K-ONE 기술 문서 #45
NFV 환경에서 서비스 고가용성 제공을 위한 구조 및 분석

Document No. K-ONE #45
Version 1.0
Date 2019-11-24
Author(s) 오재욱, 양현식

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성자	내용
0.1	19.11.12	오재욱, 양현식	Draft 작성
0.9	19.11.23	오재욱, 양현식	Draft 작성 완료
1.0	19.11.24	오재욱	오탈자 검증

본 문서는 2019년도 정부(과학기술정보통신부)의 재원으로
정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2015-0-00575,
글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2015-0-00575, Global SDN/NFV OpenSource Software Core Module/Function Development)

기술문서 요약

본 고는 NFV의 고가용성을 위한 기술 동향에 대한 기술문서이다. 네트워크 기능 가상화(Network Function Virtualization) 는 네트워크 라우터, IDS, 방화벽, 로드 밸런서(load balancer) 등의 전용 하드웨어 장치를 범용 컴퓨팅 환경 위에 가상화하여 소프트웨어로 구성하는 기술이다. 네트워크 기능 가상화 기술은 기존 전용 하드웨어 기반 시스템과 달리 하드웨어와 소프트웨어를 분리하여 기능을 구성하여 제공하는 기술로, 서비스 요구사항 및 상황에 따라 동적으로 서비스 환경을 제공하기에 적합하다. NFV 기술은 대표적으로 통신망의 인프라 구축 시 사용되는데, 이때 서비스를 안정적으로 제공하기 위해서는 가용성(Availability)과 신뢰성(Reliability)은 필수적으로 고려해야 한다.

NFV 환경에서 고가용성에 대한 연구는 다양한 플랫폼에서 논의가 진행되고 있는데, 특히 OPNFV(Open Platform for NFV) 에서는 고가용성을 주제로한 여러가지 프로젝트가 진행되었다. OPNFV에서 진행되는 고가용성 관련 프로젝트의 결과들은 업스트림 프로젝트에 반영되어 정식 릴리즈에 포함되기도 하였으며, 현재까지도 연구가 진행중이다.

뿐만 아니라, NFV 환경에서 최근 급속도로 성장하고 있는 IoT 클라우드에 배포되는 여러 서비스들의 특성을, 몇가지 기준 요소에 따라 분석 및 분류하여 IoT 클라우드 서비스에 알맞은 가용성 모델과 복원 플랜, 모니터링 타겟등을 정의하였다.

따라서 본 고에서는 OPNFV 고가용성 관련 프로젝트를 기반으로 고가용성 제공 방법에 대한 연구 결과 및 동향을 소개하고, NFV 환경에서 IoT 클라우드 서비스를 복원 방법, 서비스에 최적화된 가용성 방법 등을 분석한 내용을 소개하고자 한다.

Contents

K-ONE #45. NFV 환경에서 서비스 고가용성 제공을 위한 구조 및 분석

그림 목차

K-ONE #45. NFV 환경에서 서비스 고가용성 제공을 위한 구조 및 분석

1. NFV의 고가용성을 제공하기 위한 오픈 소스 프로젝트

가상화(Virtualization) 기술은 물리 서버를 구성하는 램(RAM), 스토리지(Storage), 네트워크 (Network), CPU 등과 같은 컴퓨팅 리소스를 가상화 하여 서비스를 제공 하는 기술이다. 네트워크 기능 가상화 (Network Function Virtualization) 기술은 라우터, 방화벽 및 로드 밸런서와 같은 전용 하드웨어 장치를 가상화 환경에서 제공하는 기술을 말한다. NFV 기술의 등장으로 인해 기존의 서비스 사업자들은 벤더에 의해 제공되는 전용 장비가 아닌 범용 장비에 네트워크 기능 소프트웨어를 설치하여 네트워크 서비스를 제공할 수 있다. 이로 인해 서비스 공급업체는 장비에 대한 투자 및 운용비용을 절감 할 수 있으며 서비스 대응 및 트래픽 변화에 대한 신속한 대응이 가능 해졌다. 한편, 가상화 환경에서 통신사업자 및 기타 클라우드 사업자가 서비스를 제공하기 위해서는 기존 물리적 환경에 동등한 서비스 가용성 기술을 제공해야 한다. 가상화 환경에 대한 산업 표준은 ETSI(European Telecommunications Standards Institute)에서 2012년 부터 통신 사업자를 중심으로 NFV 표준 그룹을 만들어 논의를 진행중이다[1]. Phase1을 시작으로 현재 5G와 관련하여 다양한 표준을 진행하고 있다. ETSI에서는 기본적인 NFV플랫폼 외에도 다양한 요구사항들을 반영하여 표준을 정의하고 있으며, 가용성 제공을 위한 구조 및 절차들도 개별 주제로 논의가 진행되었다. 대표적인 ETSI Reliability (REL003) 문서에서는 가용성 제공을 위한 모델들을 정의하고 오류 관리부터 복원까지 각 절차에 따른 요구사항들을 나열하였다[2].

ETSI에서 정의된 NFV 플랫폼은 다양한 클라우드 OS에 의해서 개발되고 있으며 대표적인 오픈소스 플랫폼으로는 OpenStack이 있다[3]. OpenStack은 NFVI(Network Function Virtualization Infrastructure)부터 MANO(Management and Orchestration) 인프라를 구축할 수 있는 환경을 제공하고 있다. 하지만 OpenStack은 NFV 환경 제공을 위한 플랫폼 구축을 중심으로 개발되고 있어 NFV를 사용하기 원하는 통신사업자의 기준에 부합하는 통합 플랫폼 구성의 필요성이 지속적으로 논의 되었다. 또한 현재 개발되고 있는 다양한 관련 솔루션들이 통합된 환경에서 동작하고 검증 할 수 있는 플랫폼 또한 지속적으로 요구되었다. 그 결과 공개소프트웨어 플랫폼으로 구성된 프로젝트가 OPNFV이다[4]. OPNFV는 통신사업자들의 요구사항을 정의하고 반영하는 동시에 NFV 환경에서 사용되는 다양한 서비스와 솔루션들을 통신장비에 적용할 수 있는 플랫폼을 개발하는 것을 목표로 하고 있다.

OPNFV의 창립 멤버는 AT&T, 브로케이드, 차이나모바일, 시스코, 델, 에릭슨, HP, 화웨이, IBM, 인텔등 다양한 통신사와 통신장비 제조사들이다. 현재 OPNFV는 통신사업자와 통신사들의 다양한 요구사항을 정의하고 구현하기 위해 각 주제별 프로젝트들을 진행하고 있으며 그 결과로 만들어진 새로운 구성요소를 통합하고 배포하기 위한 환경을 제공하는데 초점을 맞추고 있다. 특히 서비스의 가용성 제공과 관련된 프로젝트로는 DOCTOR와 HA(High Availability) 프로젝트가 있다[5] [6]. DOCTOR

는 NFV 환경에서의 오류 복구를 위한 구성요소 및 요구사항들을 정의하는 프로젝트로 초기버전부터 현재까지 지속적인 활동을 하고 있다. 초기 목표들을 마무리 하고 현재는Rolling Upgrade를 주제로 프로젝트가 이어지고 있다. HA 프로젝트는 고가용성 제공을 위해 NFV 구성 요소에 대한 모델 및 다양한 오류 시나리오 정의, 그리고 이를 기반으로 한 다양한 고려사항들을 논의하는 프로젝트이다. 최근에는 다양한 NFV 인프라에서의 가용성 제공 요소들에 대한 분석이 진행중이다. 따라서 본 고에서는 OPNFV HA와 DOCTOR 프로젝트를 중심으로 NFV 환경에서의 서비스 가용성 제공을 위한 구조 및 동향을 분석하고자 한다.

1.1. OPNFV HA 프로젝트

OPNFV HA(High Availability) 프로젝트는 Carrier Grade NFV 시나리오와 관련하여 OPNFV 플랫폼의 고가용성 위해 필요한 요구사항을 정의하는 것을 목표로 하고 있다. 본 프로젝트에서는 고가용성에 대한 요구 사항과 솔루션을 3 가지로 분류하여 정의하고 있는데 하드웨어의 고가용성과 가상 인프라의 고가용성, 그리고 마지막으로 서비스에 대한 고가용성 이다.

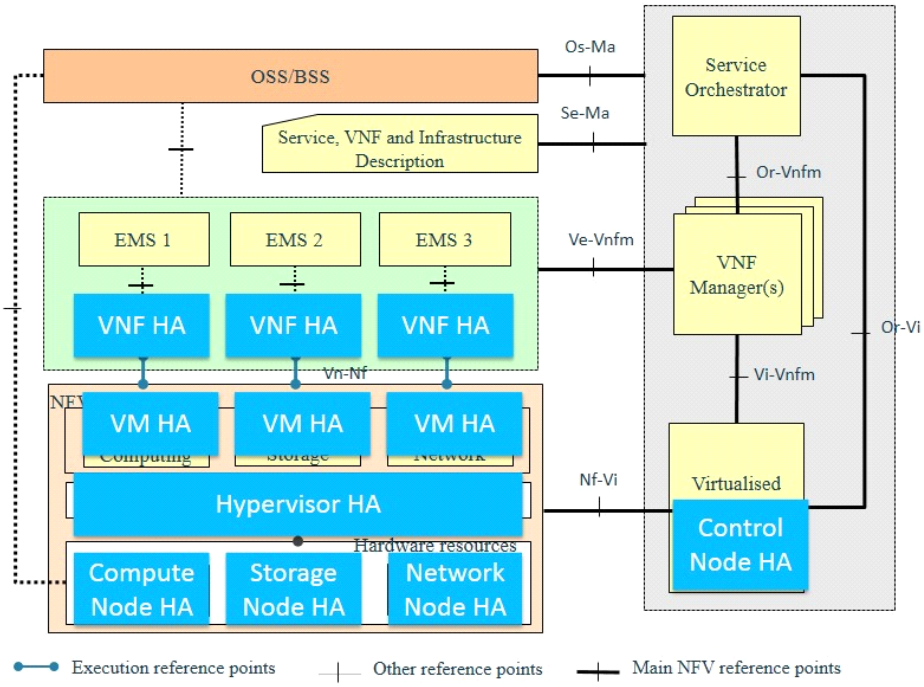

그림 1. NFV 레퍼런스 구조에서의 고가용성 제공 요소

하드웨어에 대한 고가용성은 컴퓨팅, 네트워크, 스토리지 등의 실제 하드웨어에 대한 가용성을 의미하며, 가상 인프라의 고가용성은 물리적인 리소스를 가상화 시키는 하이퍼바이저(Hypervisor)의 가용성을 의미하며 서비스의 가용성은 서비스를 제공하는 VNF와 가상 머신 등을 의미한다. 하드웨어 환경 및 가상 인프라 환경에

서의 장애는 VNF에도 영향을 미치기 때문에 고가용성 제공을 위해서는 모든 부분이 고려되어야 한다. 또한, 전체 인프라를 관리하는 컨트롤러에 대해서도 고가용성 제공 구조가 고려되어야 한다. OPNFV HA 프로젝트에서는 그림 1의 NFV 레퍼런스 구조를 바탕으로 VNF 특성 및 가용성 제공방법 등을 고려하여 8개의 유즈케이스를 구성하였다. OPNFV HA 프로젝트에서 구성한 8개의 유즈케이스는 표 1과 같다.

	VNF Statefullness	VNF Redundancy	Failure detection	Use Case
VNF	yes	yes	VNF level only	UC1
			VNF & NFVI levels	UC2
		no	VNF level only	UC3
			VNF & NFVI levels	UC4
	no	yes	VNF level only	UC5
			VNF & NFVI levels	UC6
		no	VNF level only	UC7
			VNF & NFVI levels	UC8

표 1. NFV 환경에서의 고가용성 제공을 위한 유즈 케이스

OPNFV에서 정의한 유즈케이스는 장애가 발생하는 원인을 어플리케이션 레벨과 인프라 레벨로 구분하였으며, 각 케이스 별로 가용성 제공을 위한 리던던시 VNF가 있는 경우와 없는 경우로 구분하여 분류하였다. 또한, VNF가 Stateful한 모델인 경우와 Stateless한 모델인 경우를 구분하여 유즈케이스를 분류하였다.

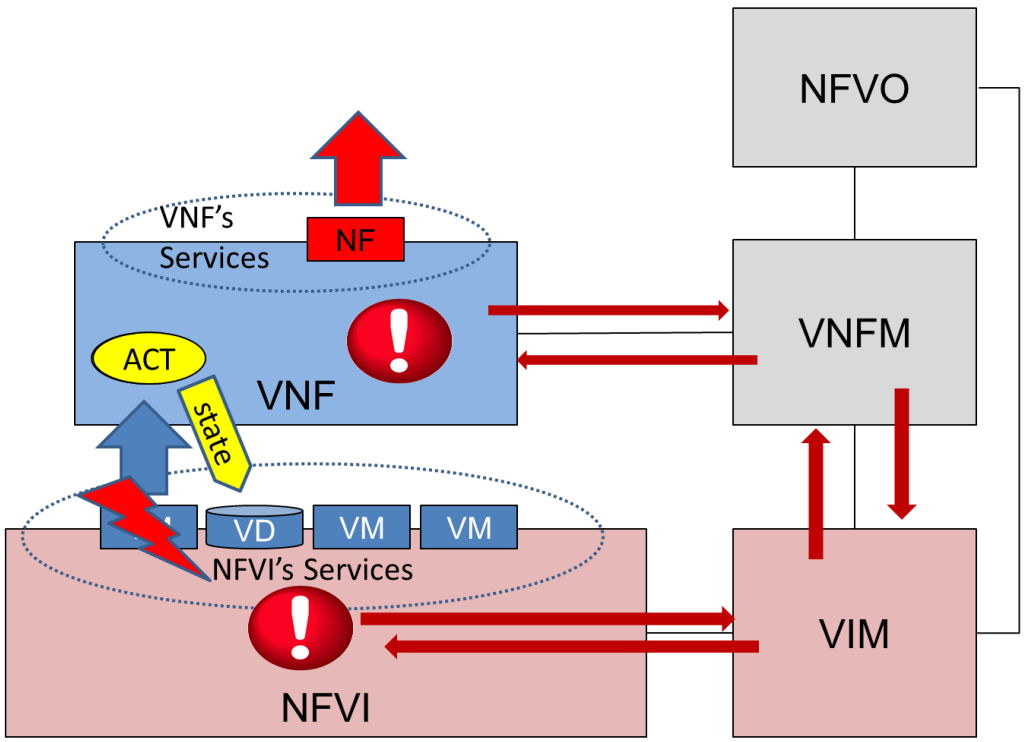

그림 2. Non Redundancy 모델에서의 Stateful VNF 장애 발생

8개의 유즈케이스는 각 시나리오에 따라 NFV 환경에 적용하여 상세 절차를 적용

하였다. 그림2는 Redundancy 모델이 적용되지 않은 환경에서 동작하던 Stateful한 VNF가 오류가 나는 경우에 대한 시나리오이다. 그림2와 같이 가상 인프라 환경에서 장애가 발생하는 경우, 그 위에서 동작하고 있던 어플리케이션에도 영향을 미치기 때문에 인프라와 어플리케이션에 대한 장애 절차가 함께 진행되어야 하며 서비스 특성에 따라 State에 대한 별도 처리 과정도 함께 고려되어야 한다. 이를 위해 VNFM(Virtual Network Function Manager)와 NFVO(Network Function Virtualization Orchestrator)와의 동작 절차들도 정의하고 있다. 이처럼 OPNFV HA 프로젝트에서는 두번째 릴리즈에서 각 장애 발생원인 및 복원 방법, VNF 특성에 따라, NFV 레퍼런스 구조에서의 장애 복원 절차들을 정의하였다. 이후 OPNFV HA 프로젝트에서는 고가용성을 위한 요구사항들을 더 세분화하여 각 환경의 고가용성 요구사항을 정의하고 있다.

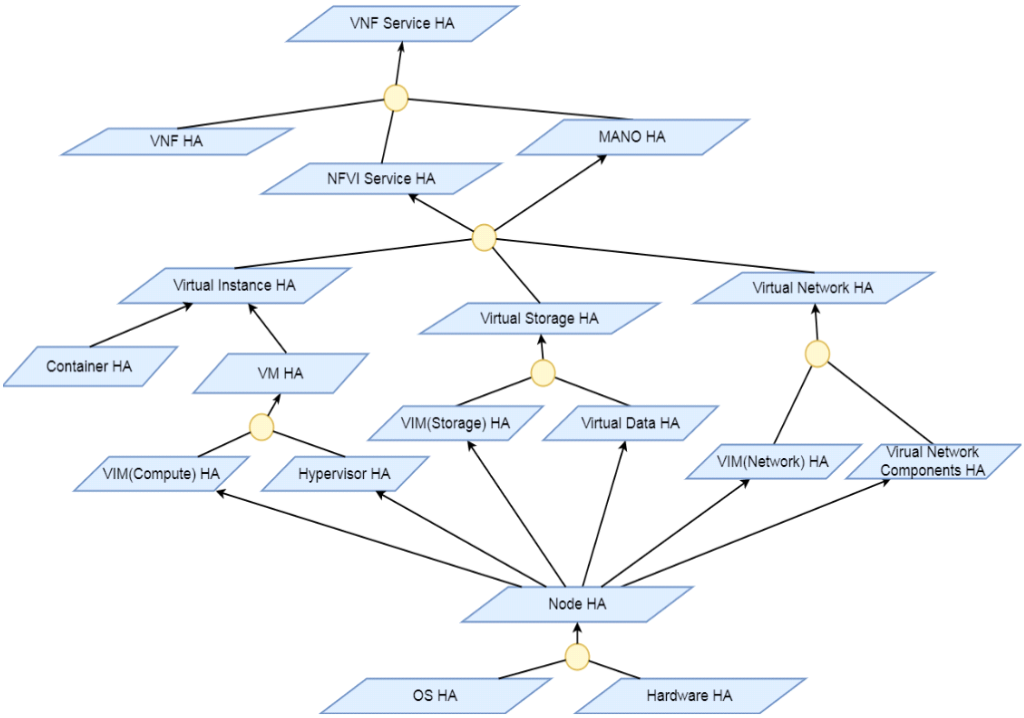

그림 3 NFV 환경에서의 HA를 위한 고려사항

그림3은 NFV 플랫폼의 전체 컴포넌트를 계층적으로 분류하고 각 계층의 컴포넌트의 가용성 제공구조를 나타낸 그림이다. NFV 환경에서 고가용성을 위해 고려되어야 하는 요소는 Hardware / Host OS / Hypervisor / SDN / VIM / Virtual Infrastructure / MANO / VNF 등이 있다. Hardware / Host OS / Hypervisor는 기존에 정의된 것들도 포함되어 있지만, SDN, VIM, Virtual Infrastructure등은 새롭게 추가된 요소이다. 특히, VIM에서는 OpenStack 및 기타 리소스 컨트롤러를 기반으로 기능에 따라 분류하여 기능별 가용성 제공을 위한 상관관계를 표시하고 있으며 최근에 많이 사용되는 컨테이너 환경의 플랫폼도 VIM 구성 시나리오로서 포함

되어 있다. 또한, 일부 컴포넌트는 OPNFV의 기능 테스트 프로젝트인 Yardstick과 연동되어 테스트 케이스로 포함되어 있다[7].

2. OPNFV Doctor

2.1. Doctor Project

OPNFV Doctor프로젝트는 OPNFV 초기 버전부터 진행된 프로젝트로 NFV 환경에서 오류 관리 및 유지 관리를 위한 프레임 워크를 구성하는 것을 목적으로 하는 프로젝트이다. 즉, DOCTOR 프로젝트의 목표는 가상화 된 인프라에서의 서비스 고가용성을 제공하는 것으로 주요 기능으로는 물리적 요소에 의해 발생한 오류에 대한 전달을 통해 서비스의 가용성을 보장하는 것으로 그림4는 DOCTOR 프로젝트의 개발 범위를 나타낸다.

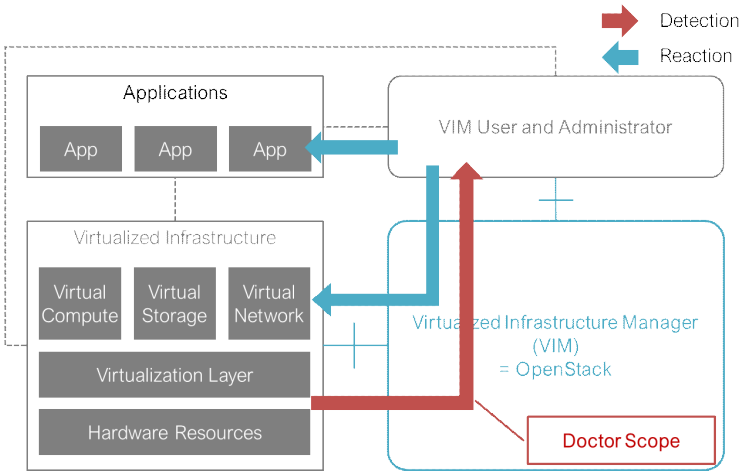

그림 4. OPNFV DOCTOR 프로젝트 개발범위

즉, 앞서 소개한 HA프로젝트에서 정의된 시나리오의 절차를 위해 필요한 기능들을 분류하고 설계하는 것이 목적이라고 할 수 있다. DOCTOR 프로젝트에서는 크게 두가지 시나리오를 해결하고자 하였다. 하나는 오류 관리 방안이고 두번째는 유지 방안(Maintenance)이다. 먼저 오류 관리 방안에 대한 시나리오는 그림 5와 같다.

오류 관리 시나리오에서는 현재 서비스와 실제 호스트간의 연관성을 확인하고 특정 서비스를 복원하는 절차등을 서술하고 있으며, 이를 위해 필요한 절차나 정보들을 함께 정의하고 있다. 즉, 특정 물리서버에서 장애가 발생하는 경우, 장애 정보를 VIM에게 알리고 VIM의 정보를 통해 발생한 장애의 영향을 받는 서비스에 대해 이전이나 복원과 같은 액션을 수행하게 하는 것을 목표로 하고 있다.

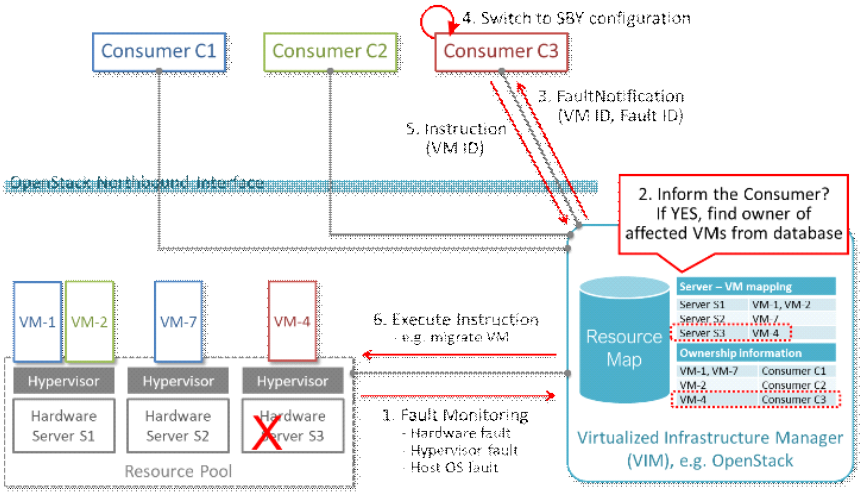

그림 5. DOCTOR 프로젝트 오류 관리 시나리오

유지 (Maintenance)관리에 대한 시나리오는 그림 6과 같다. 유지 관리 시나리오에서는 클라우드 관리자가 특정 물리 서버에 대한 유지 보수를 위해 요청을 하면, 요청된 물리서버에 의해 영향을 받는 특정 서비스에 대한 조치를 취함으로써 서비스가 영향을 받지 않도록 하는 것을 목표로 하였다.

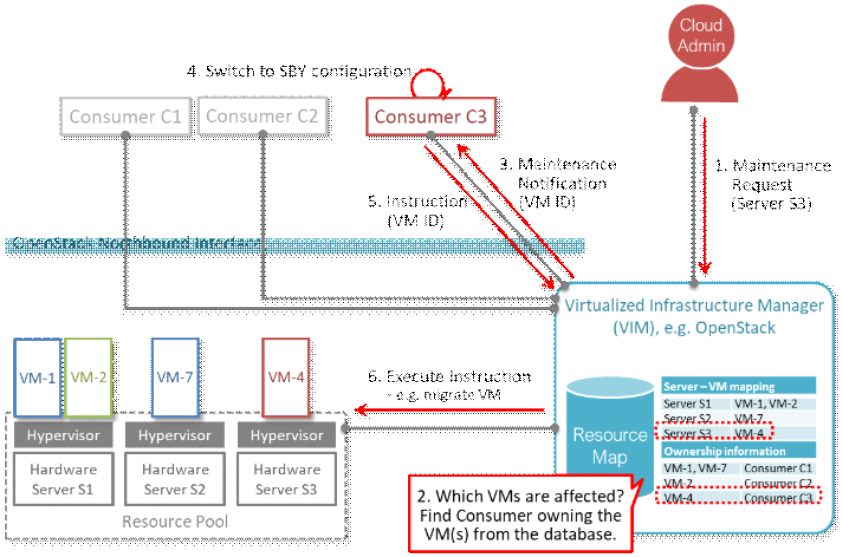

그림 6. DOCTOR 프로젝트 인프라 유지 관리 시나리오

이 두가지 시나리오와 같이 오류 및 유지 관리를 위한 프레임 워크 구성을 위해 DOCTOR 프로젝트에서는 총 4가지의 컴포넌트를 설계하였다. 그림 7은 NFVI에 대한 정보 수집을 통해 장애와 관련된 VNF에 대한 상태 확인 및 고가용성을 제공하기 위해 제안된 기능 구조이다.

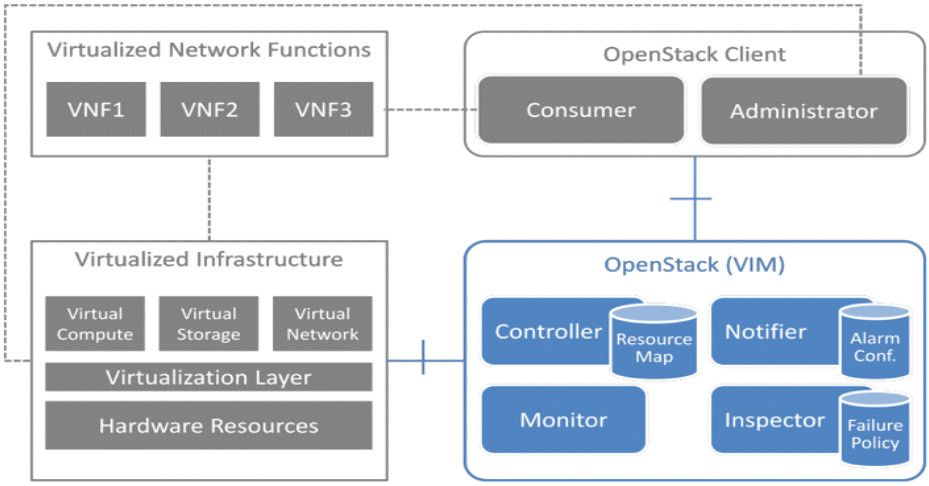

그림 7. 요구사항에 따른 기능블록

DOCTOR 프로젝트에서는 VIM안에 상태 관리를 위한 Monitor 기능, 받은 모니터링 정보를 가지고 결함을 결정하는 Inspector, 물리 리소스와 가상 리소스에 대한 매핑 테이블을 관리하는 Controller, 이벤트를 발생시키는 Notifier 등의 기능 블록을 정의하였으며 이에 따라 관련 오픈소스들도 함께 언급하였다.

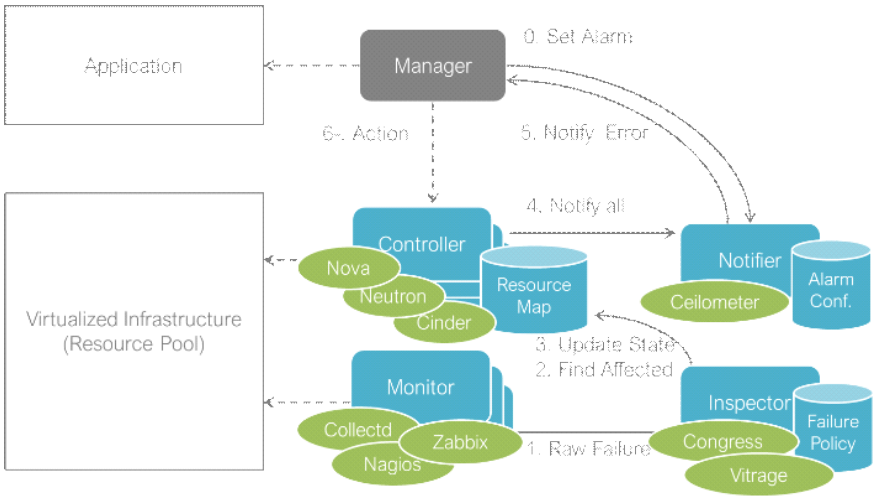

그림 8. 기능 블록별 관련 오픈소스 및 절차

먼저 현재 리소스를 모니터링 하는 기능은 Zabbix, Nagios 등과 같은 오픈소스 모니터링 프로젝트들이 해당된다[8][9]. 이를 통해 리소스에 대한 모니터링을 실시하고 문제가 발생하는 경우 Inspector에게 전달한다. Inspector는 수집된 정보를 바탕으로 오류를 결정하고 그에 따른 액션이 수행되도록 특정기능에게 전달하는 역할을 한다. Inspector에 대표적인 예로는 OpenStack 프로젝트의 Congress 와 Vitrage가 있다[10][11]. Congress는 사용자 요구에 따라 유연하고 동적인 정책을 수행시키는 정책 엔진 기능을 제공하는 프로젝트이다. Monitor로부터 장애 정보가 오면 등록된

정책에 따라 장애 정보를 평가하고 컨트롤러에게 장애 정보를 업데이트 하도록 메시지를 전달한다. Vitrage 프로젝트 역시 OpenStack 프로젝트로 RCA(Root Cause Analysis)를 목표로 한다. 이를 위해 다양한 모니터링 기능과 연동되어 있다. DOCTOR 구조에서는 장애 알람 정보를 기반으로 RCA를 분석하여 최종 장애 원인에 대한 정보를 컨트롤러에게 전달하는 기능을 제공한다. Controller는 Inspector에게 받은 정보를 바탕으로 리소스 맵을 통해 실제 영향이 미치는 기능에 대한 정보를 확인하고 해당 정보를 업데이트 한다. Controller에는 Nova, Cinder, Neutron등이 해당된다. 이처럼 정의된 블록에 따라 기능들이 구현이 되는데, 정의된 기능들은 실제 관련 오픈소스 프로젝트를 통해 구현된다. 대표적인 예로 Notifier가 있다. Notifier는 기존에 등록된 알람에 따라 이벤트를 발생시키는 역할로, OpenStack Telemetry 프로젝트인 Ceilometer 통해 개발되었다. DOCTOR 프로젝트에서는 기존의 Ceilometer의 경우 발생하는 지연을 줄이기 위해 새로운 Notification agent를 구성하였다[12].

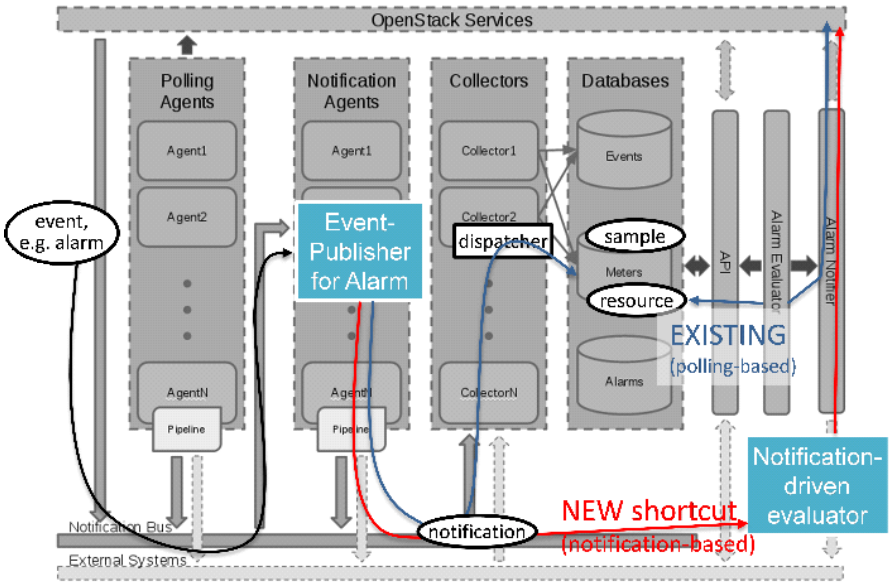

그림 9. Ceilometer에서의 DOCTOR 기능 구현

그림 9는 DOCTOR에서 기능을 정의하고 구현된 Ceilometer의 기능이다. 그림 9와 같이 기존에는 Collector와 Polling-based Alarm Evaluator 컴포넌트를 거쳐서 데이터가 전달되어 지연이 발생하였다. 이를 줄이기 위해 DOCTOR 프로젝트에서는 새로운 컴포넌트인 “Notification-driven Alarm Evaluator(NAE)” 구성하여 추가적인 지연을 줄였다. NAE는 Ceilometer의 “Alarm Evaluator”와 비슷하지만 알람 메시지를 위해 데이터베이스에서 주기적으로 데이터들을 가져오는 것이 아니라 알림 메시지를 통해서 NAE가 동작을 시작하게 하여 지연을 줄이는 역할을 한다.

2.2. OpenStack Fenix 프로젝트

OPNFV Doctor프로젝트는 OPNFV 초기 버전부터 진행된 프로젝트로 NFV 환경에서 오류 관리 및 OPNFV DOCTOR 프로젝트에서는 앞서 언급한 것처럼 기능을 정의하고 실제 오픈소스에 코드를 구현한다. OPNFV Arno를 시작으로 현재 진행중인 Gambia 버전까지 지속적으로 활발하게 프로젝트가 진행되고 있다. 이전 버전인 Euphrates에서는 두가지 유즈케이스 중 하나인 오류 관리에 대한 구현이 완료되었다. Fraser에서는 DOCTOR의 구성 기능으로 앞서 언급한 Congress, Vitrage가 포함되었으며, Neutron에게 현재 발생한 오류에 대해 알려줄 수 있는 기능도 추가되었다. 최근에는 Rolling upgrade, Maintenance and Scaling with Zero impact to VNF라는 주제로 새로운 논의가 진행되고 있다. 이 주제는 OPNFV DOCTOR의 후속 주제이자 OpenStack의 Fenix 프로젝트의 주요 내용이기도 하다. OpenStack Fenix에서는 롤링 인프라 유지 관리, 업그레이드 및 확장을 목표로 하고 있다[13]. Fenix 프로젝트에서는 이를 위해 크게 3가지를 목표로 하고 있다. 첫번째는 플러그인 형태의 Maintenance 관리를 목표로 한다. 현재는 OpenStack을 기준으로 여러가지 기능 구성 및 테스트를 진행하고 있지만 최종목표는 여러 플랫폼에서 이용될 수 있는 구조를 고려하고 있다. 이를 위해 General API, General Notification, Pluggable 한 구조를 목표로 한다. 두번째 목표는 Rolling upgrade, Maintenance, Scaling이다. Fenix를 통해 호스트 및 서비스에 대한 업그레이드나 스케일링을 서비스 중단없이 하는 것을 목표로 하고 있다. 마지막 목표는 표준화된 기능과의 연동이다. Fenix에서는 모든 절차를 ETSI에서 정의한 VNFM/EM과 같은 컴포넌트와의 연동을 고려하여 설계하고 있다. Fenix 프로젝트의 구조는 그림 10과 같다.

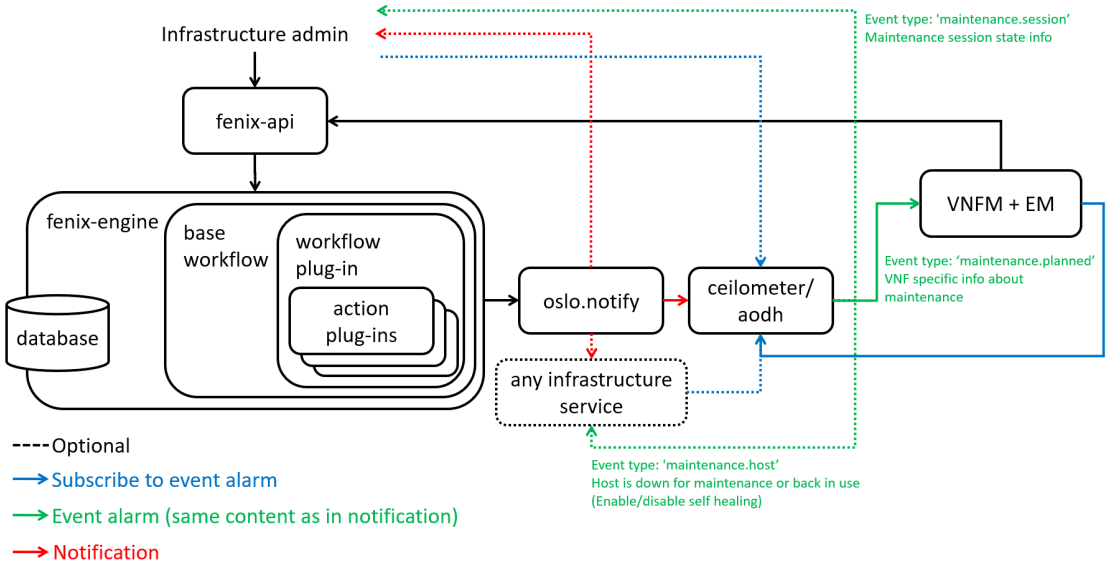

그림 10. Fenix 컴포넌트 및 동작 구조

그림10은 Fenix의 기본 컴포넌트 및 OpenStack과의 연동 구조를 나타내고 있다.

Fenix는 Fenix API, Fenix-Engine, Base workflow plug-in, Action Plug-in등으로 구성된다. Fenix API는 유지 관리를 위한 워크 플로우 세션을 구성하고 관리자와 통신하기 위한 API를 제공하는 역할을 하며 Fenix engine은 유지관리 워크플로 세션을 실행하고 데이터 베이스에 정보들을 저장하는 역할을 한다. Base workflow는 각 유지 보수 세션에서 사용되는 제공되는 기본 절차들을 나타내는데 Base workflow는 플랫폼에 따라 달라 질 수 있다. Action plug-in은 Base workflow로부터 호출되는 기능을 제공하는 것으로 여러 개의 플러그인을 가질 수 있다. 위와 같은 Fenix 컴포넌트는 여러가지 플랫폼에 적용될 때 공통으로 적용된다. 현재 OpenStack 환경에서는 유지 보수에 대한 세션 관리를 Ceilometer와 Aodh를 통해 시작을 하고 있다. 'maintenance.host' 와 'maintenance.planned' 그리고 'maintenance.session' 등의 메시지를 정의하여 OpenStack에서 사용되는 메시지 버스를 통해 전달 시켜 유지보수 환경을 관리하고 있다.

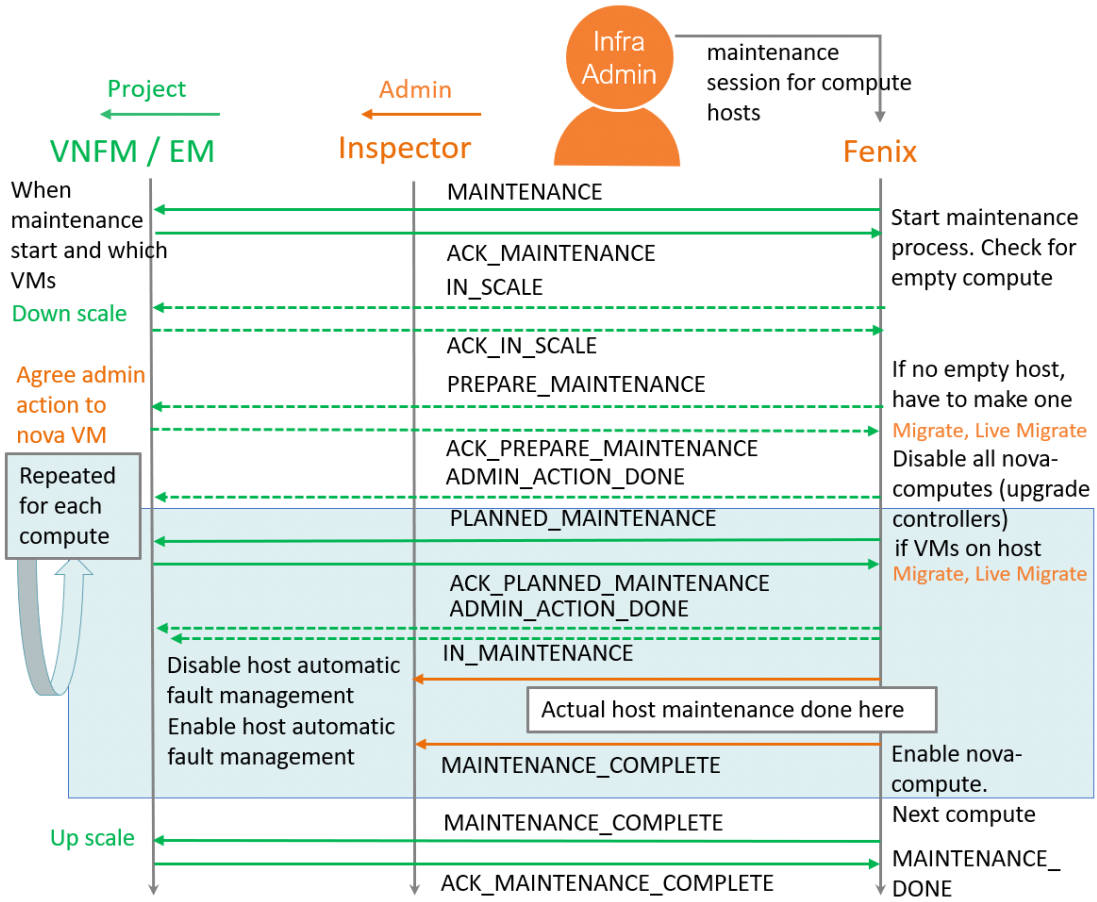

그림 11. OpenStack에서의 Base workflow

그림11은 OpenStack환경을 기반으로 구성된 기본 워크 플로우이다. 기본 워크 플로우는 Fenix와 인프라 관리자인 VNFM과의 통신으로 구성되며, 관리자의 유지관리 요청에 의해 통신이 시작된다. 그림11에서의 기본 절차에서는 호스트 업그레이

드를 위해 공간을 확보하고 서비스를 이전하는 등의 절차를 담고 있지만, 기본적인 절차는 정책에 따라 달라질 수 있다. Fenix 프로젝트에서는 OpenStack과의 연동 후에는 다양한 플랫폼과의 연동을 통해 독립된 유지 관리 플랫폼을 구성하는 것을 목표로 하고 있다.

3. IoT 클라우드 서비스의 가용성 제공을 위한 분석

3.1. IoT 클라우드 서비스의 가용성

이전까지 NFV 환경에 고가용성을 제공하기 위한 다양한 방법과 프로젝트를 살펴보았다. 이처럼 클라우드 환경에 가용성 기술을 효율적으로 적용하기 위해서는, 제공하고자 하는 클라우드 환경과 서비스의 특성에 따라서 알맞게 가용성 모델을 적용할 필요가 있다.

대표적으로 IoT 클라우드 환경에는 서비스의 분야에 따라 다양한 특성들이 존재한다. 예를 들어, V2V (Vehicle-to-vehicle)나 비디오 기술을 활용한 방범 시스템처럼 실시간성 데이터 송수신 기능이 필요한 서비스의 경우, 가용성을 제공하기 위해서는 실시간성 데이터의 송수신을 위한 최단시간 장애 복구가 필수적이므로, Redundancy 형태의 장애 복원 모델을 이용해야 한다. 반면에, 스마트 그리드 용 서비스는 지속적인 데이터 수신은 이루어지나, 실시간성은 낮으며 데이터의 변화량이 크지 않다는 특징이 있어 데이터의 송수신 빈도도 낮은 편에 속한다. 이러한 서비스의 경우, 효율적인 가용성 제공을 위해서는 이중화 모델보다는 자동 복원의 기능을 사용하는 것이 더 알맞으며, 서비스 계층에서는 데이터를 받아서 다시 돌려주지 않기 때문에, 물리적 계층에 대한 강화된 장애 탐지 기능이 제공되어야 한다. 추가적으로 물리적인 리소스 자체가 제약되어 있는 Edge 클라우드 환경까지 고려하게 된다면 문제는 더 복잡해진다.

이처럼 클라우드 환경에서는 NFV의 가용성을 제공하기 위해 해당 서비스가 제공되는 환경과 서비스의 특성에 맞춰 장애 탐지 및 장애 복원 모델이 다르게 적용되어야 한다. 이를 위해서는 서비스마다의 가용성 제공을 위한 요구사항을 분석하여야 하며, 그에 따른 자동화된 장애 탐지 및 복원 기능이 제공되어야 한다.

이에 본 3장에서는 앞서 알아본 다양한 고가용성 기능들과 더불어 IoT 클라우드 환경의 서비스 특성을 반영한 가용성 제공 구조를 설계하고, 이에 대한 성능을 측정한 것을 보인다.

3.2. 대표적인 IoT 클라우드 서비스 특성에 따른 분석

3.2.1. 서비스 특성에 따른 분석을 위해 정의한 요소

서비스의 특징을 분류하기 위해서 본 고에서는 다음과 같은 요소를 정의하였다.

- **Latency** – 실시간성 데이터 처리가 보장되어야 하는 서비스인가?
- **Data Processing** – 데이터의 처리량이 많아, 고성능의 컴퓨팅 기능이 필요한 서비스인가?

- **Service Continuity** – 서비스의 성격이 Stateful한가? 또는 Stateless한가?
- **Data Storage** – 데이터의 저장을 위해 별도의 외부 스토리지가 필요한 서비스인가?
- **Network Size** – 서비스가 제공되는 네트워크 환경의 규모는 어느 정도인가?
- **Location** – 서비스의 위치가 리소스가 제한적인 Edge 클라우드인가? 아니면 리소스가 풍부한 Core 클라우드인가?

위와 같은 기준에 따라 서비스의 특징을 분류하였으며, 분류한 결과는 특수한 환경보다는 일반적인 환경을 가정하였다. 또한 이렇게 분류한 항목에 따라서 가용성을 제공하기 위한 요소인 모니터링의 타겟과 복원 플랜을 정의하였다.

모니터링 타겟은 인프라 환경 모니터링, 애플리케이션 모니터링, 데이터베이스 모니터링으로 분류하였다. 기본적으로는 각 인스턴스 내부에 데이터 및 정보를 저장하는 것으로 가정하고, 외부 스토리지를 사용할 경우, 모니터링 타겟에 포함하였다. 복원 플랜으로는 Redundancy 방법과 Failover로 분류하였으며, 서비스 특성에 따른 추가 요구사항도 함께 표기하였다.

3.2.2. Healthcare

Healthcare 서비스는 일반적인 사용자의 모니터링이 아닌, 환자를 위한 헬스케어 서비스로, 환자의 상태를 실시간으로 모니터링하고 상태 변화에 따라 빠르게 대처해야 하는 서비스이다. 따라서 실시간으로 데이터 처리가 이루어져야 하지만, 모니터링 값에 따라 그 다음 Data Processing이 진행되기 때문에 기본적으로는 Data Processing 양이 많은 편은 아니다. 그러나 장애가 발생하게 되면, 현재까지 처리된 데이터부터 다시 처리를 시작하여야 하기 때문에 기존의 정보는 유지되어야 한다. 환자의 건강 정보는 각 환자마다 개별 보관되어야 하기 때문에 외부 스토리지를 사용한다. 또한 실시간 관리 및 Latency를 줄이기 위해서는 Edge 클라우드 쪽에서 동작하여야 한다. 이와 같은 환경에서, Healthcare 서비스에 가용성을 제공하기 위해서는 리소스가 제한적인 것을 고려하여, Failover 형태의 복원 기술이 적합하지만, 데이터의 처리량이 적다는 특징과 Latency에 민감하다는 특징 때문에 Redundancy 방법이 더 알맞을 수 있다. 이때, 장애 감지를 위해서는 인프라 환경에 대한 모니터링과 애플리케이션 모니터링, 그리고 데이터의 저장을 위한 외부 스토리지에 대한 모니터링이 제공되어야 한다.

3.2.3. Smart Home and Metering

Smart Home과 Metering 서비스는 스마트 홈 환경을 제공하기 위해 스마트 홈에 설치된 센서들로부터 데이터를 수집하고, 가공하여 사용자에게 스마트 홈 환경을

제공하는 서비스이다. 스마트 홈 서비스에서 수집되는 데이터들은 주로 온도, 습도 등 현재 상태를 포함하는 데이터이기 때문에 Latency에 Critical하지는 않다. 또한 데이터를 처리하는 과정이 복잡하지 않고 데이터의 양도 변화가 크지 않기 때문에 서비스의 특성은 Stateless에 가깝다. 데이터의 경우, 크기가 작고 변화량이 적어 개별적인 스토리지를 별도로 사용하지 않아도 서비스의 제공이 가능하다. 스마트 홈 서비스는 수집되는 데이터 종류나 양이 다른 서비스에 비해 많은 편이기 때문에 네트워크 비용 절감을 위해서는 분산된 형태로 각 Edge에 저장되어야 한다. 이와 같은 환경에서 가용성을 제공하기 위해서는 제한적인 리소스 환경을 고려하여 Failover 형태의 복원 플랜이 적합하며, 동일한 인스턴스를 새로 생성하거나 기존의 인스턴스를 재사용하는 방식의 복원 기능이 제공되어야 할 것이다. 또한 장애 감지를 위해 인프라 환경과 애플리케이션에 대한 장애 모니터링이 제공되어야 한다.

3.2.4. Video Surveillance

Video Surveillance 서비스는 카메라의 역할을 하는 클라이언트로부터 실시간으로 영상 데이터를 수신하고 분석하여 데이터를 저장한다. 영상 데이터는 일반적인 데이터에 비해 크기가 큰 편이고, 주로 넓은 지역에 분포하고 있으므로 데이터의 저장을 위한 외부 스토리지를 별도로 사용하게 된다. 이 서비스는 Latency에 민감하고 데이터의 크기도 크기 때문에, 영상 데이터의 빠른 처리와 가공을 위해 주로 Edge에 배치한다. 그러나 영상 데이터가 지속적으로 유지되어야 하는 것은 아니므로, Stateful한 서비스라 보기는 어렵다. 이 상황에서 본 서비스에 가용성을 제공하기 위해서는 Redundancy 모델이 적합하지만, Edge 환경의 제한적인 리소스 상태를 고려하였을 때, ACT-STB 모델을 복원 플랜으로 사용할 수 있을 것이다. 또한 장애 감지를 위해서 인프라 환경, 애플리케이션 뿐만 아니라, 데이터베이스를 이용하는 경우에는 DB에 대한 장애 탐지 및 감지 기능도 요구된다.

3.2.5. Automotive and Smart Mobility

Automotive and Smart Mobility 서비스는 차량 간의 센싱 및 네트워킹을 위한 길가 인프라 시설과의 통신 및 데이터 처리를 위한 서비스를 제공한다. 이 서비스는 빠르고 동적인 데이터의 처리가 특징이기 때문에 다른 서비스에 비해 Latency에 있어 더욱 민감한 편이다. 또한 여러 정보를 수집하여 계산과정을 통해 결과 값을 전달하거나 예측을 위한 계산 등이 필요하기 때문에 Data Processing이 높은 편이며, 기존 단말과의 지속적인 통신을 위해 서비스의 State (상태) 가 유지되어야 한다. 이 서비스는 Latency를 줄이기 위해 통신을 위한 유닛이 여러 곳에 분포되어 있다. 다시 말해, 서비스의 배치 장소는 Edge 클라우드에 근접하지만, Latency에 민

감하고 Data Processing의 양과 State 유지를 위해서는 ACT-ACT와 같은 Redundancy 모델이 복원 모델로 적합하다. 또한 장애 탐지를 위해서는 인프라 환경과 애플리케이션에 대한 모니터링 기능이 제공되어야 할 것이다.

3.2.6. Smart Energy and Smart Grid

Smart Energy 및 Smart Grid 서비스는 에너지 소비 및 분배를 효율적으로 할 수 있도록 환경을 제공하고 관리하는 서비스이다. 시스템의 규모가 다른 서비스에 비해 큰 편이지만, Latency에 민감하지 않고, Data Processing 양도 많은 편은 아니다. 하지만 여러 단말로부터 수집되는 데이터의 속도가 각기 다르기 때문에, 장애 발생 시 데이터가 분실될 수 있다. 다만 데이터를 주기적으로 수신하기 때문에 일부 손실이 발생하더라도 그 다음 데이터를 통해 서비스를 제공할 수 있다는 특징이 있다. 이 서비스는 규모가 크고 데이터의 사이즈 작아, 코어 클라우드를 통해 제공할 수 있다. 이 서비스의 경우, 가용성 제공을 위해서는 Redundancy 방법을 사용할 수 있다. 서비스의 특성에 따라서 ACT-STB 복원 모델도 사용은 가능하지만, 리소스가 충분하다면 ACT-ACT 형태의 복원 모델도 사용할 수 있을 것이다. 자원의 절약을 위해서는 상기 모델들이 아닌 Failover 형태의 복원 방법을 사용할 수도 있다. 또한 장애 탐지를 위해서는 인프라 환경과 애플리케이션에 대한 모니터링 기능이 제공되어야 할 것이다.

3.2.7. Smart Logistics

Smart Logistics 서비스는 물품이 원산지로부터 소비자에게 전달되는 모든 과정에 대한 관리를 단순화하며 물건을 배달하는 운송 수단에 대한 관리 기능을 제공하는 서비스이다. Smart Logistics 서비스는 넓은 환경에서 발생하는 다양한 데이터를 수집하고, 이렇게 수집된 데이터를 자동화된 알고리즘을 통해 다양한 의사결정 기능을 제공하기도 한다. 따라서 Smart Logistics 서비스는 Latency에 민감한 편은 아니다. 그러나 수집된 데이터를 취합하여 Data Processing을 하며, 하나의 단말에 대한 서비스가 종료될 때까지 해당 데이터의 값이 재사용되므로, Stateful한 서비스 특성을 갖는다. Smart Logistics 서비스는 넓은 네트워크 환경에서 데이터를 수집하고, 프로세싱하기 때문에 서비스는 코어 클라우드에 위치하게 된다. 그러므로 Smart Logistics 서비스를 위한 가용성 제공을 위해서는 ACT-ACT와 같은 Redundancy 방법이 이용되어야 하며, 장애 탐지를 위해서는 인프라 환경과 애플리케이션에 대한 모니터링 기능이 제공되어야 할 것이다.

3.2.8. Environmental Monitoring

Environmental Monitoring 서비스는 넓은 환경을 실시간으로 모니터링하여 '지능형 탐지', '지능형 재배', '지능형 산림 관리' 등의 서비스를 제공할 수 있다. 또한 Environmental Monitoring 서비스는 특정 데이터에 대한 실시간 데이터 스트리밍 서비스를 제공할 수 있다. 즉 실시간 모니터링 서비스를 제공하기 위해 Latency가 어느 정도 보장되어야 한다. 다만, Data Processing의 양이 많은 편은 아니다. 뿐만 아니라, 넓은 환경에서 다양한 데이터가 수집되어야 하기 때문에, 데이터 스토리지를 개별적으로 사용할 필요가 있으며, 일부의 서비스를 제공하기 위해서는 Stateful 한 서비스 환경도 고려하여야 한다. 이 서비스는 앞서 언급한 것처럼 넓은 지역에서 다양한 데이터를 수신해야하기 때문에 코어 클라우드에 위치하게 된다. 그러나 IoT 단말의 특성에 따라서 Edge 클라우드에도 함께 배치될 수 있다. 본 서비스에 가용성을 제공하기 위해서는 ACT-ACT 모델과 같은 Redundancy 방법이 복원 모델로 적합할 것이다. 그리고 장애 탐지를 위해서는 인프라의 환경과 애플리케이션 및 데이터베이스에 대한 모니터링 기능도 함께 제공되어야 할 것이다.

4. 결론

본 문서에서는 NFV의 고가용성을 위한 기술의 동향을 분석하였다. NFV 환경에서의 고가용성 연구는 앞서 논의된 다양한 Opensource 프레임워크를 통해 지속적인 논의가 진행되고 있다. HA 프로젝트에서 정의된 다양한 유즈케이스는 고가용성 제공을 위한 고려사항들을 정의하였으며 DOCTOR에서는 실제적인 가용성 제공 프레임워크를 위한 기능을 정의하고 정의된 기능을 구현하였다. 특히, DOCTOR 프로젝트에서 개발된 결과들은 실제로 Upstream 프로젝트에 반영되어 사용되고 있다. 또한 본 문서에서는 다양한 IoT 클라우드 서비스를 비교 및 분석하여, 각 서비스에 최적화된 가용성 모델과 복원 방법, 모니터링 타겟을 소개하였다.

최근에는 호스트의 실제적인 유지 관리를 위한 구조를 제시하고, 이를 기반으로 데모를 시연하였으며 관리자를 위한 새로운 툴 개발을 통해 실제적인 서비스 가용성 제공방안을 보이기 위한 논의가 진행중이다. 서비스의 가용성을 높이기 위한 다양한 연구들은 기존 클라우드 기반 서비스 사업자들 뿐만 아니라, IoT 클라우드 기반의 서비스를 제공하는 통신사업자 및 다양한 서비스 사업자에게 많은 도움이 될 것으로 예상된다.

References

[1] “Network Functions Virtualisation (NFV); Architectual Franework”, ETSI GS NFV 002 v1.2.1, December 2014.

[2] “Network Functions Virtualisation (NFV); Reliability;Report on Models and Features for End-to-End Reliability“, ETSI GS NFV-REL 003 V1.1.1 April,2016.

[3] Openstack : https://www.openstack.org/

[4] OPNFV: https://wiki.opnfv.org/display/PROJ/ Project+Directory (Accessed 19.10.07).

[5] HA: https://wiki.opnfv.org/display/availability/High+Availability+For+OPNFV (Accessed 19.10.09).

[6] Doctor: https://wiki.opnfv.org/display/doctor (Accessed 19.10.09).

[7] Yardstick : https://wiki.opnfv.org/display/yardstick/Yardstick (Accessed 19.10.14).

[8] zabbix: https://www.zabbix.com/ (Accessed 19.10.14).

[9] Nagios : https://www.nagios.org/ (Accessed 19.10.14).

[10] congress: https://wiki.openstack.org/wiki/ Congress (Accessed 19.10.23).

[11] vitrage: https://wiki.openstack.org/wiki/ Vitrage (Accessed 19.10.23).

[12] ceilometer: https://wiki.openstack.org/wiki/ Telemetry (Accessed 19.10.23).

[13] Fenix: https://wiki.openstack.org/wiki/Fenix (Accessed 19.10.23).

[14] H., Zhu; C., Huang; Cost-Efficient VNF Placement Strategy for IoT Networks with Availability Assurance. IEEE 86th Vehicular Technology Conference (VTC-Fall), 2017; pp. 1-5

K-ONE 기술 문서

- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.

- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  (Homepage: http://opennetworking.kr/projects/k-one-collaboration-project/wiki, E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2019/11