SDN Based Network Project Documentation

Hadi Rezay

Abstract

The following documentation report covers the steps, commands, screenshots and explanation of the proposed SDN network project.

I. Equipment

The equipment/software used were home desktop, GNS3, Ubuntu Linux, Wireshark, Open-source controllers.

II. LAN Deployment

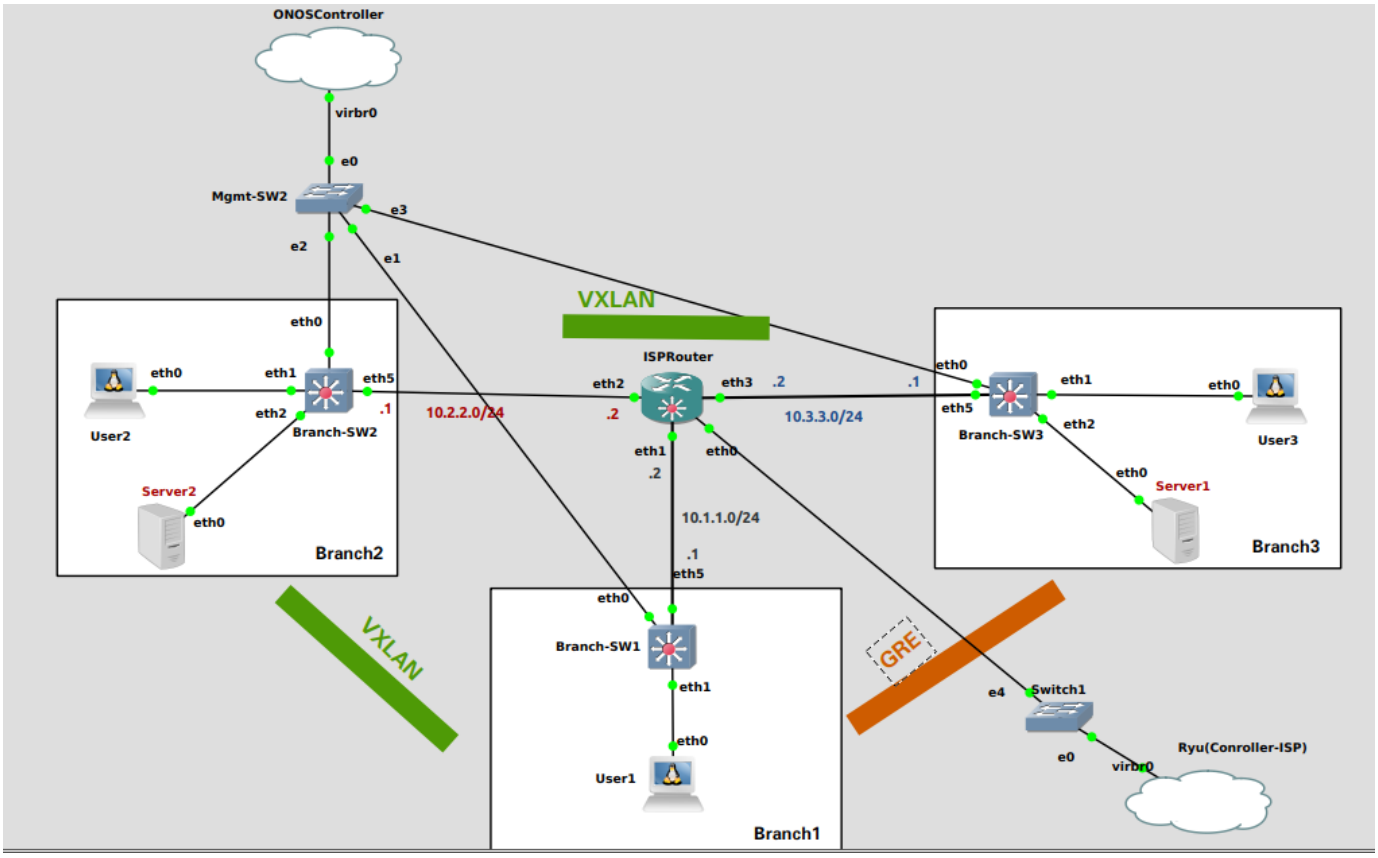

Fig. 1. LAN Deployment

A. PCs and Servers Configuration

For all user PCs and servers permanent IP configuration `/etc/network/interfaces` file is modified, User2 as shown below in fig2 as an example.

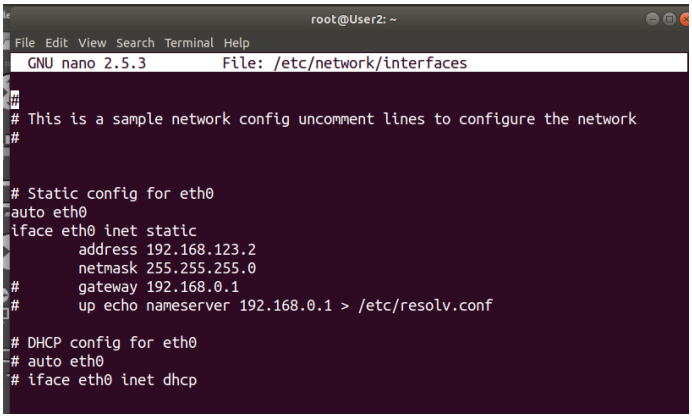

Fig. 2. User2 IP Config

B. Branch OpenvSwitches Deployment

At each branch, the branch switch's default bridge is deleted and a new bridge called `my-br` is created, eth1 with tag of 10 and eth2 with tag of 20 are added to `my-br` bridge. Eth0 and eth5 are removed from the bridge and configured static permanent IP addresses, eth5 is also configured with default gateway address pointing towards ISPRouter. Each branch switch's OpenFlow version changed from 1.1 to 1.4 and STP is enabled. All OVS switches are connected to Cloud object.

C. ONOS Deployment Instructions

Since ONOS controller needs to run outside GNS3, to start the controller follow the instructions:

1) Start ONOS: `sudo /opt/onos/bin/onos-service start`

2) Open a new terminal, log in to ONOS, username is ***onos*** *and passwrd is* ***rocks****:* `ssh -p 8101 -o StrictHostKeyChecking=no onos@localhost`

3) Change the ONOS port to 7777: `cfg set org.onosproject.openflow.controller.impl.OpenFlowControllerI openflowPorts 7777`

Once ONOS is started and the required packages are installed, the ONOS GUI should show all devices and hosts as shown in below screenshot:

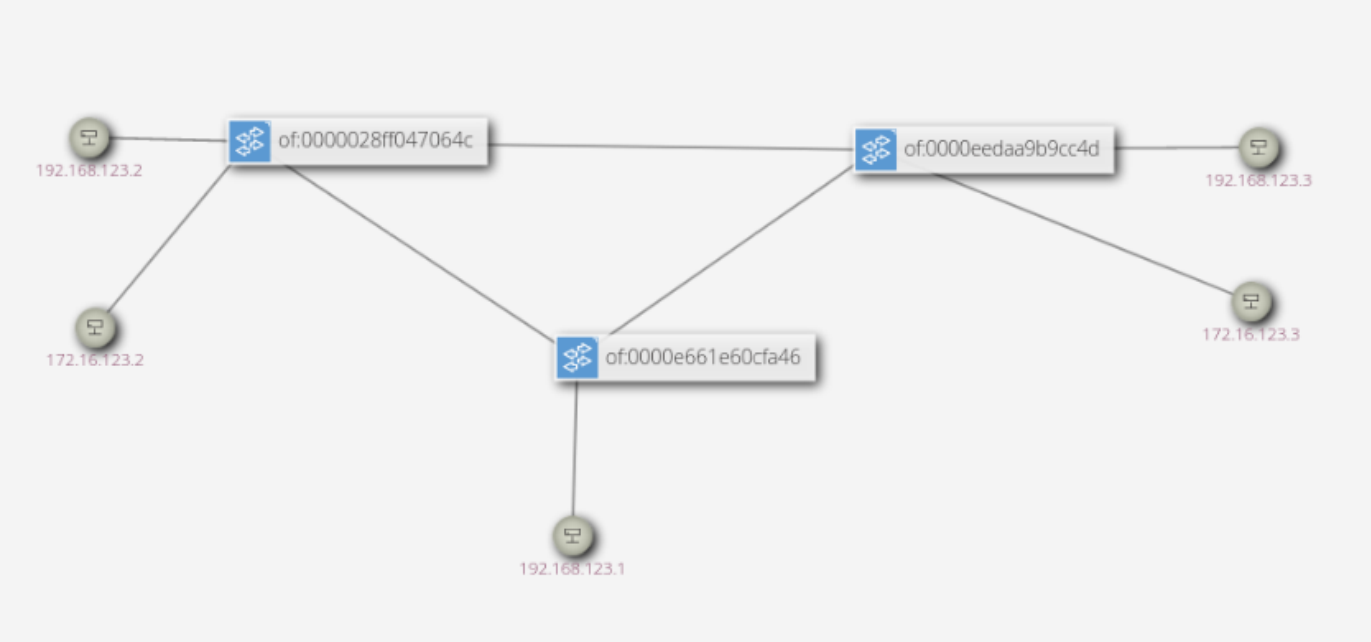

Fig. 3. ONOS GUI Topology

Below is an example of switch flow view on ONOS GUI of one of the Branch switch.

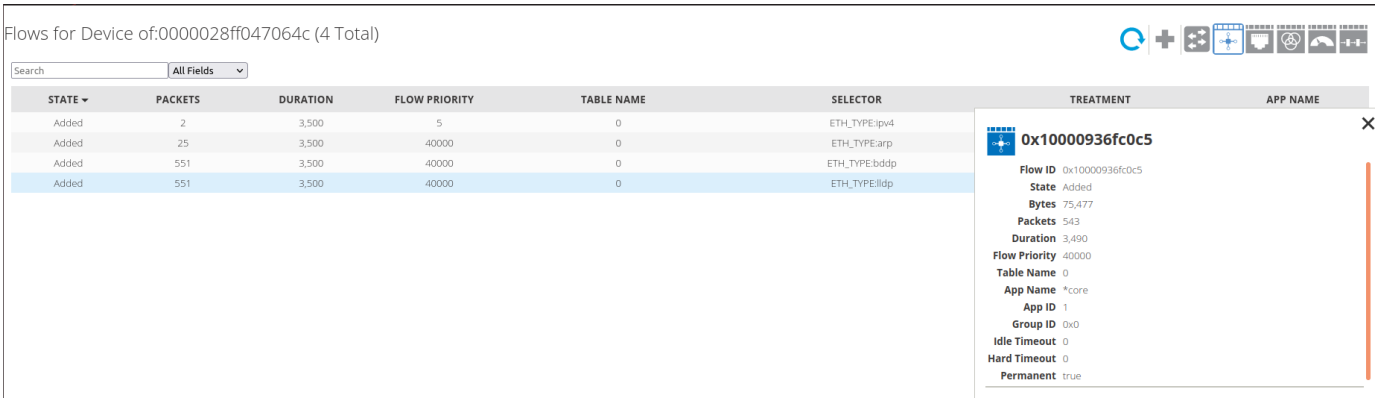

Fig. 4. ONOS Switch FLow

III. WAN Deployment

A. WAN ISP Router

Default bridge is deleted, a new bridge called `my-br` is created and eth1, eth2, eth3 are added. Management interface eth0 is not included in the new bridge and is configured statis IP address. ISPRouter is also connected to Ryu controller. The three branch facing interfaces are later configured via REST API Curl commands.

B. Ryu Controller Deployment and Instructions

Ryu Controller has been tested and is able to detect ISPRouter and its flows as shown below:

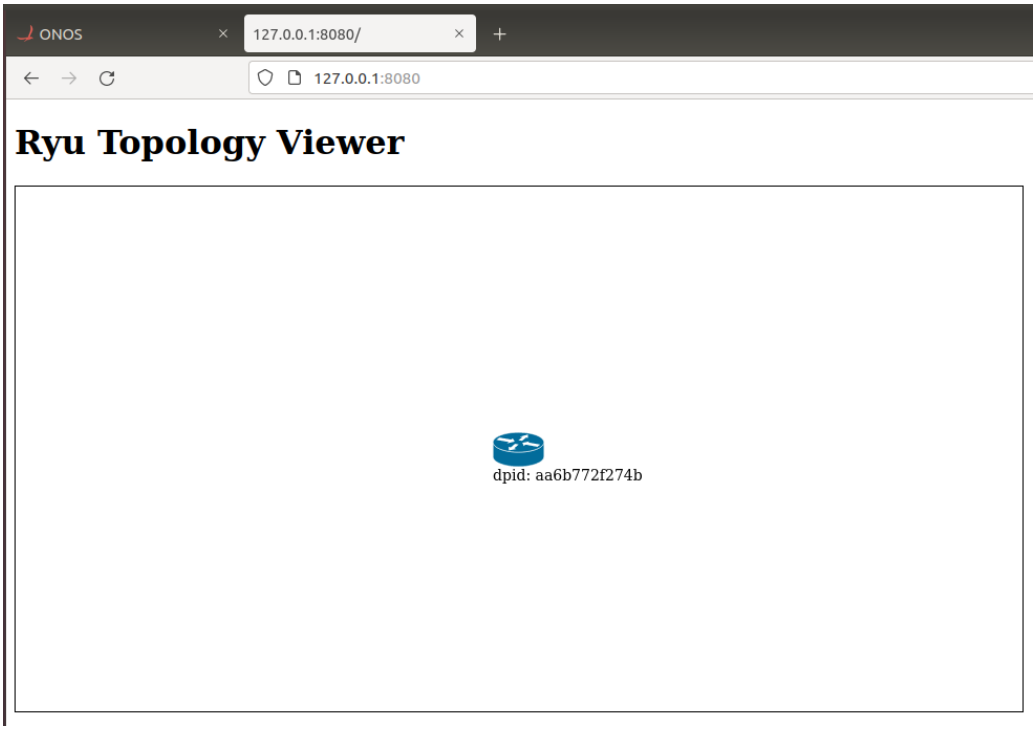

Fig. 5. Ryu GUI Topology and flows

- { "priority": 1037, "cookie": 1, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 294, "duration_sec": 1901, "duration_nsec": 886000000, "packet_count": 3, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.1.1.2" }, "table_id": 0 }
- { "priority": 1037, "cookie": 2, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1587, "duration_nsec": 405000000, "packet_count": 0, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.2.2.2" }, "table_id": 0 }
- { "priority": 1037, "cookie": 3, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1560, "duration_nsec": 575000000, "packet_count": 0, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.3.3.2" }, "table_id": 0 }
- { "priority": 35, "cookie": 1, "idle_timeout": 1800, "hard_timeout": 0, "byte_count": 370282, "duration_sec": 42, "duration_nsec": 404000000, "packet_count": 2042, "length": 136, "flags": 0, "actions": ["DEC_NW_TTL", "SET_FIELD: {eth_src:56:f7:8d:cd:a7:3a}", "SET_FIELD: {eth_dst:12:f6:7b:cf:d7:22}", "OUTPUT:2"], "match": { "dl_type": 2048, "nw_dst": "10.1.1.1" }, "table_id": 0 }
- { "priority": 35, "cookie": 3, "idle_timeout": 1800, "hard_timeout": 0, "byte_count": 367043, "duration_sec": 11, "duration_nsec": 432000000, "packet_count": 2025, "length": 136, "flags": 0, "actions": ["DEC_NW_TTL", "SET_FIELD: {eth_src:d6:47:ac:de:3d:68}", "SET_FIELD: {eth_dst:02:fa:55:99:cc:30}", "OUTPUT:1"], "match": { "dl_type": 2048, "nw_dst": "10.3.3.1" }, "table_id": 0 }
- { "priority": 35, "cookie": 2, "idle_timeout": 1800, "hard_timeout": 0, "byte_count": 385095, "duration_sec": 2, "duration_nsec": 214000000, "packet_count": 2039, "length": 136, "flags": 0, "actions": ["DEC_NW_TTL", "SET_FIELD: {eth_src:7a:cf:f0:26:92:0c}", "SET_FIELD: {eth_dst:6a:55:13:8e:72:e6}", "OUTPUT:3"], "match": { "dl_type": 2048, "nw_dst": "10.2.2.1" }, "table_id": 0 }
- { "priority": 36, "cookie": 1, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1901, "duration_nsec": 887000000, "packet_count": 0, "length": 112, "flags": 0, "actions": ["OUTPUT:NORMAL"], "match": { "dl_type": 2048, "nw_src": "10.1.1.0/255.255.255.0", "nw_dst": "10.1.1.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 36, "cookie": 2, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1587, "duration_nsec": 406000000, "packet_count": 0, "length": 112, "flags": 0, "actions": ["OUTPUT:NORMAL"], "match": { "dl_type": 2048, "nw_src": "10.2.2.0/255.255.255.0", "nw_dst": "10.2.2.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 36, "cookie": 3, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1560, "duration_nsec": 575000000, "packet_count": 0, "length": 112, "flags": 0, "actions": ["OUTPUT:NORMAL"], "match": { "dl_type": 2048, "nw_src": "10.3.3.0/255.255.255.0", "nw_dst": "10.3.3.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 2, "cookie": 1, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 0, "duration_sec": 1901, "duration_nsec": 888000000, "packet_count": 0, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.1.1.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 2, "cookie": 2, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 189, "duration_sec": 1587, "duration_nsec": 407000000, "packet_count": 1, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.2.2.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 2, "cookie": 3, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 363, "duration_sec": 1560, "duration_nsec": 576000000, "packet_count": 2, "length": 96, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2048, "nw_dst": "10.3.3.0/255.255.255.0" }, "table_id": 0 }
- { "priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 74298, "duration_sec": 2247, "duration_nsec": 242000000, "packet_count": 1769, "length": 88, "flags": 0, "actions": ["OUTPUT:CONTROLLER"], "match": { "dl_type": 2054 }, "table_id": 0 }
- { "priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 80964, "duration_sec": 2247, "duration_nsec": 242000000, "packet_count": 446, "length": 64, "flags": 0, "actions": [], "match": { "dl_type": 2048 }, "table_id": 0 }
- { "priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 280, "duration_sec": 2247, "duration_nsec": 242000000, "packet_count": 4, "length": 80, "flags": 0, "actions": ["OUTPUT:NORMAL"], "match": {}, "table_id": 0 }

Fig. 6. Ryu GUI ISPRouter's flows

ISPRouter is connected to Cloud object's `virbr0` interface. Since Ryu controller needs to run outside GNS3, to start the controller follow the instructions:

1) : Open a new Ubuntu Terminal

2) : Change directory `cd /home/sdn/Downloads/ryu/ryu/app`

3) : Start Ryu controller `ryu-manager --verbose rest_router.py gui_topology/gui_topology.py`

C. Configure ISPRouter's interfaces

To configure IP interfaces for ISPRouter's eth1, eth2, eth3. The following Curl commands have executed on another Ubuntu terminal. when opening the GNS3 project file on another machine, if for some reason ISPRouter's ID changes or the interfaces configured disappear, below Curl commands may have to be re-executed to enable connectivity between ISPRouter and the switches.

```
curl -X POST -d '"address":"10.1.1.2/24"' http://localhost:8080/router/0000aa6b772f274b
curl -X POST -d '"address":"10.2.2.2/24"' http://localhost:8080/router/0000aa6b772f274b
curl -X POST -d '"address":"10.3.3.2/24"' http://localhost:8080/router/0000aa6b772f274b
```

IV. Site to Site Connectivity

A. Branch1 to Branch2 - VxLAN

VxLAN has been configured between Branch 1 and Branch 2 to allow vlan 10 traffic of User1 and User2. Wireshark capture, and ping screenshots validates ping connectivity from User1 to User2 and Vxlan encaputlation of traffic captured on ISPRouter-BranchSW2 link:

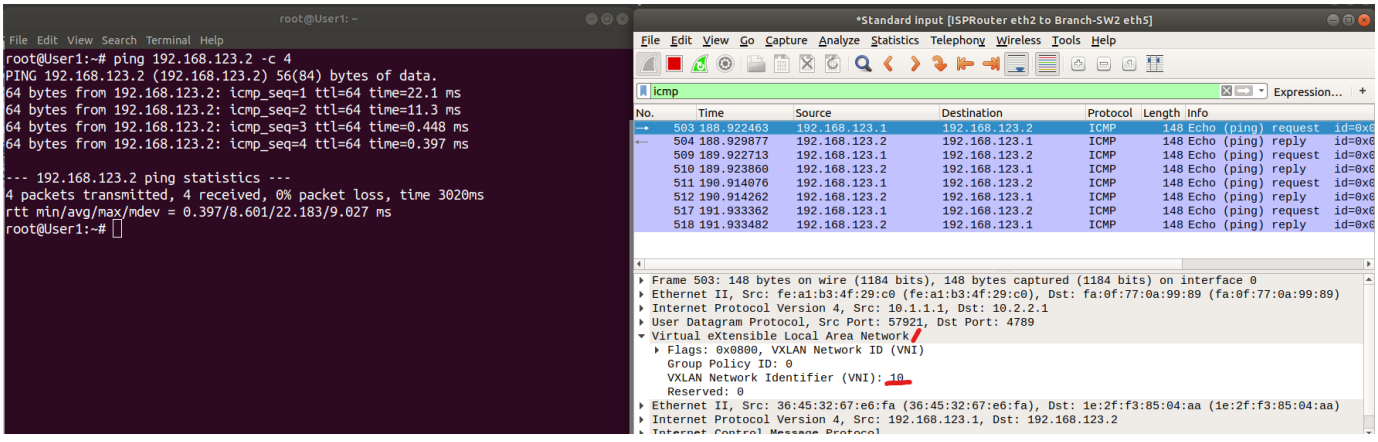

Fig. 7. Ping - Wireshark

1) On SW1, command used to configure VxLAN: `ovs-vsctl add-port my-br vxlan10 tag=10 -- set interface vxlan10 type=vxlan options:key=10 options:remote_ip=10.2.2.1`

2) On SW2, command used to configure VxLAN: `ovs-vsctl add-port my-br vxlan10 tag=10 -- set interface vxlan10 type=vxlan options:key=10 options:remote_ip=10.1.1.1`

B. Branch1 to Branch3 - GRE Tunnel

GRE tunnel has been configured between Branch 1 and Branch 3 to allow vlan 10 traffic of of User1 and User3. Lecture notes have been used to for GRE commands. Wireshark capture, and ping screenshots validates ping connectivity from User1 to User3 and GRE encaputlation of traffic captured on ISPRouter-BranchSW3 link:

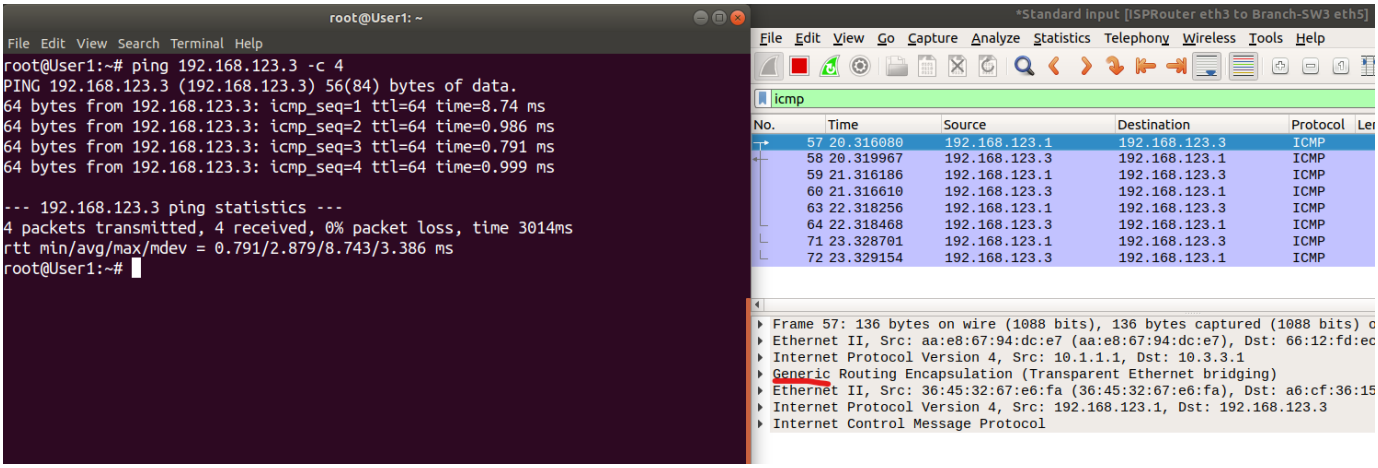

Fig. 8. Ping - Wireshark

1) On SW1, command used to configure GRE: `ovs-vsctl add-port my-br gre0 tag=10 -- set interface gre0 type=gre options:remote_ip=10.3.3.1`

2) On SW3, command used to configure GRE: `ovs-vsctl add-port my-br gre0 tag=10 -- set interface gre0 type=gre options:remote_ip=10.1.1.1` dwadawdwadwad

C. Branch2 to Branch3 - VxLAN Tunnel

VxLAN tunnel has been configured between Branch 1 and Branch 3 to allow vlan 20 traffic of the servers.

1) On SW2, command used to configure GRE: `ovs-vsctl add-port my-br vxlan20 tag=20 -- set interface vxlan20 type=vxlan options:key=20 options:remote_ip=10.3.3.1`

2) On SW3, command used to configure GRE: `ovs-vsctl add-port my-br vxlan20 tag=20 -- set interface vxlan20 type=vxlan options:key=20 options:remote_ip=10.2.2.1`