NetArbiter

-Multi-site Network Emulation -

Hee Won Lee

Cloud Platform Software Research

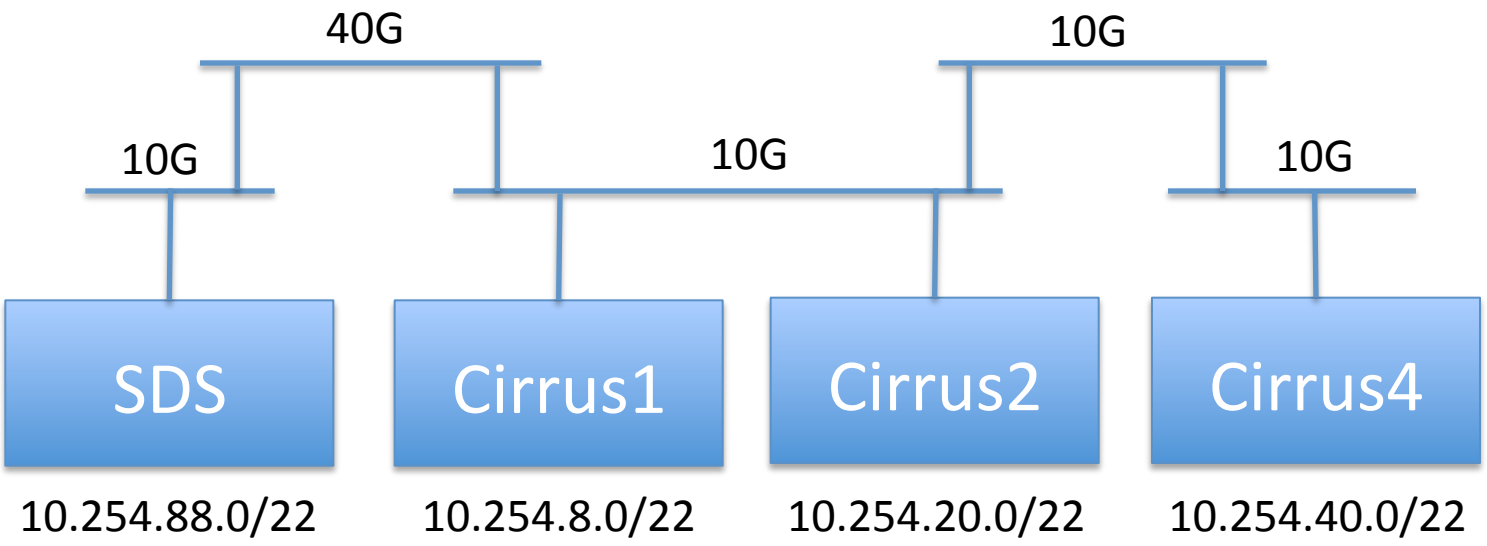
NetArbiter

INTRODUCTION

NetArbiter Project

- **Project Description**
  - Build a framework for multi-site network emulation
- **Goal**
  - Emulate network delay and bandwidth between geographically distributed cloud sites
  - Use DevOps for easy implementation and deployment

IP Address Assignment Scheme



- Other clusters
  - nimbus1: 10.254.28.0/22
  - agave: 10.254.0.0/22

Required Components

- Router
  - Will create virtual routers using **OpenStack Neutron**.
- Network emulator
  - Will use **TC** for rate control.
  - Will use **NETEM** for delay emulation.
- Configuration automation
  - Will use a DevOps platform called **Ansible**.

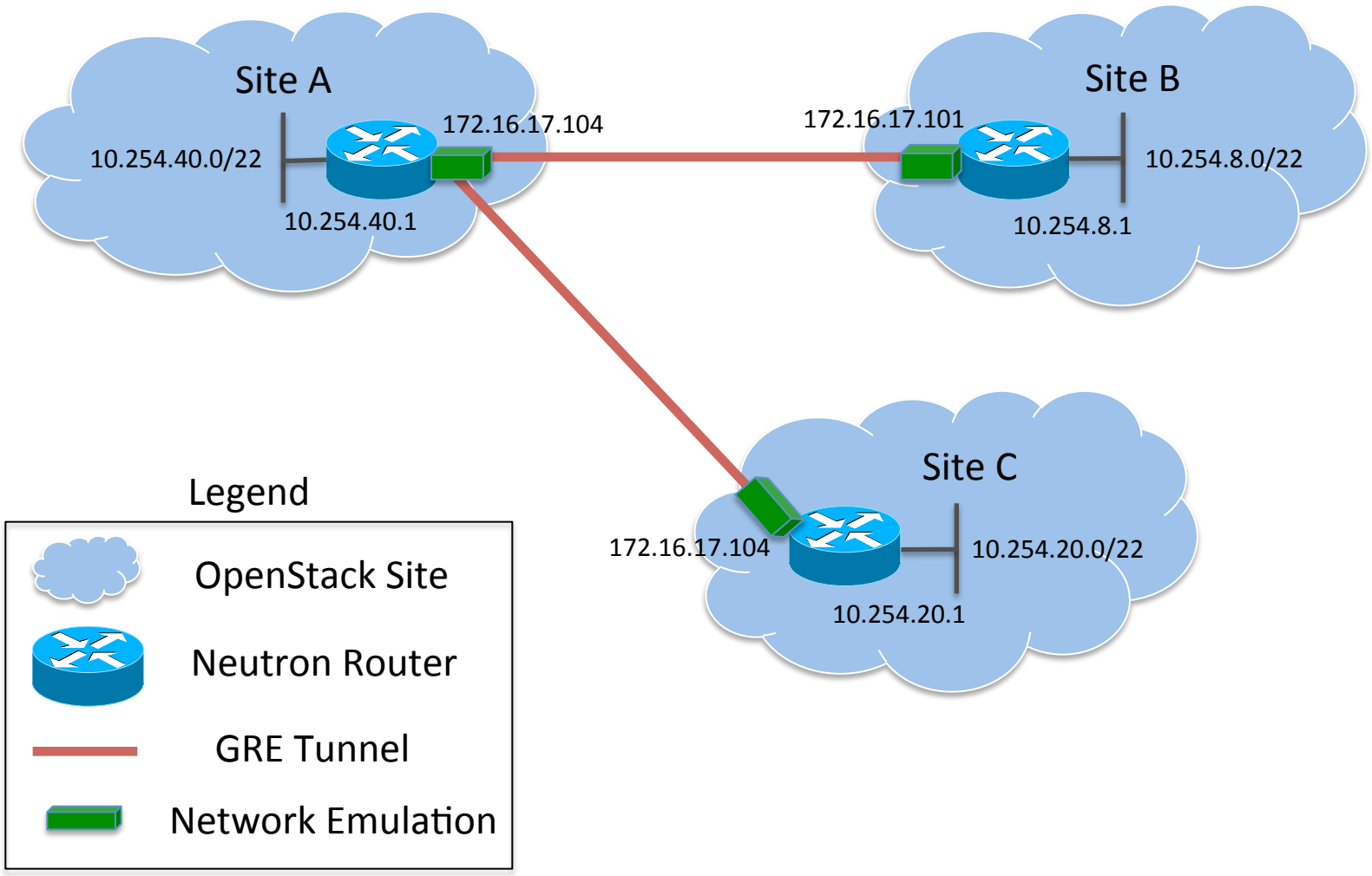
NetArbiter

DESIGN AND IMPLEMENTATION

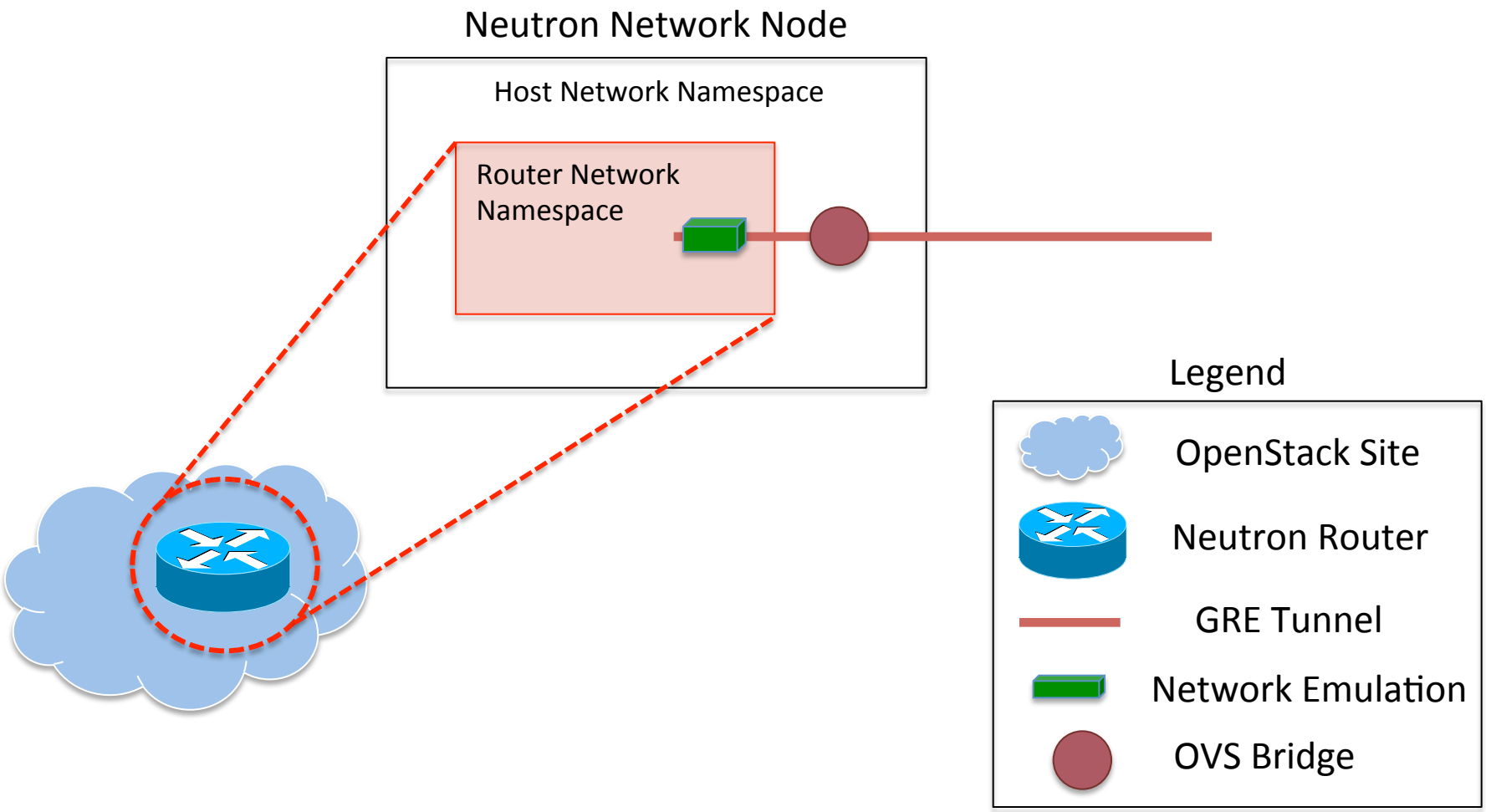
Design & Implementation

- Design
  - Shell scripts
    - Create routers
    - Create tunnels and routes
    - Add network emulation
  - Ansible Playbook
    - Run shell scripts remotely
- Implementation
  - ~ 600 Lines of Code

Configuration Procedure



Creating Neutron Router



Linux Network Namespace

- How to list all network namespaces

  # ip netns list

- How to execute a command in a network namespace

  # ip netns exec $ROUTER_NS $COMMAND

Shell Scripts

- Create an OVS bridge
- Create a network, subnet, and router

} qrouter-create.sh

- Create a tunnel

} tunnel-create.sh

- Set up an interface IP address

} route-init.sh

- Add a route

} route-add.sh

- Add network emulation

} netem-add.sh

qrouter-create.sh

INPUT

- **BRIDGE_NAME**
- **NET_NAME**
- **SUBNET_NAME, SITE_CIDR, SITE_GATEWAY**
- **ROUTER_NAME**

```
#!/bin/bash
OVSVSCTL=/usr/bin/ovs-vsctl
NEUTRON=/usr/bin/neutron

$OVSVSCTL add-br $BRIDGE_NAME

$NEUTRON net-create $NET_NAME –shared

$NEUTRON subnet-create $NET_NAME $SITE_CIDR --name $SUBNET_NAME --gateway $SITE_GATEWAY

$NEUTRON router-create $ROUTER_NAME

$NEUTRON router-interface-add $ROUTER_NAME $SUBNET_NAME
```

tunnel-create.sh

INPUT

- BRIDGE_NAME
- **TUNNEL_NAME**
- **REMOTE_IP**

```
#!/bin/bash
OVSVSCTL=/usr/bin/ovs-vsctl

$OVSVSCTL add-port $BRIDGE_NAME $TUNNEL_NAME -- set interface $TUNNEL_NAME type=gre
options:remote_ip=$REMOTE_IP
```

route-init.sh

```
#!/bin/bash
IP=/sbin/ip
IFCONFIG=/sbin/ifconfig

$IP link set $BRIDGE_NAME netns $ROUTER_NS

$IP netns exec $ROUTER_NS $IFCONFIG $BRIDGE_NAME $WANIP_PREFIXLEN
```

INPUT

- BRIDGE_NAME
- **ROUTER_NS**
- **WANIP_PREFIXLEN**

route-add.sh

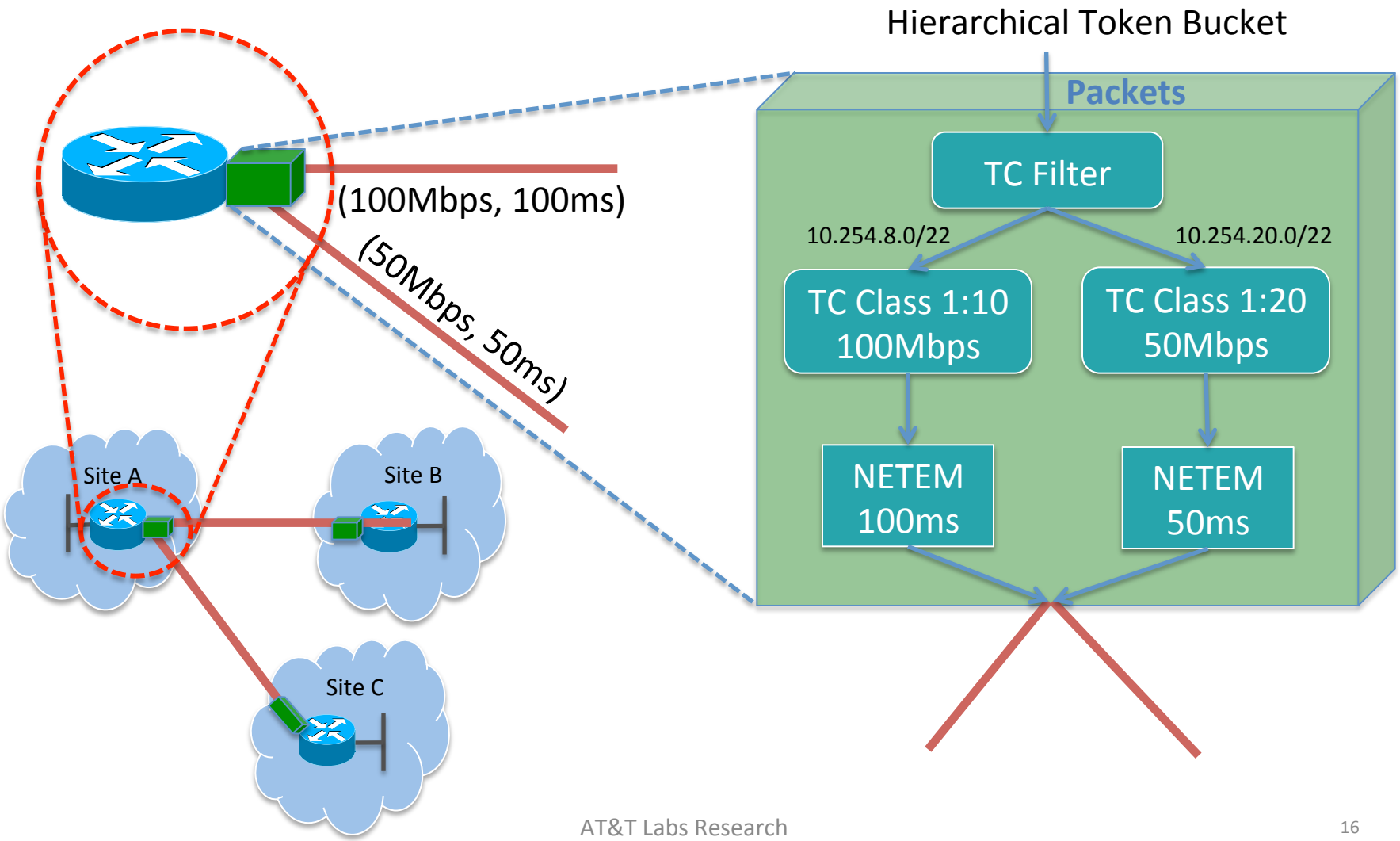
INPUT

- ROUTER_NS
- **REMOTE_CIDR**
- **REMOTE_GATEWAY**

```
#!/bin/bash
IP=/sbin/ip
ROUTE=/sbin/route

$IP netns exec $ROUTER_NS $ROUTE add -net $REMOTE_CIDR gw $REMOTE_GATEWAY
```

Network Emulation



netem-add.sh

INPUT

- ROUTER_NS
- BRIDGE_NAME
- **RATE, CEIL**
- **DELAY**
- **REMOTE_CIDR**

```
#!/bin/bash
IP=/sbin/ip
TC=/sbin/tc
NEUTRON=/usr/bin/neutron

$IP netns exec $ROUTER_NS $TC qdisc replace dev $BRIDGE_NAME handle 1: root htb default 10

$IP netns exec $ROUTER_NS $TC class replace dev $BRIDGE_NAME parent 1: classid
1:$TC_CLASSID htb rate $RATE ceil $CEIL

$IP netns exec $ROUTER_NS $TC qdisc replace dev $BRIDGE_NAME parent 1:$TC_CLASSID handle
$TC_CLASSID: netem delay $DELAY

$IP netns exec $ROUTER_NS $TC filter replace dev $BRIDGE_NAME parent 1: protocol ip prio 1
u32 match ip dst $REMOTE_CIDR flowid 1:$TC_CLASSID
```

To Clean Up

- Scripts for clean-ups
  - qrouter-destroy.sh
  - tunnel-destroy.sh
  - route-del.sh
  - netem-del.sh

NetArbiter

INTEGRATION WITH ANSIBLE

What is Ansible?

- What is Ansible?
  - *Ansible* is an IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, etc.
  - Red Hat announced the acquisition of *Ansible* on October 16, 2015.
- Features
  - Uses no agents and no additional custom security infrastructure.
    - → Easy deployment
  - Uses YAML in the form of Ansible Playbooks
    - → Automation jobs in YAML

NetAribiter Directory Structure

Configuration Files

```
netarbiter
├── hosts
│   ├── all
│   ├── cirrus1_cirrus4
│   └── cirrus2_cirrus4
├── host_vars
│   ├── cirrus101
│   ├── cirrus201
│   └── cirrus401
├── netemu-create.yml
├── netemu-destroy.yml
├── qrouter-create.yml
├── qrouter-destroy.yml
├── route-create.yml
└── route-destroy.yml
```

Playbook.yml

```
├── roles
│   ├── netem
│   │   ├── files
│   │   │   ├── netem-add.sh
│   │   │   └── netem-del.sh
│   │   └── tasks
│   │       └── main.yml
│   ├── qrouter
│   │   ├── files
│   │   │   ├── qrouter-create.sh
│   │   │   └── qrouter-destroy.sh
│   │   └── tasks
│   │       └── main.yml
│   ├── route
│   │   ├── files
│   │   │   ├── route-add.sh
│   │   │   ├── route-del.sh
│   │   │   └── route-init.sh
│   │   └── tasks
│   │       └── main.yml
│   └── tunnel
│       ├── files
│       │   ├── tunnel-create.sh
│       │   └── tunnel-destroy.sh
│       └── tasks
│           └── main.yml
└── README
```

Playbook Roles

Link Configuration (ex. cirrus1_cirrus4)

```
[multisite_link]
cirrus101 REMOTE_IP=135.197.240.47 REMOTE_CIDR=10.254.40.0/22
REMOTE_GATEWAY=172.16.17.104
cirrus401 REMOTE_IP=135.197.240.11 REMOTE_CIDR=10.254.8.0/22
REMOTE_GATEWAY=172.16.17.101

[multisite_link:vars]
DELAY=80ms
RATE=10mbit
CEIL=100mbit
TUNNEL_NAME=gre_cirrus1_cirrus4
```

Host Configuration (ex. cirrus101)

```
---
WANIP_PREFIXLEN: 172.16.17.101/24
SITE_CIDR: 10.254.8.0/22
SITE_GATEWAY: 10.254.8.1
BRIDGE_NAME: br-multisite
NET_NAME: multisite-net
SUBNET_NAME: multisite-subnet
ROUTER_NAME: multisite-router

# Neutron-related
os_env:
  OS_PROJECT_NAME: admin
  OS_TENANT_NAME: admin
  OS_USERNAME: admin
  OS_PASSWORD: ******
  OS_AUTH_URL: http://controller:35357
  OS_VOLUME_API_VERSION: 2
  OS_IMAGE_API_VERSION: 2
```

Playbook: qrouter-create.yml

```
---
- hosts: "{{ HOST }}"
  remote_user: yourusername
  sudo: yes
  gather_facts: no

  roles:
     - {role: qrouter, subcommand: qrouter_create}
```

Playbook: route-create.yml

```
---
- hosts: multisite_link
  remote_user: yourusername
  sudo: yes
  gather_facts: no

  roles:
    - {role: tunnel, subcommand: tunnel_create}
    - {role: route, subcommand: route_init}
    - {role: route, subcommand: route_add}
```

Playbook: netem-create.yml

```
---
- hosts: multisite_link
  remote_user: yourusername
  sudo: yes
  gather_facts: no

  roles:
    - {role: netem, subcommand: netem_add}
```

Playbook Role: roles/netem/main.yml

```
---
- name: Add network emulation - netem-add.sh
  script: netem-add.sh {{ BRIDGE_NAME }} {{ ROUTER_NAME }} {{ REMOTE_CIDR }}
{{ DELAY }} {{ RATE }} {{ CEIL }}
  environment: os_env
  when: subcommand == 'netem_add'
  register: netem_add

- name: Delete network emulation - netem-del.sh
  script: netem-del.sh {{ BRIDGE_NAME }} {{ ROUTER_NAME }} {{ REMOTE_CIDR }}
  environment: os_env
  when: subcommand == 'netem_del'
  register: netem_del

- debug: var=netem_add.stdout_lines
- debug: var=netem_del.stdout_lines
```

How to Run

0. Setup

- Store all hosts' domain names (or IP addresses) to an inventory file "all".
- Create an inventory file for each host.
- Create an inventory file for each link between two hosts.

1. Create/destroy a qrouter (and an ovs bridge) in a host

```
$ ansible-playbook -K -i hosts/all qrouter-create.yml -e HOST=<hostname>
$ ansible-playbook -K -i hosts/all qrouter-destroy.yml -e HOST=<hostname>
```

2. Create/destroy routes (and tunnels) to connect two sites

```
$ ansible-playbook -K -i hosts/<inventory> route-create.yml
$ ansible-playbook -K -i hosts/<inventory> route-destroy.yml
```

3. Create delay, rate, and ceil with TC/NETEM

```
$ ansible-playbook -K -i hosts/<inventory> netemu-create.yml [-e DELAY=<delay> -e RATE=<rate> CEIL=<ceil>]
$ ansible-playbook -K -i hosts/<inventory> netemu-destroy.yml [-e DELAY=<delay> -e RATE=<rate> CEIL=<ceil>]
```

How to Run

1. Create virtual routers

$ ansible-playbook -K -i hosts/all qrouter-create.yml -e HOST=cirrus401

$ ansible-playbook -K -i hosts/all qrouter-create.yml -e HOST=cirrus201

→ Check the Network tab of OpenStack Horizon to see if routers are created.

2. Create routes between virtual routers

$ ansible-playbook -K -i hosts/cirrus2_cirrus4 route-create.yml

3. Launch two instances from cirrus2 and cirrus4 via OpenStack Horizon

→ Check connectivity by running ping from instances.

4. Add delay

$ ansible-playbook -K -i hosts/cirrus2_cirrus4 netemu-create.yml

$ ansible-playbook -K -i hosts/cirrus2_cirrus4 netemu-create.yml -e DELAY=80ms

→ Check delays by running 'ping 10.254.20.3' from a cirrus4's instance (i.e. 10.254.40.3).

How to Clean Up

Clean up all configurations

$ ansible-playbook -K -i hosts/all qrouter-destroy.yml -e HOST=cirrus201
$ ansible-playbook -K -i hosts/all qrouter-destroy.yml -e HOST=cirrus401

Note

- In order to destroy Neutron routers, first terminal all instances connected to the routers, or detach all connections to the routers.
- *qrouter-destroy.sh* deletes router, subnet, net from Neutron and bridge from OVS.

qrouter-destroy.sh

```
#!/bin/bash
OVSVSCTL=/usr/bin/ovs-vsctl
NEUTRON=/usr/bin/neutron

$NEUTRON router-interface-delete $ROUTER_NAME $SUBNET_NAME
$NEUTRON router-delete $ROUTER_NAME
$NEUTRON subnet-delete $SUBNET_NAME
$NEUTRON net-delete $NET_NAME
$OVSVSCTL del-br $BRIDGE_NAME
```

How to Debug

0. Connect via ssh to the Neutron network node of a cloud site

1. Check the creation of an OVS bridge
ovs-vsctl show

2. Find *router_id* **of a newly-created virtual router**
neutron router-list

3. Find the network namespace of a virtual router (i.e., *qrouter-router_id***)**
ip netns list

4. Check the routing table of a virtual router
ip netns exec *qrouter-router_id* netstat –rn

4. Check TC/NETEM configuration
ip netns exec *qrouter-router_id* tc qdisc show
ip netns exec *qrouter-router_id* tc class show dev *bridge_name*
ip netns exec *qrouter-router_id* tc filter show dev *bridge_name*