Ryu

Página do projeto

www.osrg.net/ryu

- Network opearting system
- OVS, openflow switch

Sobre o código

Linguagem

Python 2.7

Licença

Apache 2.0 license

Suporte a versões do OpenFLOW

- 1.0 with Nicira Extensions
- 1.2
- 1.3

Requisitos

- ambiente Python 2.7
  - python-gevent python-routes python-webob python-paramiko

Independência de linguagem de programação

Comunicação por mensagens, então, pode conversar com qualquer aplicativo que implemente essas mensagens.

Modularidade

???

Repositório

git://github.com/osrg/ryu.git

Recursos básicos do controlador

- GRE tunneling
- VLAN support
- IPv6
- topology discovery
- MAC based segregation

Aplicações

https://github.com/osrg/ryu/tree/master/ryu/app

- cbench.py
- client.py
- conf_switch_key.py
- gre_tunnel.py
- ofctl_rest.py

- quantum_adapter.py
- rest.py: app/rest
- rest_conf_switch.py
- rest_firewall.py
- rest_nw_id.py
- rest_quantum.py
- rest_router.py
- rest_topology.py
- rest_tunnel.py
- simple_isolation.py
- simple_switch.py
- simple_switch_12.py
- simple_switch_13.py
- simple_switch_lacp.py
- simple_vlan.py
- tunnel_port_updater.py
- wsgi.py

Testes

https://github.com/osrg/ryu/tree/master/ryu/tests

- app
- lib
- ofproto
- packe
- sample

Disponibilidade nas distribuições GNU/Linux

Pode ser instalado com o gerenciador 'pip'.

Último release / commit

- Aplicações: "a month ago"
- Testes: "a day ago"
- Core: "a day ago"

Documentação

Documentação oficial

http://ryu.readthedocs.org/en/latest/index.html

Tutoriais

https://github.com/osrg/ryu/wiki/OpenFlow_Tutorial

Suporte junto ao OpenStack

Quantum Ryu plugin. VM pré-configurada para testar OpenStack + Ryu.

https://github.com/osrg/ryu/wiki/OpenStack

- Flat networks regardless of the underlying physical network
- Scalable multi-tenant isolations
  - tunneling based isolations
  - VLAN does not scale larger than 4096
  - do not need high-end switches
- Future items:
  - integration with quantum IPAM and L3 API
  - firewall
  - virtual network <--> physical network
  - distributed controllers (redundancy)

Communication

'nova requests quantum-server for network configuragion'

OVS <--> Ryu: openflow

Ryu agent <--> Ryu: REST API

Quantum server <--> Ryu: REST API

Ryu agent <--> Quantum db: db access

VIF driver <--> quantum: Quantum API
linux_net driver <--> server: Quantum API

Network Creation (network-node = nova-network + linux_net driver + OVS)

1. to nova-network: create netork
2. nova network --> quantum server: create net
3. quantum server --> Ryu: network <uuid>, <tunnel key>
4. quantum server --> nova network: <uuid>
5. nova network --> linux_net driver: plug
6. linux_net driver --> OVS: create gw-xxx
7. linux_net driver --> quantum server: tentant id, network id, dp id, port, mac
8. quantum server --> Ryu: network id, dp id, port, mac
9. ryu --> OVS: set flow entryies

Instance creation (compute-node = nova compute + VIF driver + OVS)

1. to nova compute: create instance
2. nova compute --> quantum server: create port
3. quantum server --> nova compute: uuid
4. nova compute --> VIF driver: plug
5. VIF driver --> OVS: create tap-xxx
6. VIF driver --> quantum server: (tenant_id, dp_id, port, mac)
7. quantum server --> Ryu: (network_id, dp_id, port, mac)
8. Ryu --> OVS: set flow entryies

Node boot up (compute/network-node = OVS + Ryu agent + ryu_v2.ini)

1. ryu_v2.ini --> Ryu agent: get ip address
2. Ryu agent --> quantum db: register (dp_id, ip_address)
3. quantum db --> Ryu agent: get list of (dp_id, ip_address, and polling)
4. Ryu agent --> OVS: create vport_grc to ip address
5. Ryu agent --> Ryu: register (dp_id, port_id, remote dp_id)

Quem usa

???

Quem está por trás

NTT Lab

Outras características