 설계에 기반으로 둔 새로운 미래 인터넷 연구는 이를 실험하고, 검증하기 위해 새로운 테스트베드를 요구한다. 현재의 인터넷은 패킷 기반의 점대점 TCI/IP 모델을 기반으로 구축되어, 새로운 패러다임에 따른 미래인터넷 핵심 기술들을 실험하고 검증하기에는 어렵기 때문이다. 이러한 테스트베드의 필요성은 1969년 ARPANet이 현 인터넷의 테스트베드로서 TCP/IP 프로토콜과 패킷 스위칭을 글로벌 네트워크상에서 구현하고 실험하였다면, 이제 미래 인터넷을 위한 동일한 목적의 새로운 테스트베드가 필요하다. 미래인터넷 테스트베드를 위한 기본적인 요구사항으로는 물리계층을 포함한 전 계층에 대해 실험이 가능하여야 하고, 네트워크를 프로그램-가능하여야 하고, ARPANet이 NSFNet으로 진화했던 것처럼 앞으로의 상용수준의 진화도 고려하여 설계, 구축되어야 한다.

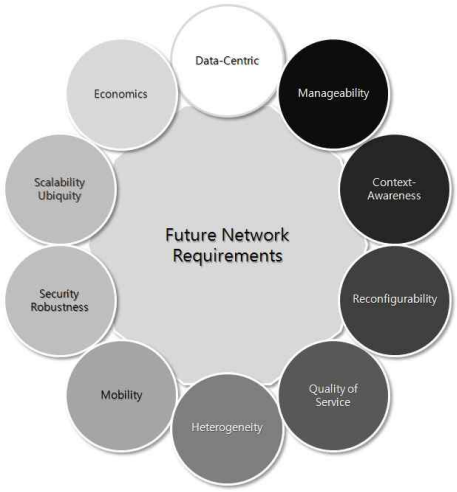

그림 1.. 미래인터넷 요구사항
Fig. 1. Requirements of Future Internet

Ⅱ. GENI

미래 인터넷의 테스트베드를 위한 대표적인 연구로는 PlanetLab [8], VINI [9], GENI [2] 등을 들 수 있다. PlanetLab 기술은 현재의 인터넷상에 새로운 서비스 및 응용을 실험하기 위한 글로벌 테스트베드 환경을 제공하며, 기존 TCP/IP 기반의 L3 이상의 환경에서 새로운 프로토콜과 응용을 실험하는 데 사용되고 있다. VINI는 사용자 제공 계층을 L2 계층에까지 확장하여 설계한 테스트베드이며, 이러한 기본적인 아이디어를 통합하여 미래인터넷을 위한 새로운 테스트베드 환경이 GENI이다. 미래인터넷 테스트베드에 대한 연구 주제로는 앞서 언급한 네트워크 가상화(Virtualization)를 비롯하여 자원통합 (Federation), 모니터링 (Monitoring), 측정(Measurement) 기술 등이 포함된다.

2-1 GENI의 WGs

GENI는 GENI Experiment Wokflow and Services WG (services-wg), GENI Control Framework WG (control-wg), GENI Campus/Operations, Management, Integration, and Security WG, 그리고 GENI Instrumentation and Measurement WG의 4개의 WG과 Cluster A ~ E의 5개의 GENI 프로젝트로 구성되었다. GENI Service WG는 연구자들의 실험에 필요한 서비스와 툴을 정의하고 있으며, 제어 프레임워크 개발자와 툴에 대한 구체적인 필요성 유추에 초점을 맞추고 있다. GENI Control Framework WG은 콤포넌트 제어, 슬라이스 제어, GENI에서의 접근제어, GENI와 와부와의 상호운용, 그리고 신원과 인증까지 연구 주제 범위를 포함하고 있으며, Spiral 2에서는 GENI Aggrergate API의 정의를 목표로 한다. GENI COMIS WG는 운영자와 연구자 시각에서 GENI의 운영과 서비스를 관리하기 위한 요구 사항과 GENI의 보안 요구사항을 연구 범위로 하고 있으며, 실험의 설정 및 운영에 대한 4가지의 Use Case를 정의하고 있다. GENI I&M WG는 I&M 아키텍쳐와 I&M 서비스의 기능을 위한 Measurement Orchestration (MO) Service, Measurement Point (MP) Service, Measurement Information (MI) Service, Measurement Collection (MC) Service, Measurement Analysis and Presentation (MAP) Service, 그리고 Measurement Data Archive (MDA) Service 등의 6가지 서비스를 정의하였다.

2-2 Plastic Slice Project

GENI의 Plastic Slice Project [10]는 Network-Oriented GENI 자원 연장 품질의 운영 및 관리하는 실험을 위한 초기 실험에서 발생할 문제를 발견하고 기록하기 위하여 8개 캠퍼스를 연결한 전국 규모의 인프라에서 3 개월 동안 지속적으로 10개의 GENI Slice를 실행하였으며, 그림 2는 단일 core VLAN에 대한 중규모(meso-scale) 배치의 다이어그램을 나타낸 것이다. 이 프로젝트에서는 Slice들을 이용하여 ICMP의 ping, 비암호화된 TCP의 netcat [11], 암호화된 TCP의 wget (HTTPS), 그리고 TCP와 UDP의 iperf [12]의 5가지 실험을 수행하였다. 이 프로젝트에 의해서 중규모의 GENI는 더 많은 실험을 위한 일반적인 운영 준비가 되었으며, 실험을 위한 어느 정도의 서로 격리가 가능하며, 연장 품질의 환경에서 사용할 수 있음을 보였다. 그러나 복잡한 실험을 위해서는 많은 도전과 지속적인 개선이 필요하다.

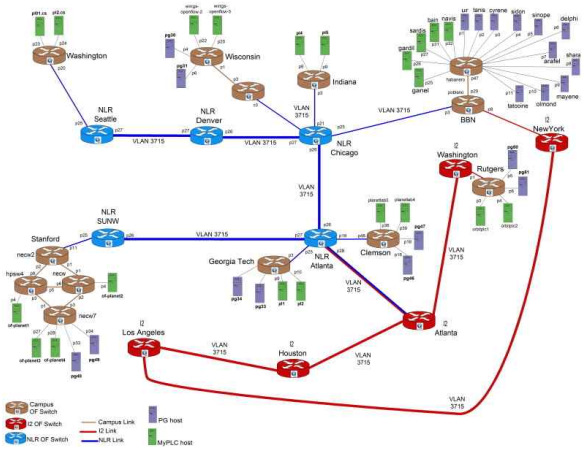

그림 2. GENI의 중규모 배치에서의 단일 Core VLAN의 다이어그램

Fig. 2. Single Core VLAN in meso-scale deployment in GENI

2-3 GENICloud, NEuca, 그리고 멀티클라우드

GENI의 Cluster C의 ProtoGENI와 Cluster D의 ORCA/BEN에서는 최근에 네트워크의 단말 노드에 클라우드 컴퓨팅을 도입하기 시작했다. 클라우드 컴퓨팅의 IaaS를 제공하는 과거 OpenSource인 Eucalyptus를 도입하여 적용을 시도하였으며, GENICloud [13]와 NEuca [14]를 제안하였다. GENICloud는 Top 접근법을 NEuca는 Bottom 접근법을 적용하였다. GENICloud는 PlanetLab의 단말 노드에 여러 개의 Eucalyptus 클라우드 컴퓨팅을 채용하였다. PlanetLab의 네트워크를 통해 컴퓨팅 자원을 이용하기 위해서는 Cloud Controller와 Cluster Controller 통해 Node Controller의 컴퓨팅 자원을 획득하는 절차를 거치게 된다. 그러나 NEuca는 Eucalyptus의 컴퓨팅 자원을 접근할 때, Cloud Controller와 Cluster Controller를 통하지 않고 하나의 클라우드 안의 많은 Node Controller들을 가상화된 단말 노드들로 채용하였다. 이러한 노드들을 Patch하여 통하여 컴퓨팅 노드들을 제공한다. 그림 3과 4는 GENICloud의 아키텍처와 NEuca의 네트워크 환경을 나타낸 것이다.

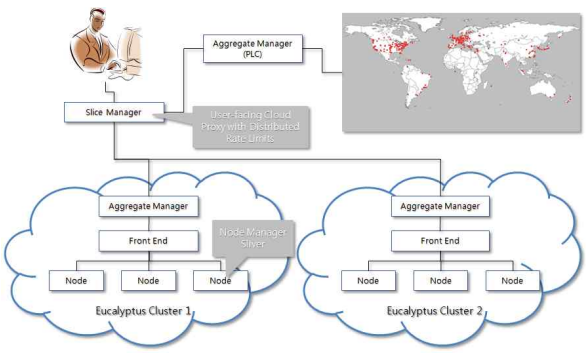

그림 3. GENICloud의 아키텍처

Fig. 3. Architecture of GENICloud

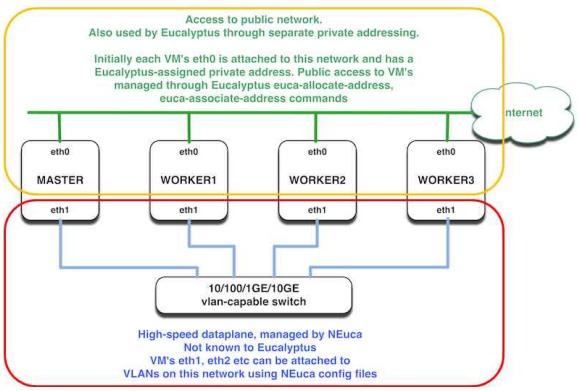

그림 4. NEuca의 네트워크 환경

Fig. 4. Network environment of NEuca

GENICloud는 InterCloud라는 개념에서 시작되었으며, 이와 유사한 개념으로 멀티 클라우드가 있다. 멀

티 클라우드 서비스의 플랫폼을 지원하는 기술은 라잇스케일 [15], 클라우드 브릿지 [16], 그리고 아헴스 하이클라우드 [17]가 있다. 라잇스케일은 클라우드 컴퓨팅 관리 플랫폼 기업으로 멀티 클라우드 하이브리드 서비스의 트렌드를 이끌고 있으며, 현재 아마존, 렉스페이스, 고그리드(Gogrid) 등이 라잇스케일 서비스에 파트너로 참여해 멀티 클라우드 서비스를 제공하고 있다. 라잇스케일 플랫폼은 서버 템플릿을 제공해 다양한 서비스 프로바이더의 가상 머신 이미지 호환성을 제공하며 표준 API를 제공해 과금과 모니터링을 통합 관리한다. 프라이빗-퍼블릭 하이브리드의 시트릭스의 클라우드 브릿지(Bridge) 솔루션은 젠(Xen, 가상화 솔루션) 커미터들의 영입을 시작으로 젠서버에서 젠앱(XenApp)에 이르는 IaaS-PaaS-SaaS 통합형 클라우드 솔루션을 제시했고 우수한 클라우드 기술을 확보하고 있다. 전통적인 데이터센터에 클라우드 브릿지 솔루션을 설치하면 시트릭스의 모든 서비스들이 브릿지 시스템을 통해 퍼블릭 데이터센터로 하이브리드할 수 있는 기능을 제공한다. 클라우드 브릿지는 네트워크 솔루션으로서 안전한 터널링 기법과 L2 오버레이 기술을 통해 서비스를 이용하는 사용자의 별도 노력 없이 안전하고 유연하게 퍼블릭 데이터센터와 하이브리드할 수 있다. 클라우드 브릿지 시스템이 L2 터널로 연결되고 IPSec을 이용하며 시트릭스의 클라우드 솔루션들을 통합한 형태(젠서버, 넷스칼라, 브랜치 리피터 등)로 구성된다.

그림 5. 시트릭스 클라우드 브릿지 개념도
Fig. 5. Concept diagram of Citrix cloud bridge

통합 하이브리드 형태의 아헴스 하이클라우드(HyCloud)는 국내 순수 기술로 개발된 프라이빗 데이터센터 관리 솔루션으로 아헴스 퍼블릭 클라우드 센터(2011년 1월 오픈) 및 글로벌 퍼블릭 클라우드와의 하이브리드 기능을 제공한다. 즉, 멀티 클라우드 및 프라이빗-퍼블릭 하이브리드 기능 모두를 통합 환경으로 제공한다.

2-4 SFA와 GENI AM API

GENI는 SFA (Slice Federation Architecture)를 정의하였으며, SFA는 제어 프레임워크 아키텍처로서, 상호 운영을 위한 Slice 기반의 네트워크 Substrate들의 연합 (Federation)을 허용하기 위한 자료 타입과 인터페이스의 최소 집합을 정의한다.

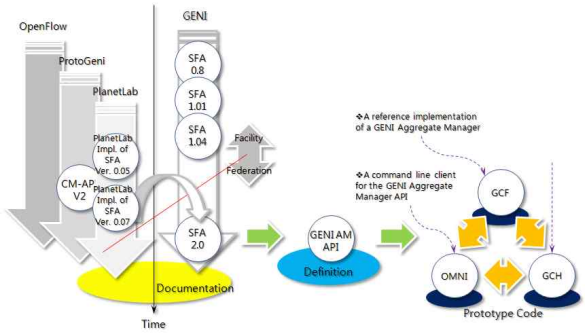

그림 6. SFA와 GENI AM API와의 관계도
Fig. 6. Relation diagram of SFA and GENI AM API

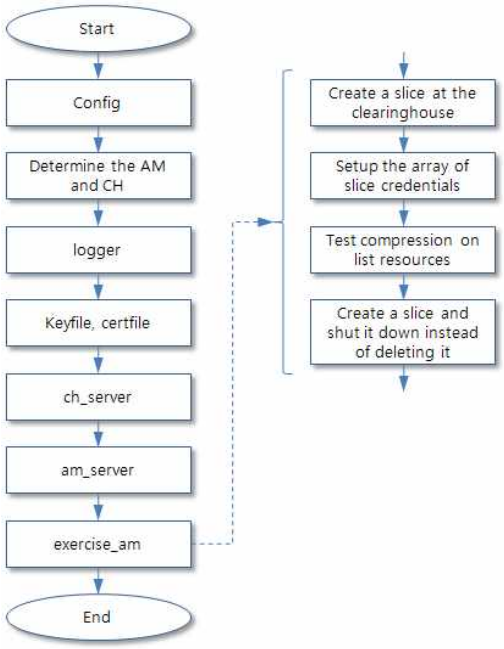

그림 7. gcf-test.py의 플로우챠트
Fig. 7. FlowChart of gcf-test.py

SFA Ver.1.04에서 SFA Ver. 2.0 [18]으로 업그레이드하면서 Slice Facility Architecture에서 Slice

Federation Architecture로 변경하였다. GENI AM API는 SFA 2.0을 기반으로 핵심 인터페이스의 프로토타입을 구현하여 배포하였다. 그림 6은 SFA와 GENI AM API [19]와의 관계를 나타내며, 그림 7은 GENI AM API의gcf-test.py의 소스 코드를 분석한 플로우챠트를 나타낸 것이다.

Ⅲ. 미래 인터넷 테스트베드의 Federated Job Control의 아이디어

미래인터넷 테스트베드에 대한 연구 테마인 Federation 분야의 주요한 연구 항목으로는 Federated Identity Management (FIdM), Federated Resource Management (FRM), 그리고 Federated Job Control (FJC) 등의 항목으로 구분되며 그림 8과 같이 표현된다.

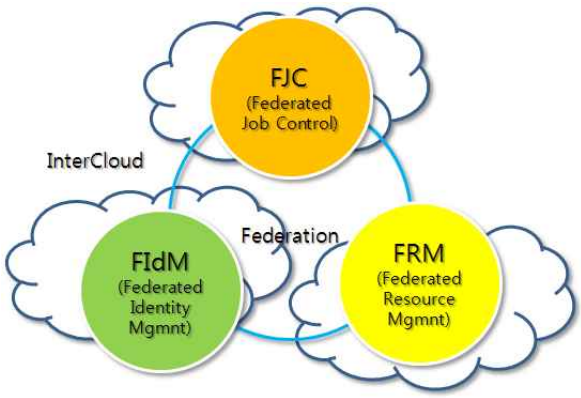

그림 8. Federation의 연구 주제
Fig. 8, Study topics of Federation

GENI의 Spiral 3까지는 Federated Identity Management, Federated Resource Management의 연구는 시작되어 어느 정도의 진척을 보이고 있으나, Federated Job Control 분야는 연구의 초기 상태이다. 본 연구에서는 FJC의 구성 요소와 이들 간의 기능, 그리고 Slice의 Federation에 대해서도 기술한다. 먼저 Job과 Process/Service를 구별하기 위하여 정의한다.

- Process: 실행된 컴퓨터 프로그램의 인스턴스이며, 프로그램 코드와 활동을 포함한다.
- Service: 재사용 가능한 소프트웨어 컴포넌트의 약한 결합에 의한 잘 정의된 비즈니스 기능을 수행하는 것.
- Job control: 컴퓨터 시스템에서의 Job 또는 멀티 Task들을 제어하는 것을 의미.

3-1 FJC의 구성 요소

FJC의 구성 요소로는 Node, Sliver [18], Slice [18] 등으로 구성된다. Node는 그림 9와 같이 나타내며, Node는 가상화에 의한 컴퓨팅 자원과 네트워킹 자원을 갖은 Site로 정의한다. 하나의 Node는 컴퓨팅과 네트워크 자원의 일부분으로 구성된 다수의 Sliver들로 구성된다. 또한 Sliver는 컴퓨팅 자원과 네트워크 자원을 갖은 가상 머신이다.

SFA에서는 하나의 슬라이스는 단순한 여러 Sliver들에 의해서 구성된다. 본 논문에서는 단순한 Sliver들이 아니라, Slice를 지원하기 위하여 좀더 전문적이며 지능적인 업무가 부여된 Sliver로 구성되는 것이다. Slice의 헤더 역할과 스케쥴링 역할이 필요한 Sliver, 실제 실험을 수행할 프로세스 역할의 Sliver, 여러 Sliver들의 상태를 파악하고 로그를 기록하는 역할을 수행할 Sliver, 그리고 컴퓨팅 및 네트워킹의 예비 기능 역할을 수행할 Sliver 등이다.

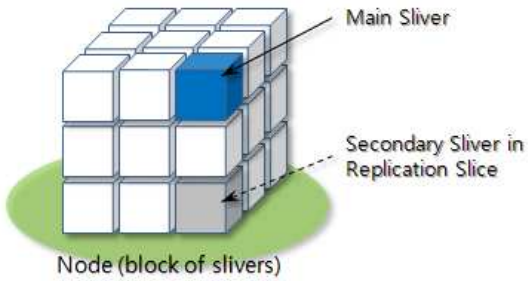

그림 9. 노드와 슬리버의 개념도
Fig. 9. Concept diagram of Node and Sliver

그림 10은 Slice를 추상적으로 나타낸 것이며, Sliver는 Master Sliver와 Slave Sliver, Monitor Sliver, 그리고 Provision Sliver로 기능에 의해서 분류되며, 다수의 Sliver들에 의해서 임의의 서비스를 제공할 수 있는 하나의 Slice를 구성하게 된다. 그림 11은 물리적 네트워크에서의 Slice를 나타낸 것이며, Slice를 구성하는 Sliver들은 Network Stitching에 의해서 연결된다.

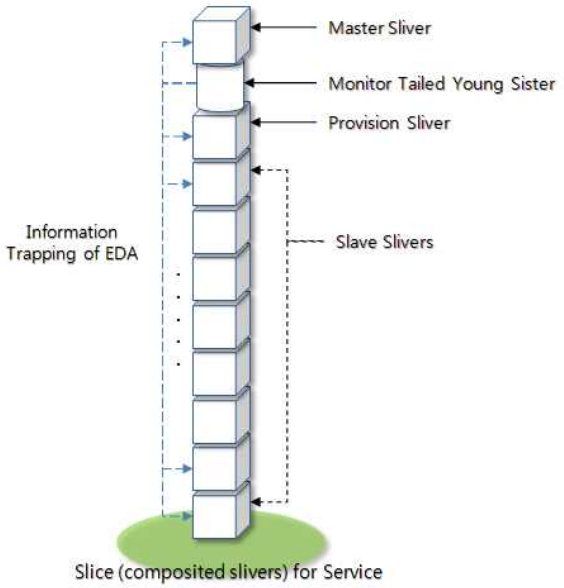

그림 10. Slice의 구성
Fig. 10. Composition of Slice

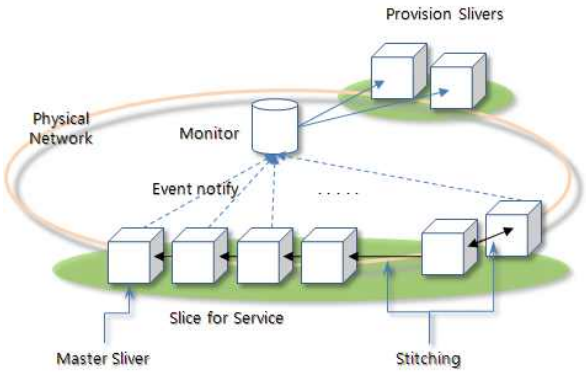

그림 11. 물리적 네트워크에서의 Slice 구성
Fig. 11. Slice composition in physical network

Monitor Sliver에 의해서 Slice를 구성하는 Sliver들의 상태 변화에 대한 비동기적 이벤트가 발생하면 Monitor Sliver에 통보하여 모니터링되며, 이에 대한 정상적인 운영을 위한 결정 및 Sliver의 프로세스에 대한 정보 저장은 Master Sliver에서 이루어진다. 또한 Provision Sliver는 서비스를 제공하고 있는 Slice의 Sliver 중에서 Node의 비정상적인 운영에 대한 예비 기능을 제공하며, 비정상적인 운영되는 Sliver에 대해서 교체 가능한 여분의 Sliver를 예비로 예약해 둔다.

3-2 Slice의 Federation

임의의 기업형 서비스 또는 공공 서비스를 지원하기 위해서는 Slice들의 Federation이 필요하게 되며, 그림 12와 그림 13과 같이 나타낼 수 있다.

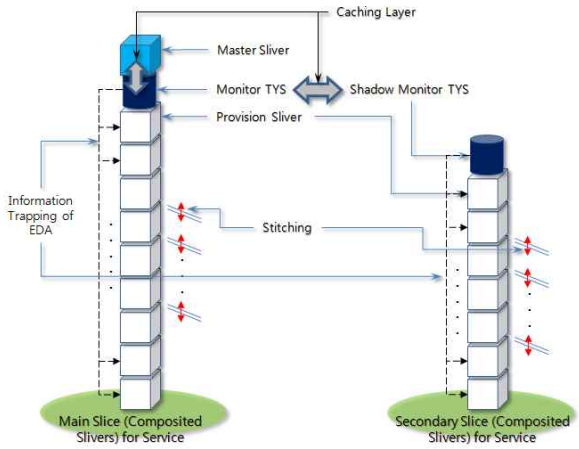

그림 12. 임의의 기업형 및 공공 서비스를 위한 Main Slice와 Secondary Slice의 Federation
Fig. 12. Federation of main slice and secondary slice for any enterprise and public services

일반적인 서비스는 하나의 Slice에 의해서 운영 및 지원이 가능하게 되지만, 기업 및 공공 서비스의 경우를 지원하기 위해서는 많은 컴퓨팅 자원과 네트워킹 자원이 필요 및 소요된다. 하나의 Slice에 의한 컴퓨팅 자원과 네트워킹 자원을 관리하기 보다는 컴퓨팅 자원 관리의 한계성 및 네트워크 자원의 경계 영역의 한계성, 그리고 시스템 성능의 효율성을 위해서 다중 Slice들의 Federation 기능에 의해서 효율적으로 지원이 가능하게 된다. 기업 또는 공공 서비스를 지원하기 위한 Federated Slices의 경우에는 크게 Main Slice와 다수의 Secondary Slice들로 구성된다. Main Slice는 앞 절에서 언급된 일반적인 Slice와 동일하다. 그러나 Federated Slice의 경우에는 Secondary Slice가 추가되는데, Secondary Slice는 Main Slice와 비교하여, Main Sliver가 존재하지 않으며, Monitor Sliver 대신에 Shadow Monitor가 Caching 역할을 수행하게 되며, Secondary Slice에서 발생된 모든 정보 교환 및 이벤트에 대하여 대행하는 역할을 수행하게 된다. Secondary Slice의 추가에 의해서 단일의 Slice에 의한 수행된 것 보다 병렬적인 측면에서 계산 복잡도와 시간 복잡도에 대해서 부분적으로 해결책을 제시할 수 있게 된다.

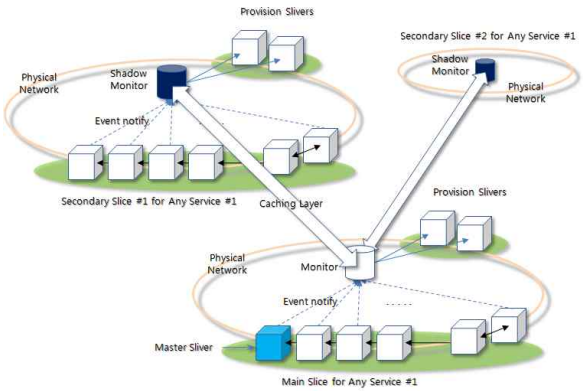

그림 13. 물리적 네트워크에서의 Main Slice와 Secondary Slice의 Federation
Fig. 13. Federation of main slice and secondary slice in physical network

Ⅳ. FJC의 적용

FJC를 적용하기 위해서는 임의의 서비스를 제공하기 위한 Slice 위에서의 스레드 (Thread)와 FJC을 제공하기 위한 기능들에 대해서 기술한다.

4-1 서비스를 위한 Slice

서비스를 제공하기 위한 Slice는 임의의 서비스를 제공하는 실행된 프로그램이 존재하여야 한다. 서비스를 제공하기 위해 실행된 프로그램은 소스 코드, 전역 데이터와 힙, 그리고 스레드들의 스택으로 구성되며, 그림 14와 같이 나타낼 수 있다.

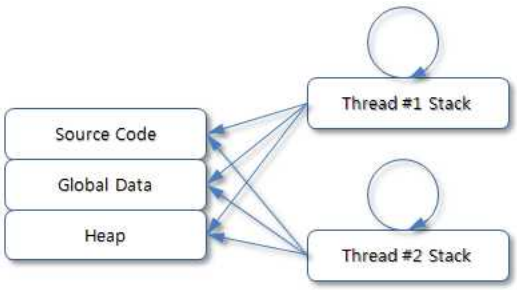

그림 14. 임의의 서비스 스레드
Fig. 14. Any service thread

임의의 서비스를 제공하기 위해 실행된 프로그램은 Sliver들로 구성된 Slice의 가상 머신들에 분산되어 적재되고, 수행되어야 하며, 그림 9와 같이 나타낼 수 있다. 임의의 서비스를 제공하기 위한 스레드 측면에서 그림 14와 그림 15를 비교하면, 단일 시스템과 달리, Slice 기반에서는 컴퓨팅 자원과 네트워크 자원이 분산되어 처리되어야 하기 때문에 Repository가 더 추가되어 있다.

Master Sliver의 가상 머신에서는 실행된 프로그램의 소스 코드, 전역 데이터, 그리고 힙의 자료가 저장 및 관리된다. 서비스를 처리하는 스레드들은 Slave Sliver들에서 분산되어 수행하게 된다. Master Sliver와 Slave Sliver들, 그리고 Monitor Sliver 간의 데이터 및 정보 교환, 그리고 스케줄링 등은 Caching Layer에 의해서 정보의 동기화가 ACID (Atomicity, Consistency, Isolation, Durability) [20] 특성에 의해서 실시간적으로 이루어지며, 이들 간의 상태 정보 및 Sliver들의 특정 이벤트에 의한 비동기적 통보 사항도 Monitor Sliver의 Repository에 저장하게 된다.

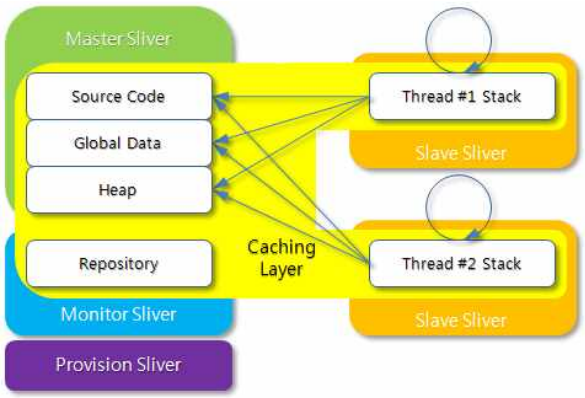

그림 15. Slice에서 임의의 서비스 스레드
Fig. 15. Any service thread in slice

4-2 Sliver의 구조

FJC의 기능을 기술하기 전에, 먼저 Sliver의 구조를 제안하며, 그림 16과 같이 Sliver는 크게 Computing 부분과 Networking 부분으로 나눌 수 있다. 즉, Computing 기능, Storage 기능, 그리고 Networking 기능으로 결합된 하나의 추상화된 객체로 정의할 수 있다.

Sliver의 Computing 부분과 Networking 부분의 사이에는 Pipe와 Channel들 그리고 Sliver gene가 존재하며, 그림 17과 같이 나타낸다. Pipe는 Main Sliver와 Secondary Sliver을 연결하는 채널이다. 이 Pipe에 의해서 Replication Sliver Switching이 이루어지게 되며,

참조되는 객체에는 영향을 미치지 않으며, Main Sliver의 권한을 위임받게 된다. Sliver Gene는 현 Sliver를 복제 및 생성 가능한 압축된 메타 데이터이며, 객체가 참조하는 모든 데이터까지 포함한다. Sliver Gene에 의해서 Node Switching이 이루어진다. 그리고 2개의 Channel에 의해서 FJC의 시스템 지원과 응용 지원을 위해서 사용하게 된다.

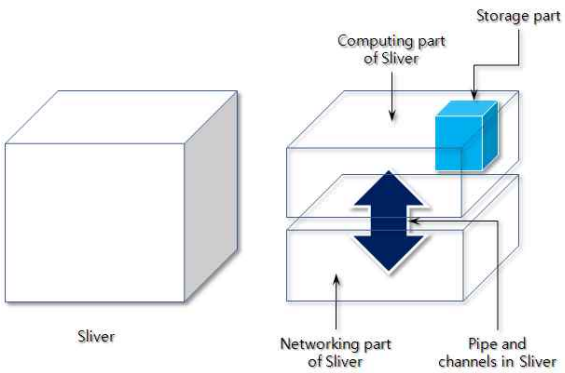

그림 16. Sliver의 구조
Flg. 16. Slicer structure

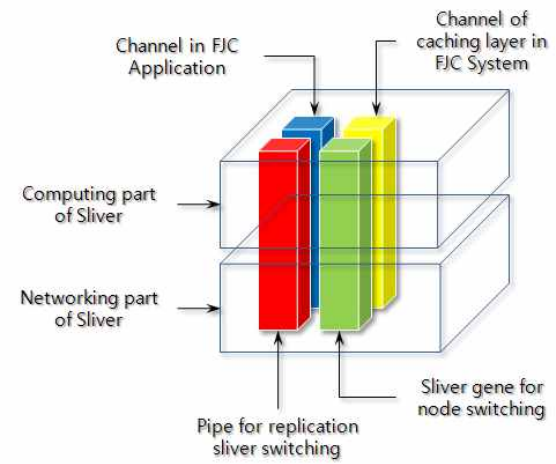

그림 17. Sliver의 Pipe, Channels 그리고 Sliver Gene
Fig. 17. Pipe, channels, and gene of sliver

4-3 FJC의 기능

임의의 서비스를 제공하기 위해서는 FJC의 기능들이 원활한 운영을 지원하여야 한다. FJC는 기본 기능과 확장 기능으로 구분된다. FJC의 기본 기능은 임의의 서비스를 지원하기 위한 FJC 기능으로 컴퓨팅 자원과 네트워크 자원의 가상화에 의한 Integration 기능과 Isolation 기능에 의해서 응용 지원과 시스템 지원으로 통합 및 격리에 의해서 구분한다. 응용 지원은 제공하고자 하는 서비스의 운용을 의미하며, 시스템 지원은 컴퓨팅과 네트워킹 자원의 Substrate 인프라 위에서 응용 지원을 위한 컨테이너 기능을 지원하는 것이다. 컴퓨팅 자원은 Bottom-Up 가상화를 네트워크 자원은 Top-Down 가상화에 통해서 지원한다. 응용 지원은 임의의 서비스를 운영하기 위한 컴퓨팅과 네트워크 자원을 할당 및 유지를 지원하는 것을 의미하며, 시스템 지원은 응용 지원을 원활한 운영을 지원하는 은닉되어 분산 제어 및 가상화된 컴퓨팅과 네트워크에 의한 보조 지원하는 Caching Layer 등이다. FJC의 확장 기능은 Sliver들로 구성된 Slice의 비정상적인 운영을 방지하기 위한 Fault Tolerance를 제공하기 위한 Replication 기능과 Provision 기능 등을 정의한다.

V. FJC의 기본 기능

FJC의 OpenCF는 크게 3계층으로 구분되는데 Substrate 계층, 시스템 계층, 그리고 응용 계층이다. Substrate 계층은 Plastic Slice의 소프트웨어 모델을 지원하는 물리적 인프라의 추상화를 의미한다.

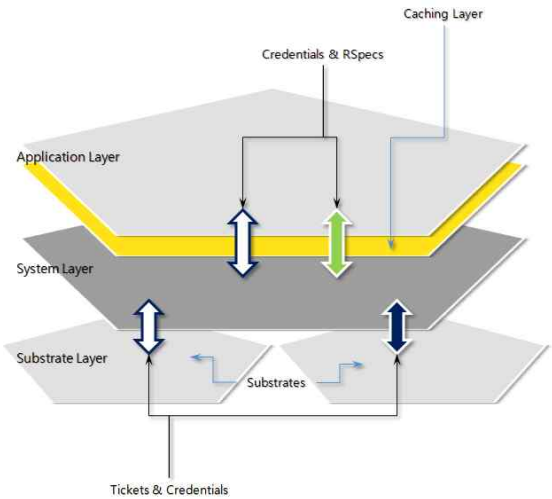

그림 18.서브스트레이트, 시스템, 그리고 응용 계층
Fig. 18. Substrate, System, and Application Layer

FJC는 시스템 계층과 응용 지원 계층에 의해서 Plastic Slice의 소프트웨어 모델을 지원한다. FJC의 기능은 크게 기본 기능과 확장 기능으로 구분된다. FJC의 기본 기능은 시스템 지원 기능과 응용 지원 기

능으로 구분되며, 그림 18과 같이 System Layer에서는 시스템 지원 기능을 제공하며, Application Layer에서는 응용 지원 기능을 제공한다.

5-1 FJC의 시스템 지원 기능

위에 기술된 FJC의 시스템 지원 기능과 역할 등에 대해서 그림 19와 같이 나타낸다.

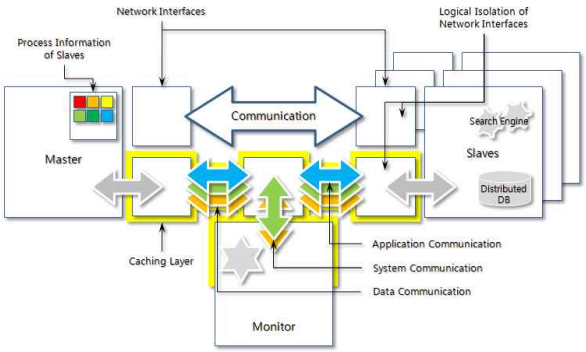

그림 19. FJC의 시스템 기능들
Fig. 19. System functions of FJC

(1) Network Interface의 Isolation

물리적 Network Interface는 하나이지만, 그림 9에 나타낸 것 같이 논리적 Network Interface를 Isolation 시켜서 응용 지원과 시스템 지원에 네트워크 자원인 대역폭을 할당한다. 네트워크 자원의 Isolation 시킴으로써, 응용 지원에서는 임의의 서비스에 필요한 네트워크 자원을 지원하며, 시스템 지원에서는 할당된 네트워크 자원을 Caching Layer에 지원하게 된다.

(2) Caching Layer

Caching Layer에 의해서 Sliver들로 구성된 Slice와 하나 이상의 Slice들을 엮어서 하나의 시스템과 같이 추상화된 Federated Job Control이 이루어질 수 있도록 지원한다.

Caching Layer에 의해서 격리된 네트워크의 대역폭을 최대한 활용하여 물리적으로 구분되어 네트워크에 연결된 분산 시스템을 하나의 시스템처럼 RMI와 유사하게 시스템 호출하듯이 수행할 기능에 대해서 실시간으로 처리한다. Caching Layer에서 발생한 비동기적 이벤트들의 이벤트 스탬프에 의해서 ACID를 지원하며, 일정한 단위 시간으로 Caching Layer의 동기화를 수행한다.

Caching Layer에 할당된 네트워크의 대역폭을 최대한 활용하기 위한 방편으로 BEEP (Blocks Extensible Exchange Protocol) [21, 22, 23]을 이용한다. BEEP은 직접적으로 데이터를 전송 및 수신을 위한 프로토콜이 아니며, 비동기 통신, 전송 계층의 보안, 피어 인증, 채널 멀티 플렉싱, 메시지 프레임, 채널 대역폭 관리 등의 여러 메커니즘을 재사용하여 임의의 특정 응용 프로토콜을 정의한다.

(3) Master Sliver

Master Sliver는 서비스를 위한 중요한 정책 및 결정을 수행하고, 시스템의 상황을 파악하고 이에 대한 요약된 정보를 기록하는 역할을 수행한다. 특히, Slave Sliver들의 프로세스 상태나 태스크에 대한 Context를 관리한다. 모든 정보는 Monitor Sliver의 Repository에 저장되어 있으며, 중요한 이벤트에 대한 요약 정보만을 Master Sliver에 기록하게 된다.

(4) Slave Sliver

Slave Sliver는 하나 이상으로 Slice를 구성하게 된다. 임의의 서비스를 지원하기 위한 실질적인 컴퓨팅 자원과 네트워킹 자원을 제공하는 역할을 수행하며, Master Sliver의 제어를 받게 되며, Slave Sliver의 모든 이벤트는 Monitor Sliver의 Repository에 저장된다.

(5) Monitor TYS와 Shadow Monitor

Monitor Sliver는 Monitor TYS와 Shadow Monitor로 구분된다. Monitor TYS는 Main Slice에 존재하는 Monitor Sliver를 나타내며, Shadow Monitor는 Secondary Slice의 존재하는 Monitor Sliver이다. Monitor TYS는 비동기 이벤트와 동기 이벤트에 대한 모니터링을 수행한다. 모든 이벤트는 비동기적으로 이벤트의 타임 스탬프 (Time stamp)에 의해서 ACID 속성으로 처리되며, 일정 단위의 주기적으로 동기화를 진행한다. Shadow Monitor는 Secondary Slice의 모든 정보를 모니터링하며, Main Slice의 Monitor TYS에 정보를 전송하는 역할을 수행한다.

5-2 FJC의 응용 지원 기능

FJC의 응용 지원 기능은 간략하게 서비스를 지원하기 위한 Initialization, Measuring, Loading, 그리고 Un-loading으로 정의하며, 나열된 순서로 진행된다.

그리고 그림 20과 같이 서비스에 대한 라이프 사이클을 갖는다.

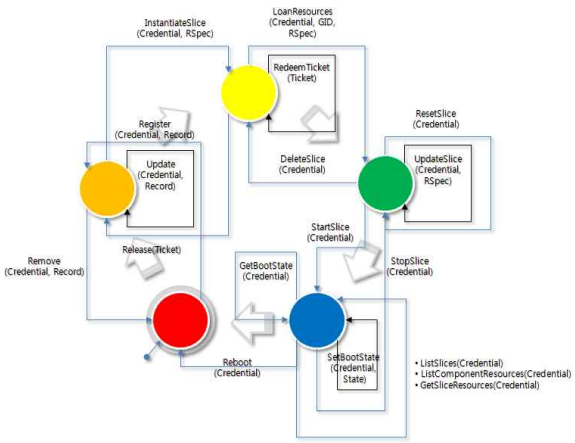

그림 20. FJC 응용 지원의 라이프 사이클
Fig. 20. Lifecycle of FJC application support

(1) Initialization

FJC의 응용 지원의 Initialization은 Sliver들 간의 Address 체계 설정과 information Trapping [24]에 의한 EDA (Event-Driven Architecture)를 설정을 초기화한다. EDA는 시스템 성능의 효율성을 위하여 주기적인 동기화 방법을 사용하지 않고, Information Trapping 기법에 의해서 특정한 이벤트들(Sliver의 상태 변화, Sliver switching, Node switching 등)을 설정한다.

(2) Measuring

Sliver들로 구성된 Slice 전반에 대한 예외 상황 및 시스템 성능에 관한 정보와 통계 정보를 제공한다.

(3) Loading과 Un-loading

수행할 임의의 서비스를 적재하고, 수행이 완료되거나 또는 비정상 종료되면 서비스를 제거하기 위한 절차를 수행하게 된다.

(4) 응용 지원의 라이프 사이클

물리적으로 분리된 Substrate들 위에 할당된 컴퓨팅 및 네트워킹 자원의 가상 머신인 Sliver들을 Network Stitching에 의해서 하나 또는 하나 이상의 논리적인 Slice들을 구성한다. 이 컨테이너를 제어하는 역할은 시스템 지원에서 담당하게 되며, 이 컨테이너 안에 임의의 서비스를 운영하게 되는데, 이 서비스는 그림 20과 같이 라이프 사이클을 갖게 된다.

VI. OpenCF의 보안 기능

본 연구에서 제안된 FJC를 OpenCF라 명명하며, 5가지의 보안 요소를 그림 21과 같이 나타낸다. OpenCF에 대해서 3단계의 절차로 구분할 수 있으며, Component와 Slice Control 단계, Authentication과 Access Control 단계, 그리고 Federation 단계이다. 컴포넌트와 슬라이스 제어 단계는 자원과 서비스에 대한 보안 절차이며, 인증과 접근 제어 단계는 FJC에 대한 인증과 자원과 서비스, 그리고 실험자들에 대한 접근 제어를 위한 보안 절차이다. 연동 단계는 각각 다른 도메인 영역에서 연합을 위한 보안 절차를 나타낸다.

기본적인 보안 절차를 간략하게 기술하면, FJC를 위한 User에 대한 식별 및 인증을 거친 다음에, 할당된 자원과 Slice의 제어권을 획득하게 된다. 접근 제어에 의해서 할당된 자원 관리와 추가적인 자원 할당의 허가 및 철회 등이 이루어지며, 확장된 접근 제어에 의해서 Federation이 지원하게 된다. 이러한 각 단계에서는 Tickets [18], Credentials [18], 그리고 RSpec (Resource specification) [18] 등의 데이터 교환이 이루어지며, 그림 18을 참조한다.

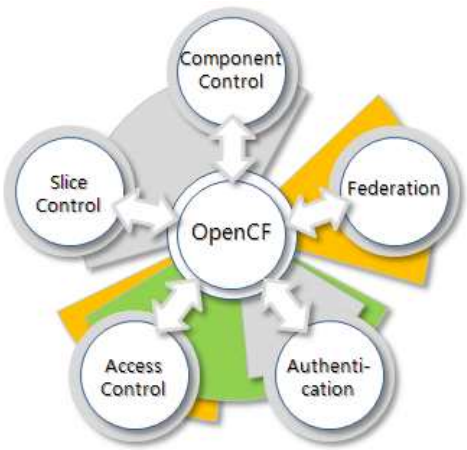

그림 21. OpenCF의 보안 요소
Fig. 21. Security components of OpenCF

Component와 Slice Control 단계에서는 사전에 사용할 자원과 서비스가 제어가 가능한 상태이어야 한

다. 이를 위해서는AM(Aggregate Manager)과 SA(Slice Authority)가 이러한 절차를 사전에 수행하게 된다. 그림 21에서의 OpenCF의 Slice Control 기능은 Slice Interface에 의해서 Slice를 실체화하고, 예비 Slice에 대한 추가적인 운영, 그리고 실체화된 Slice를 제어 및 Slice에 대한 정보를 제공한다. 또한 OpenCF의 Component Control 기능은 Component Management Interface에 의해서 Slice Interface를 지원한다.

Authentication과 Access Control 단계는 실험자 및 사용자의 인증과 자원과 서비스에 대한 접근 권한을 얻게 된다. 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. 사용자에게 주어진 권한 내에서 자원의 추가 할당의 허가와 철회가 이루어지며, 또한 권한에 의해서 서비스 이용 권한도 처가와 철회가 이루어지게 된다. 이러한 사항들을 지원하기 위한 과거의 접근 제어 기법은 IA&A (Identification, Authentication, & Authorization)에 의해서 IBAC (Identification-Based Access Control)이 적용되었다. IBAC은 중앙 집중의 단일 시스템을 지원하는 문제점을 갖고 있으며, 이를 해결하는 하나의 방안으로 FIdM (Federated Identity Management)으로 개선 및 보완되었다. 또 다른 방안으로 RBAC (Role-Based Access Control)이 제안되었다. RBAC은 간단한 시스템에는 매우 적합하지만, 많은 역할과 이에 대응하는 많은 정책 (Policy)을 적용하기에는 너무 복잡하다는 문제점을 갖고 있으며, Context를 지원하기에는 너무 역부족이다. 이러한 단점을 극복과 Fine-grained 접근 제어를 지원하고, 클라우드 컴퓨팅 환경에 따른 ABAC (Attribute-Based Access Control)가 현재 GENI에서 논의되고 있으며, 이에 준하는 CBAC (Claim-Based Access Control), 그리고 RAdAc (Risk Adaptive Access Control) 등이 연구되어 졌다.

Federation 단계는 자원과 서비스에 기전 도메인에 존재하지 않거든 대단위의 Job을 지원하기 위한 도메인(Domain) 간의 연합이 필요한 경우에 진행된다. Federation단계 또한 각각의 도메인에서의 자원들과 서비스들에 대한 접근 권한이 필수적으로 이루어져야 하며, 다른 도메인에 존재하는 자원들과 서비스들을 이용하기 위해서는 Federation 을 위한 절차를 진행하게 된다. 각각 다른 도메인의 자원들과 서비스들을 이용하기 위해서는 상호 도메인들 간의 세부적인 정보교환 보다는 추상화된 프로토콜 차원에서 광의의 협의가 사전에 이루어져야 한다. 이 또한 컴퓨팅과 네트워크 자원에 대한 속성의 의미를 동의하기가 어렵다는 이슈가 제기되었으며, 이를 해결하기 위한 방안으로 NBAC (autheNication-Based Access Control)과 ZBAC (authoriZation-Based Access Control) 등이 제안되었다 [25]. FJC를 지원하기 위한 접근제어는 Domain 영역 안에서는 NBAC에 의한 접근제어를 제공하며, Domain과 Domain 간의 접근제어는 ZBAC에 의해서 제공될 수 있으며, 그림 13을 참조한다.

부가적으로 안전한 실험을 제공하기 위한 컴퓨팅 서비스와 네트워킹 서비스에 대한 감내 기법(Fault Tolerance), 취약점 공격, 진화된 DoS 및 DDoS 공격, IDS(Intrusion Detection System), IPS(Intrusion Prevent System), Honeypot, 그리고 바이러스 및 좀비 등에 대한 연구가 진화적으로 계속 되어야 할 것이다.

VII. FJC의 Fault Tolerance을 위한 확장 기능

FJC의 확장 기능은 기본 기능의 원활한 수행을 위한 예외 상황 처리 및 Fault Tolerance을 제공하기 위한 기능을 제공한다.

임의의 서비스의 실체인 Slice를 구성하는 Sliver의 구현된다. Sliver의 구현은 가상 머신위에서 수행되는 하나 또는 하나 이상의 클래스로 대응될 것이다. 더불어, 모든 클래스는 클로닝을 위한 clone() 메소드가 제공된다. [26] clone() 메소드는 객체가 유지하고 있는 상태들을 그대로 복사하며, 복사되는 원래의 객체에는 아무런 변화를 주지 않으므로, 객체 구조의 완벽한 복사 기능을 제공한다. 객체의 복사 측면에서 shallow cloning과 deep cloning으로 구분할 수 있다. shallow cloning은 객체 구조를 복사하지만 객체 레퍼런스의 경우 레퍼런스만 복사한다. 즉, 객체 레퍼런

스가 가리키고 있는 실제 객체는 복사하지 않는다. 레퍼런스로 가리키는 객체까지 모두 복사하는 deep cloning과 구별된다. Sliver Switching은 shallow cloning에 의해서 구현되고, Node Switching은 deep cloning에 의해서 구현된다. 다음의 그림 22는 Slice와 Replication Slice의 이중 구조와 Sliver switching 및 Node switching을 나타낸 것이다.

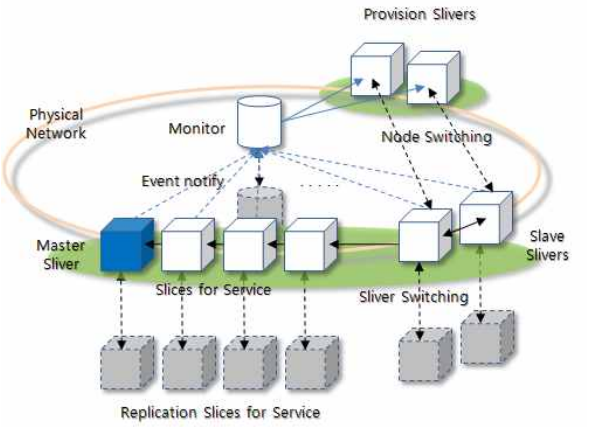

그림 22. Slice와 Replication Slice의 이중 구조
Fig. 22. Dual structure of slice and replication slice

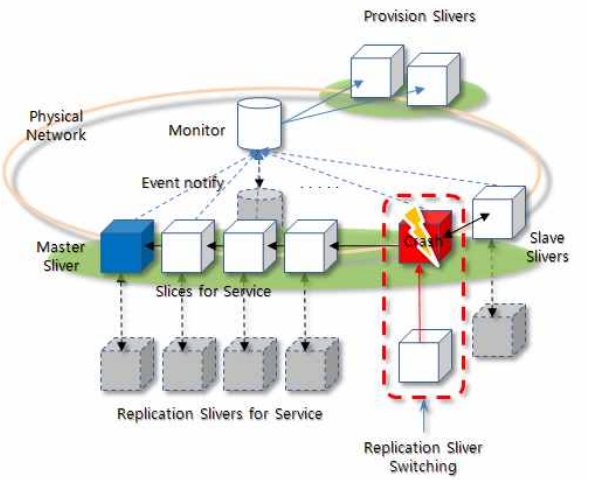

그림 23.슬리버 스위칭
Fig. 23, Sliver switching

6-1 Sliver의 이중 구조

Node의 모든 Sliver는 복제된 클래스가 이중 구조로 서비스를 수행하게 된다. 즉, 서비스를 위한 Slice와 Replication Slice의 이중 구조이다. Node에서 표면적으로 포그라운드에서 수행되는 Sliver를 Main Sliver라하며, Main Sliver를 복제하여 백그라운드에서 수행되는 Sliver를 Secondary Sliver라 한다. 만약 Node의 Main Sliver에서 문제가 발생하게 되면, 자동적으로 Secondary Sliver가 Main Sliver로 바뀌며, 문제가 발생된 객체는 가베지 켈렉션(Garbage collection)으로 자동적으로 자원이 회수된다. Main Sliver의 객체 구조와 레퍼런스만으로 구성된 Secondary Sliver가 shallow cloning에 의해서 복사되어 생성되고, Main Sliver의 서비스를 계속적으로 수행하게 되며, 그림 23과 같이 나타낸다.

그림 24는 그림 23의 Main Sliver의 객체 구조와 레퍼런스만으로 구성된 Secondary Sliver의 shallow cloning에 의한 복사 및 데이터 공유가 Pipe에 의해서 수행되는 것을 나타 낸 것이다. Main Sliver와 Secondary Sliver는 이 Pipe에 의해서 모든 데이터 교환 및 새로운 Secondary Sliver 생성 등에 사용된다. 또한 그림 20의 허가된 RSpec에 의한 Slice의 갱신 요구 시에도 Slice를 구성하는 Sliver들의 컴퓨팅 및 네트워킹 자원의 갱신도 동일한 절차에 의해서 수행하게 된다.

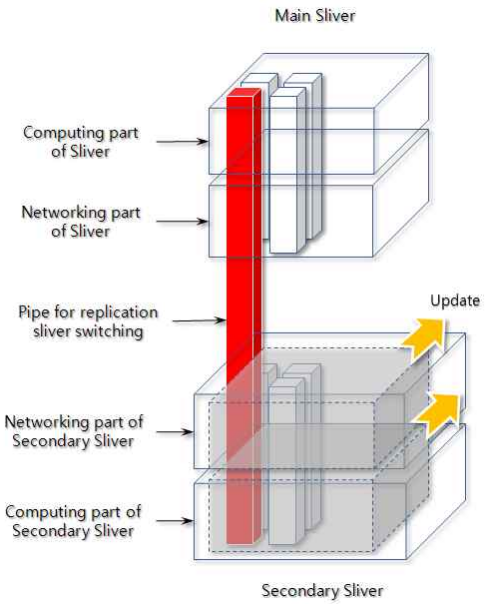

그림 24. Main Sliver와 Secondary Sliver의 Pipe와 Slice의 갱신
Fig. 24. Pipe of mail sliver and secondary sliver, and update of slice

Sliver를 포함하고 있는 Node에서 문제가 발생하게

되면, Monitor Sliver에 비동기적인 이벤트가 통보됨과 동시에 Provision Sliver에 전달하게 된다. Provision Sliver에 의해서 대체될 Node와 Node의 Sliver가 할당된다. 이때에 Node switching이 이루어지게 되며, deep cloning에 의해서 객체의 구조 및 레퍼런스가 가리키는 Node에 존재하는 객체까지 모두 복사하여 서비스의 연속성을 제공하게 됨과 동시에 Node들 간의 Network Stitching도 재설정하게 된다. 그림 25는 이러한 상황을 나타낸 것이다.

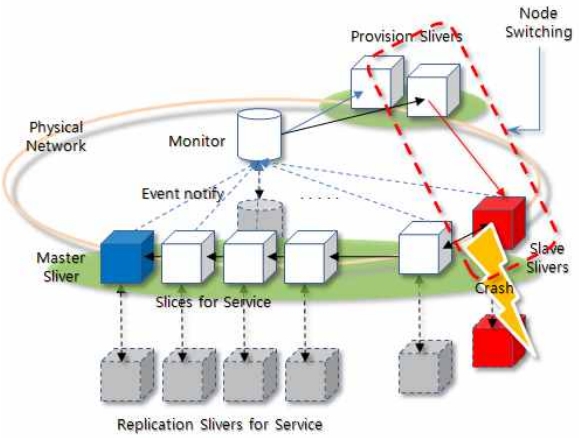

그림 25. 노드 스위칭
Fig. 25. Node switching

그림 25는 Sliver Gene 객체에 의해서 Node Switching을 나타낸 것이다. Sliver Gene 객체는 인간의 유전자 정보와 유사하게 기존의 문제가 된 Node의 정보를 Provision Sliver에 의해 제공되는 새로운 Node로 복사하기 위한 deep cloning에 의해서 객체의 구조 및 레퍼런스가 가리키는 기존 Node에 존재하는 모든 객체들까지 복사 및 압축된 통합된 데이터이다.

6-2 Sliver Gene

Sliver는 컴퓨팅/네트워킹 자원의 추상화된 객체로 정의하였다. Slice를 구성하는 한 Sliver의 감내(Fault Tolerance)가 불가능한 상태에 도달하면, 이러한 한계를 극복하고 안정성을 높이기 위해서 Provision Sliver와의 연동을 통해 안정적이며 효율적인 서비스를 지원할 수 있다.

Sliver Gene 객체에 의해서 기존의 Node의 역할이 대체/위임되어서 수행할 수 있는 새로운 Node를 구축하게 된다. Sliver Gene 객체는 Public Template와 Private Template로 구성된다. Public Template은 Slice를 구성하는 Sliver의 공통분모의 아키텍처 정보를 갖고 있으며, 매개변수의 속성 (Attribute)에 의해서 Master, Slave, Monitor, 그리고 Provision으로 구분된다. Private Template는 복사 및 압축이 필요한 Sliver가 갖고 있는 공통분모의 아키텍처를 제외한 나머지 정보들과 이 Sliver 안에 구성된 참조 가능한 객체들을 압축[27]하여 구성한다. Sliver Gene 객체는 Public Template에 의해서 Overloading 되며, Private Template에 의해서 Overriding 되어서 예비 Node에서 새로운 Sliver를 구축하여 서비스의 연속성을 제공하게 된다.

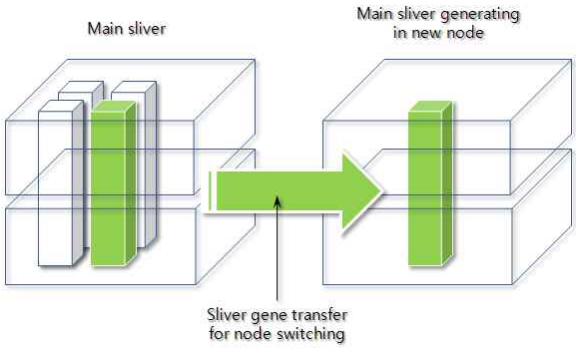

그림 26. Sliver Gene에 의한 노드 스위칭
Fig. 26. Node switching by sliver gene

VIII. 결 론

통신 및 컴퓨팅 환경의 급격한 변화 및 다양한 사용자 요구사항의 증대로 인해 현재의 인터넷이 갖는 근본적인 문제를 해결하기 위한 노력으로 미래인터넷 연구가 국내외로 활발히 진행되고 있다. 본 연구에서는 미래 인터넷의 연구 주제인 Federation Job Control을 위한 Plastic Slice의 소프트웨어 모델에 대한 기초 개념과 아이디어를 제안하였으며, FJC의 기본 기능의 응용 및 시스템 지원과 확장 기능에 대해서도 정의하였다.

감사의 글

이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임.

[NRF-2009-353-D00048].

이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2012R1A1A2041274)".

참 고 문 헌

[1] FIND, http://www.nsf.gov/pubs/2007/nsf07507/nsf07507.htm.
[2] GENI, http://www.geni.net.
[3] http://ec.europa.eu/research/fp7/index_en.cfm
[4] EIFFEL, http://www.future-internet-eu.
[5] NWGN, http://nwgn-forum.nict.go.jp
[6] 미래인터넷포럼, http://fif.kr/.
[7] 신명기, 김은숙, “Problem Statements and Requirements for Future Internet,” in Proc. ITU-T NGNGSI Meeting, 2007.
[8] PlanetLab, http://www.planet-lab.org.
[9] VINI, http://www.vini-veritas.net.
[10] http://groups.geni.net/geni/wiki/PlasticSlices/Final Report
[11] Netcat, http://en.wikipedia.org/wiki/Netcat
[12] Iperf, http://en.wikipedia.org/wiki/Iperf
[13] GENICloud, http://groups.geni.net/geni/wiki/GENICloud
[14] NEuca, https://geni-orca.renci.org/trac/wiki/NEuca-overview
[15] http://www.rightscale.com
[16] http://www.citrix.com
[17] http://www.ahems.co.kr
[18] SFA 2.0, http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf
[19] GENI AM API, http://groups.geni.net/geni/wiki/GeniApi
[20] ACID, http://en.wikipedia.org/wiki/ACID
[21] BEEP, http://beepcore.org/
[22] RFC3080, http://tools.ietf.org/html/rfc3080
[23] RFC3081, http://tools.ietf.org/html/rfc3081
[24] Tara Calishain, "Information Trapping: Real-Time Research on the Web," ISBN: 0-321-49171-8, New Riders, 2007.
[25] Alan H. Karp, Harry Haury, Michael H. Davis, "From ABAC to ZBAC: The Evolution of Access Control Models," Hewlett-Packard Development Company, L. P., Feb. 21, 2009.
[26] 객체 클로링, http://javacan.tistory.com/30
[27] Compressing VM, http://wiki.xensource.com/xenwiki/Compressing_VM_exports

차 병 래 (車炳來)

2004년 2월 : 국립 목포대학교 컴퓨터공학과 (공학박사)
2005년 3월 ~ 2009년 2월 : 호남대학교 컴퓨터공학과 전임강사
2009년 9월 ~ 현재 : 광주과학기술원(GIST), 정보통신공학부 연구조교수
관심분야 : 정보보안, Intrusion Detection System, 신경망, 클라우드 컴퓨팅, NFC기반 전자결제 시스템, Future Internet 등

김 종 원 (김종원)

1997년 8월 ~ 2001년 7월 : University of Southern California 연구 조교수
1999년 12월 ~ 2000년 7월 Technology Consultant for VProtect Systems Inc.
2000년 7월 ~ 2001년 6월 Technology Consultant for Southern California Division of InterVideo Inc.
2001년 9월 ~ 2008년 3월 광주과학기술원 정보기전공학부 부교수
2008년 4월 ~ 현재 광주과학기술원 정보기전공학부 교수
관심분야 : Networked Media Systems and Protocols focusing "Reliable and Flexible Delivery for Integrated Media over Wired/Wireless Networks“