Opendaylight

How to build opendaylight based applications?

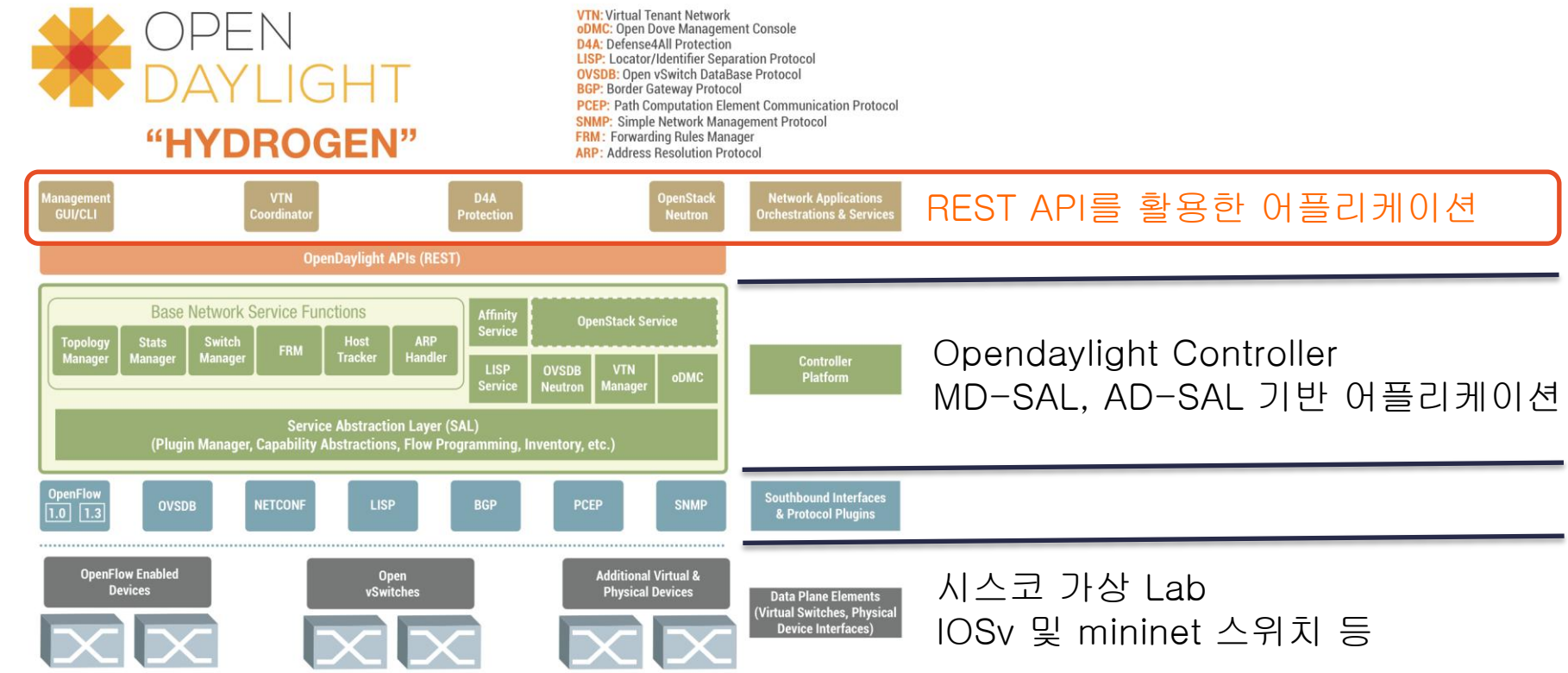
김형수 부장
Solution SE

2014년 9월 25일

Application Development Guide

- Opendaylight VM 구성하기
- Northbound REST API 살펴보기
  - Switch Manager, Connection Manager, User Manager, Bridge Domain
  - Static Routing, Host Tracker, Flow Programmer, Subnets, Statistics
- Python REST Sample Client

Codefest 프로그램밍 영역



Opendaylight VM 구성하기

- Opendaylight controller 와 mininet 으로 구성된 VM 구성

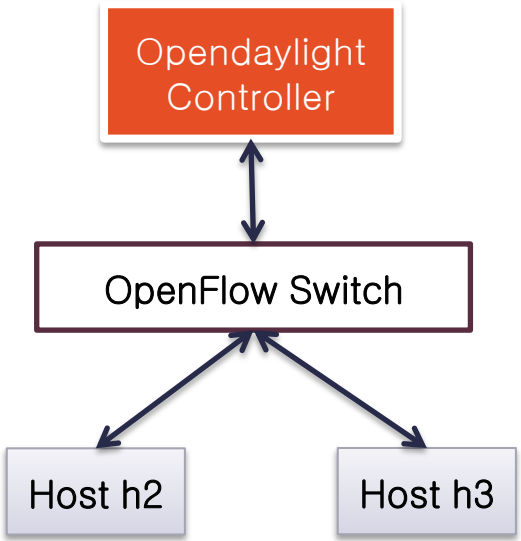
  https://wiki.opendaylight.org/view/CrossProject:Integration_Group:Test_VMs
- VM내 구성 내역
  - Opendaylight 3개 버전(base, virtualization, service provider)
  - Mininet 2.1.0 & ovs 2.0.0
  - Integration tests
  - VTN Coordinator
- 구성후 self test shell로 정상동작 확인

Mininet 소개

- Mininet은 PC 환경에서 가상 네트웍을 만들 수 있는 Network Emulator
- Mininet은 end-host, switch, router 기능을 emulation 해준다
- 다양한 Topology 구성이 가능하며
- Linux 기반에서 동작하는 실제 프로그램을 Mininet과 연동해서 사용 가능하다
- 단점으로 Linux 기반에서만 동작하며
- Single host 구성만 가능해서 성능상의 제약 사항이있다

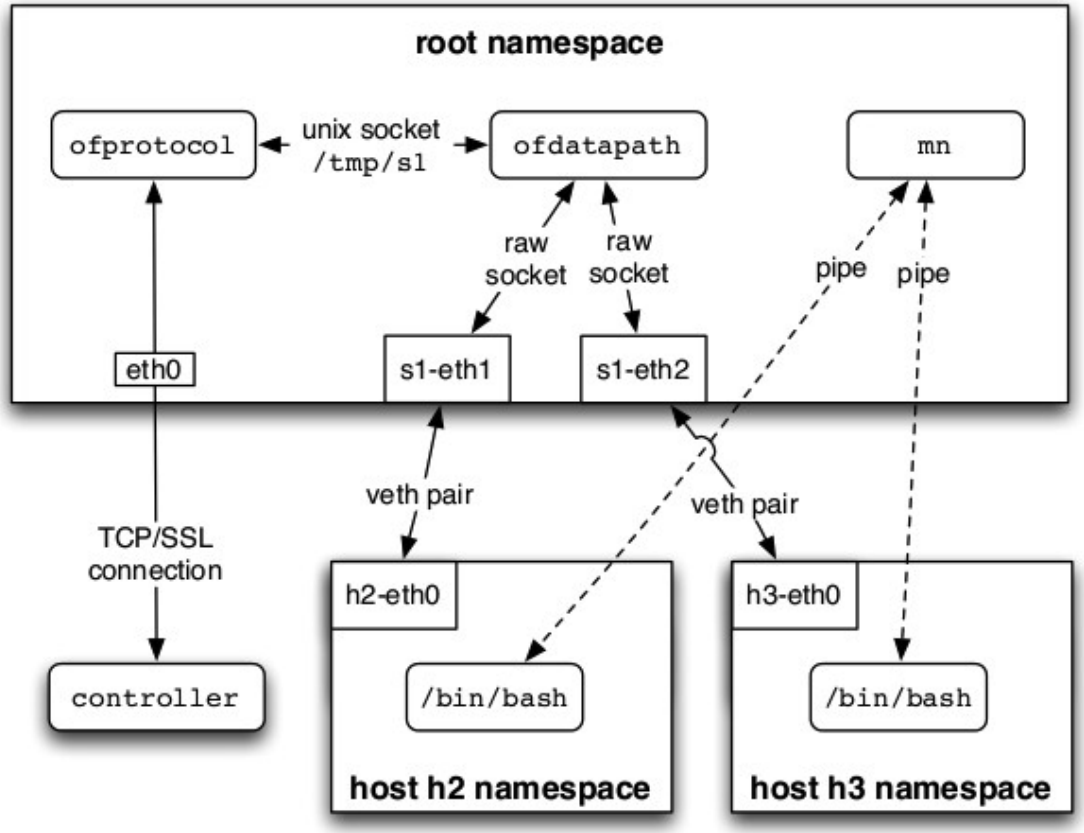
https://github.com/mininet/mininet/wiki/Documentation

Mininet Architecture



Logical View

Physical View



REST API

- HTTP 와 JSON/XML로 구성된 웹서비스
  - GET http://host/api/user/2423 (사용자 2423에 대한 정보 조회)
- Data관점에서 보면 데이터 생성/조회/삭제/갱신 동작을 HTTP의 PUT/GET/DELETE/POST를 이용해 구현
- JSON(JavaScript Object Notation)을 도입해서 XML 보다 간단하게 데이터를 표현, JavaScript와의 연동을 편리하게 제공
- 상대적으로 복잡하고 어려운 SOAP에 대안
- 다양한 언어 Java, .NET, Python, JavaScript에서 쉽게 사용할수 있는 library 존재

Controller Northbound API

- Controller에서 제공하는 Base Network Service Function REST API

https://wiki.opendaylight.org/view/OpenDaylight_Controller:REST_Reference_and_Authentication

- HTTP Basic Authentication 제공 (controller에서 생성한 local 사용자)
- curl 사용시
  - curl –u username:password http://controller_ip:8080/{rest_api} –H "Content-Type: application/xml" –X {http mothod, GET,PUT,DELETE}
  - Data POST 시 --data @{file_name} 사용
- Chrome POSTMAN 사용

Topology

- Controller에서 관리하는 네트웍 디바이스의 topology 정보를 제공하는 API
- GET /controller/nb/v2/topology/{containerName}
  - 지정된 container에 속한 device 정보와 상호 연결된 정보 출력
  - 해당 응답을 분석해서 device간 연결 관계를 출력
- PUT/DELETE /controller/nb/v2/topology/{containerName}/userLink/{name}
  - 사용자 링크 생성 및 삭제
- GET /controller/nb/v2/topology/{containerName}/userLinks
  - 사용자 링크 정보 조회

Switch Manager

- Nodes, NodeConnector 및 해당 속성정보 조회/생성/삭제
- GET /controller/nb/v2/switchmanager/{containerName}/nodes
  - 모든 node 및 node의 속성정보 조회
- GET /controller/nb/v2/switchmanager/{containerName}/node/{nodeType}/{nodeId}
  - 지정된 node의 nodeconnector 및 속성정보 조회
- PUT /controller/nb/v2/switchmanager/{containerName}/node/{nodeType}/{nodeId}/property/{propertyName}/{propertyValue}
  - 지정된 node에 속성정보를 추가

User Manager

- 사용자 등록 및 삭제 기능
- POST /controller/nb/v2/usermanager/users
  - 신규 사용자 생성 user, roles, password 3개 정보 입력
- DELETE /controller/nb/v2/usermanager/users/{userName}
  - 지정된 사용자를 삭제

Container

- Multi-tenant를 제공해주는 개념
- 별도로 container를 생성하지 않으면 default 로 존재
- 네트웍 장비들은 default container에 연결되어 제공됨

Static Routing

- Static Routing 경로를 추가/삭제 및 조회
- GET /controller/nb/v2/staticroute/{containerName}/routes
  - 해당 container에 존재하는모든 static routes 정보 조회
- GET/PUT/DELETE /controller/nb/v2/staticroute/{containerName}/route/{route}
  - 해당 container에 필요한 static routes 정보 추가/삭제

```
<staticRoute>
    <name>route-1</name>
    <prefix>10.10.1.0/24</prefix>
    <nextHop>1.1.1.1</nextHop>
</staticRoute>
```

Connection Manager

- Controller에 연결된 node 관리
- GET /controller/nb/v2/connectionmanager/nodes
  - 해당 controller에 연결된 모든 node 조회
- DELETE /controller/nb/v2/connectionmanager/node/{nodeType}/{nodeId}
  - 지정된 node를 제거 (node id와 type을 지정)
- PUT /controller/nb/v2/connectionmanager/node/{nodeId}/address/{ipAddress}/port/{port}
  - 해당 node에 사용자지정 이름, IP주소, management tcp port를 설정

Python REST client snippet (using requests module)

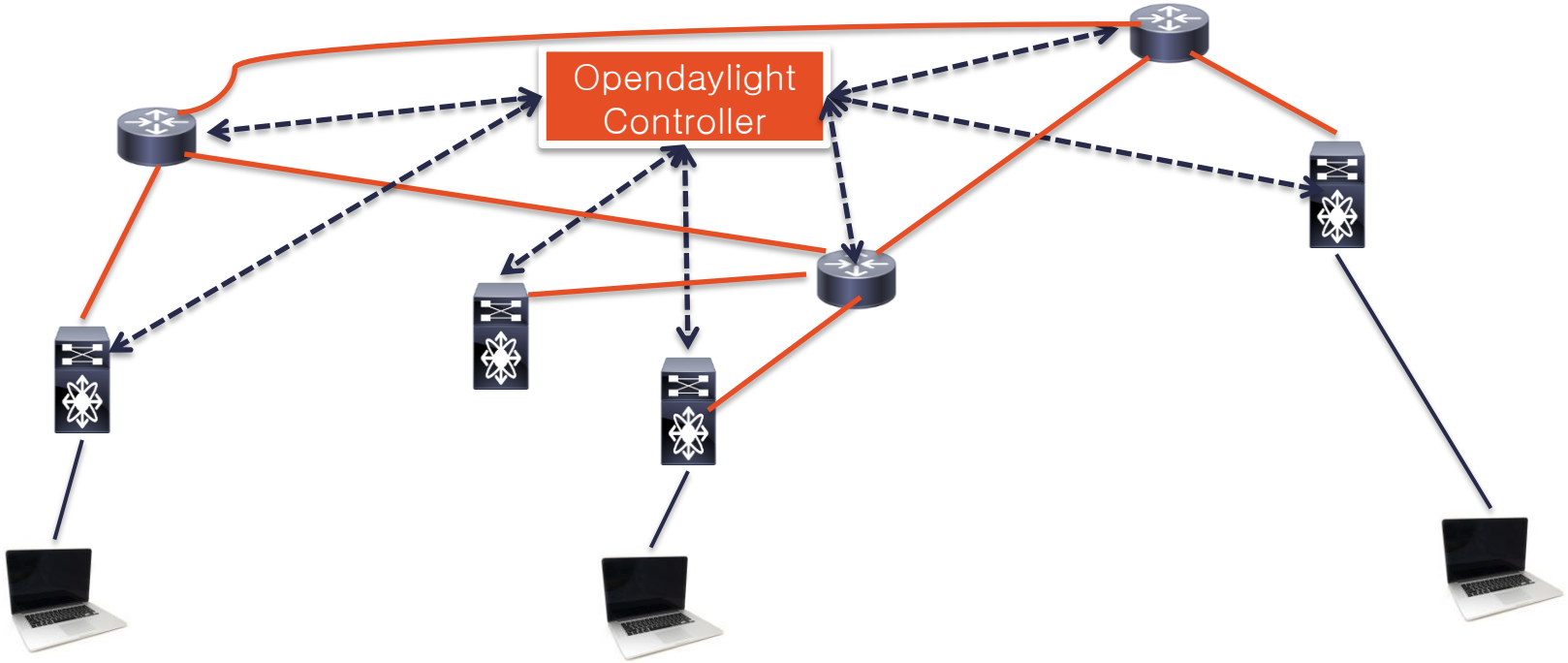
```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import json
import requests

def get_nodes(base_url, user, pwd):
    resp = {}
    r = requests.get(base_url, auth=requests.auth.HTTPBasicAuth(user,pwd))
    if r.status_code == 200 and r.headers['Content-Type'].lower() == 'applicaiton/json':
        resp = json.loads(r.content)
        return resp
    else:
        return None


def post_data(base_url, data, user, pwd):
    resp = {}
    r = requests.post(base_url, data=data, auth=requests.auth.HTTPBasicAuth(user,pwd))
    if r.status_code == 200 and r.headers['Content-Type'].lower() == 'applicaiton/json':
        resp = json.loads(r.content)
        return resp
    else:
        return None
```

가상 네트웍 Topology 구성 안



참고할 만한 자료

- Opendaylight : http://www.opendaylight.org
- Opendaylight 개발자 wiki : https://wiki.opendaylight.org/
- 가상 네트웍 관련 : http://mininet.org/
- Community Labs : http://www.opendaylight.org/developers/community-labs
- Python : http://www.python.org
- Python Package Index : http://pypi.python.org

Thank you.