Using Open Virtual Network Lab
Layer-3 Switch Replacement

Table of Contents

1. Motivation

The current switch is a Cisco Catalyst 4006 w/Supervisor V. The chassis is 17 years old and the supervisor almost eight years old. The age plus the amount of power required to run I decided it was time to go. I have a significantly newer switch that is unfortunately only layer-2. Of course a sane person would say that is good enough.

2. Configuration

Just a fair warning I would never claim to be an expert on networking, Open vSwitch or OVN. To get this to work was mostly working from other individual's blog posts, documentation, etc. If something is incorrect please either submit an issue or pull request. With that said let's get started.

2.1. Installation and initial configuration

All the nodes directly participating geneve overlay will be Fedora 27. All the following commands will be based on Fedora. The configuration of OVN should generally be distribution independent.

2.1.1. OVN northbound database node

First let's install some packages.

```
dnf install openvswitch \
            openvswitch-ovn-central \
            openvswitch-ovn-common \
            openvswitch-ovn-host \
            python2-openvswitch \
            python3-openvswitch -y
```

If you want to use the container version of Skydive install docker as well. (recommended)

```
dnf install docker \
            docker-compose -y
```

Then start and enable openvswitch and OVN services.

```
systemctl start openvswitch
systemctl enable openvswitch
systemctl start ovn-northd
systemctl enable ovn-northd
```

2.1.2. Hypervisor nodes

Hypervisor in this case is any node that is to be native on OVN. In my case that is any Fedora physical node including my laptop - at least when its docked.

```
sudo dnf install openvswitch \
                 openvswitch-ovn-common \
                 openvswitch-ovn-host
```

If you want Skydive install docker.

```
sudo dnf install docker \
                 docker-compose -y
```

Enable and start services, wait to start `ovn-controller` until the next section.

```
systemctl start openvswitch
systemctl enable openvswitch
systemctl enable ovn-controller
```

2.1.3. Allow north and south bound connections

On the node running `ovn-northd` execute the following commands to allow remote connections.

```
ovn-nbctl set-connection ptcp:6641
ovn-sbctl set-connection ptcp:6642
ovs-appctl -t ovsdb-server ovsdb-server/add-remote ptcp:6640
```

2.1.4. Configure ovn-controller on hypervisor nodes

The docker how to provides the commands required to configure the controller. I also listed them below.

```
CENTRAL_IP=172.30.1.10      ①
ENCAP_TYPE=geneve
LOCAL_IP=172.30.1.52        ②

ovs-vsctl set open . external_ids:ovn-remote="tcp:${CENTRAL_IP}:6642"
ovs-vsctl set open . external_ids:ovn-nb="tcp:${CENTRAL_IP}:6641"
ovs-vsctl set open . external_ids:ovn-encap-ip=${LOCAL_IP}
ovs-vsctl set open . external_ids:ovn-encap-type="${ENCAP_TYPE}"
```

① IP address of the node running `ovn-northd`

② IP address of current node

After configuration start the `ovn-controller`.

```
systemctl start ovn-controller
```

If you run `ovn-vsctl show` you should see at least a new bridge `br-int`.

```
$ sudo ovs-vsctl show
90a55931-3706-4d55-9913-4a6e1f4b09e5
    Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
```

2.1.5. Gateway node

While I am in the process of migrating to OVN I need access to existing devices on the *legacy* network. It will also be required for devices that do not support OVN. Not sure if this is technically the best way to do this but it works. Additional OVN configuration is required which will be listed below.

I made the assumption that IP forwarding would be required so I enabled it.

```
net.ipv4.ip_forward = 1
```

Configure the `br-dmz` Open vSwitch bridge. This will be in file `/etc/sysconfig/network-scripts/ifcfg-br-dmz`

```
DEVICE=br-dmz
ONBOOT=yes
BOOTPROTO=none
TYPE=OVSBridge
DEVICETYPE=ovs
OVS_EXTRA="set Open_vSwitch . external-ids:ovn-bridge-mappings=dmz_localnet:br-dmz" ①
```

① Map the `dmz_localnet` switch port to the `br-dmz` bridge.

Configure the physical interface attached to the `legacy` network and add to the `br-dmz` bridge. The IP address will be defined in OVN. This configuration will be in file `/etc/sysconfig/network-scripts/ifcfg-enp2s5`.

```
NAME="enp2s5"
DEVICE="enp2s5"
ONBOOT="yes"
NETBOOT="yes"
IPV6INIT="no"
BOOTPROTO="none"
DEFROUTE="no"
IPV4_FAILURE_FATAL="no"
IPV6_AUTOCONF="no"
IPV6_DEFROUTE="no"
IPV6_FAILURE_FATAL="no"

TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-dmz
```

Finally bring up the bridge and interface.

```
ifup br-dmz
ifup enp2s5
```

2.1.6. Hypervisor node network config

```
DEVICE=bromine
NAME=bromine
TYPE=OVSIntPort
DEVICETYPE=ovs
OVS_BRIDGE=br-int
OVS_EXTRA="set Interface bromine external_ids:iface-id=sw51_bromine"    ①
IPADDR=172.31.51.52
NETMASK=255.255.255.0
GATEWAY=172.31.51.1
DEFROUTE=yes
MTU=1500
DNS1=10.53.252.123
DNS2=10.53.252.246
```

① The `iface-id` will be name provided in the `lsp-add` command.

Bring up the interface but it won’t pass traffic until the logical switch port is created.

```
ifup bromine
```

2.2. Create OVN switches, routers and static routes

The topology and most of the OVN configuration below were modifications from this guide so I recommend reading it for additional information.

2.2.1. Adding Logical Switches

Normal switches: `sw50`, `sw51`, and `sw52`.

Transit switch is between router `r0` and the gateway router `gr0`. Assuming this is to allow `r0` to be distributed while maintaining a connection to the localized `gr0`.

```
desk=sw51
transit=tsw0
outsw=osw0
prod=sw52
ose=sw50

ovn-nbctl --may-exist ls-add ${desk}
ovn-nbctl --may-exist ls-add ${transit}
ovn-nbctl --may-exist ls-add ${outsw}
ovn-nbctl --may-exist ls-add ${prod}
ovn-nbctl --may-exist ls-add ${ose}
```

2.2.2. Adding Logical Routers

Only need two routers `r0` and `gr0`.

```
router=r0
gr=gr0

ovn-nbctl --may-exist lr-add ${router}
chassis_uuid=$(ovn-sbctl --bare --columns name find Chassis hostname=ovn-
gateway0.virtomation.com)
ovn-nbctl create Logical_Router name=${gr} options:chassis=${chassis_uuid}      ①
```

① The gateway router must be configured on a specific node or chassis.

2.2.3. Adding Logical Router Ports

Create logical router ports with mac and ip addresses for each network.

```
ovn-nbctl --may-exist lrp-add ${router} ${router}_${desk} 02:ac:10:1f:33:01
172.31.51.1/24
ovn-nbctl --may-exist lrp-add ${router} ${router}_${prod} 02:ac:10:1f:34:01
172.31.52.1/24
ovn-nbctl --may-exist lrp-add ${router} ${router}_${ose} 02:ac:10:1f:32:01
172.31.50.1/24
ovn-nbctl --may-exist lrp-add ${router} ${router}_${transit} 02:ac:10:1f:ff:02
172.31.255.2/30
ovn-nbctl --may-exist lrp-add ${gr} ${gr}_${transit} 02:ac:10:1f:ff:01 172.31.255.1/30
ovn-nbctl --may-exist lrp-add ${gr} ${gr}_${outsw} 02:ac:10:1f:0c:f6 10.53.12.246/24
```

2.2.4. Adding Static Routes

Create static routes to enable traffic between networks.

```
ovn-nbctl lr-route-add ${gr} 0.0.0.0/0 10.53.12.1           ①
ovn-nbctl lr-route-add ${gr} 10.53.0.0/16 10.53.12.254      ②
ovn-nbctl lr-route-add ${gr} 172.31.0.0/16 172.31.255.2     ③
ovn-nbctl lr-route-add ${router} 0.0.0.0/0 172.31.255.1     ④
```

① Static route for internet traffic.

② Static route for `legacy` networks.

③ Static route for overlay networks.

④ Static route for all external networks.

2.2.5. Adding Logical Switch Ports

Create logical switch ports for each router, physical device and the gateway.

```
# Router
ovn-nbctl --may-exist lsp-add ${desk} ${desk}_${router}
ovn-nbctl --may-exist lsp-add ${prod} ${prod}_${router}
ovn-nbctl --may-exist lsp-add ${ose} ${ose}_${router}
ovn-nbctl --may-exist lsp-add ${transit} ${transit}_${router}
ovn-nbctl --may-exist lsp-add ${outsw} ${outsw}_${gr}
ovn-nbctl --may-exist lsp-add ${transit} ${transit}_${gr}

# Physical
ovn-nbctl --may-exist lsp-add ${desk} ${desk}_bromine
ovn-nbctl --may-exist lsp-add ${prod} ${prod}_uranium

# Gateway
ovn-nbctl --may-exist lsp-add ${outsw} ${outsw}_localnet
```

2.2.6. Setting Logical Switch Port Configuration

For each port configure the type, allowed address, and appropriate options.

```
# Router
ovn-nbctl lsp-set-type ${desk}_${router} router
ovn-nbctl lsp-set-addresses ${desk}_${router} 02:ac:10:1f:33:01
ovn-nbctl lsp-set-options ${desk}_${router} router-port=${router}_${desk}

ovn-nbctl lsp-set-type ${prod}_${router} router
ovn-nbctl lsp-set-addresses ${prod}_${router} 02:ac:10:1f:34:01
ovn-nbctl lsp-set-options ${prod}_${router} router-port=${router}_${prod}

ovn-nbctl lsp-set-type ${ose}_${router} router
ovn-nbctl lsp-set-addresses ${ose}_${router} 02:ac:10:1f:32:01
ovn-nbctl lsp-set-options ${ose}_${router} router-port=${router}_${ose}

ovn-nbctl lsp-set-type ${outsw}_${gr} router
ovn-nbctl lsp-set-addresses ${outsw}_${gr} 02:ac:10:1f:0c:f6
ovn-nbctl lsp-set-options ${outsw}_${gr} router-port=${gr}_${outsw}

ovn-nbctl lsp-set-type ${transit}_${gr} router
ovn-nbctl lsp-set-addresses ${transit}_${gr} 02:ac:10:1f:ff:01
ovn-nbctl lsp-set-options ${transit}_${gr} router-port=${gr}_${transit}

ovn-nbctl lsp-set-type ${transit}_${router} router
ovn-nbctl lsp-set-addresses ${transit}_${router} 02:ac:10:1f:ff:02
ovn-nbctl lsp-set-options ${transit}_${router} router-port=${router}_${transit}

# Gateway
ovn-nbctl lsp-set-type ${outsw}_localnet localnet
ovn-nbctl lsp-set-addresses ${outsw}_localnet unknown
ovn-nbctl lsp-set-options ${outsw}_localnet network_name=dmz_localnet

# Physical
ovn-nbctl lsp-set-addresses ${desk}_bromine unknown
ovn-nbctl lsp-set-addresses ${prod}_uranium unknown
```

3. Virtualization

After getting a few physical machines up and running on OVN the next step was my real hypervisor nodes. This was more of a challenge than I originally thought it was going to be. I started by reviewing this blog post which certainly provided valuable insight. Though being a lazy programmer there had to be a better way - libvirt hooks.

3.1. Installing the libvirt qemu hook

Provided in this repository is a `qemu` hook for OVN. It adds and removes the switch port when the machine is started or stopped. The configuration for the `ovn-northd` node and the switch name is stored in the virtual machines metadata.

First some prerequisites.

```
dnf install git -y
pip install ovsdbapp
git clone https://github.com/jcpowermac/homelab-ovn
```

If the directory doesn't exist (which it didn't on my hypervisor) create it.

```
mkdir -p /etc/libvirt/hooks/
cp homelab-ovn/libvirt-hook/qemu /etc/libvirt/hooks/
chmod 744 /etc/libvirt/hooks/qemu
```

After the hook is available libvirtd needs to be restarted.

```
systemctl restart libvirtd
```

3.2. Add OVN metadata to virtual machine

The `virt-install` command is an example the `--network` option that must be used to connect a virtual machine to a specific logical switch. The `virsh metadata` command below adds metadata to a defined virtual machine. This command **must** be written exactly as below for the `qemu` hook to function properly.

```
virt-install --import --name $vm --memory 8192 --vcpus 2 \
             --graphics none --console pty,target_type=serial \
             --os-type linux --os-variant rhel7.0 --noautoconsole \
             --disk path=/instances/$vm.qcow2,format=qcow2,bus=virtio \
             --network bridge=br-int,virtualport_type=openvswitch          ①
virsh metadata $vm --uri ovs \
                   --key ovn \
               --set '<parameters northd="172.30.1.10" switch="sw50"/>'    ②
```

① The virtual machine must attach to the `br-int` bridge and have a `virtualport_type` of `openvswitch`.

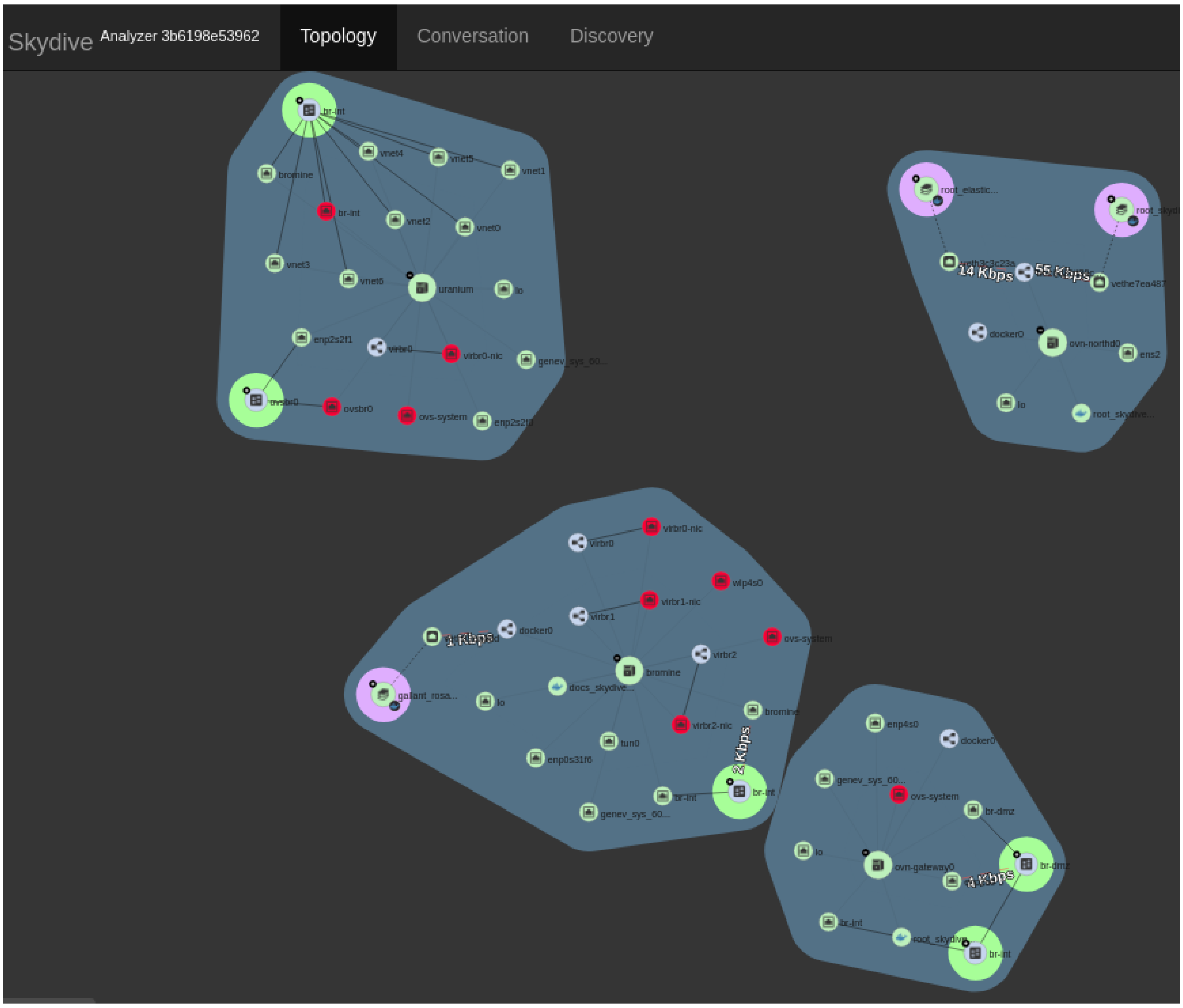
② The parameters are farily simple, `northd` is the ip address of `ovn-northd` node and `switch` is where the virtual machine should be connected.

4. Visualizations

Sometimes its better to have a picture or two.

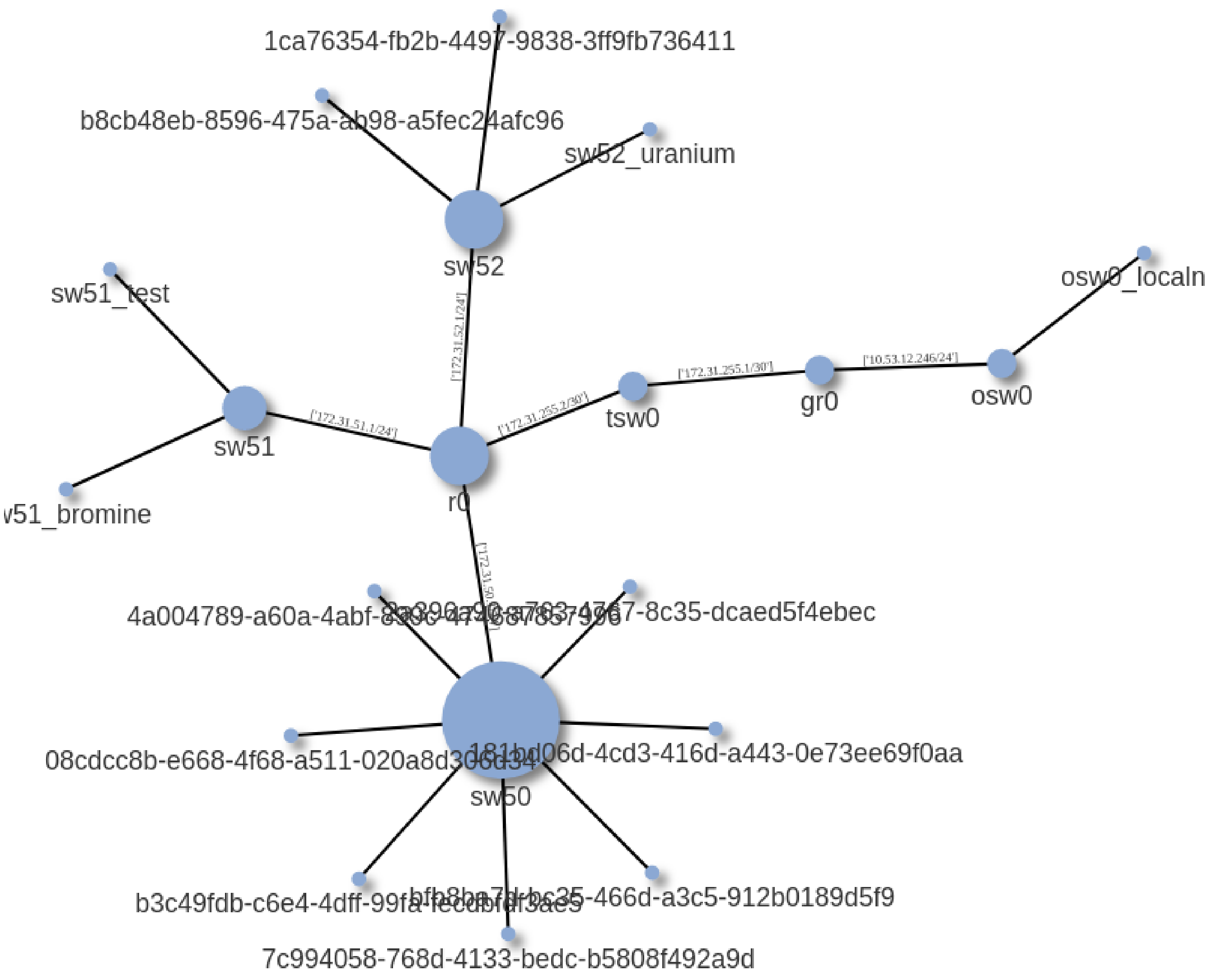
4.1. Skydive

As I suggested above running Skydive is a good idea. At least then you have a visual representation of the interfaces and bridges that are used.



4.2. OVN topology using Jupyter

Included in this repo is a Jupyter notebook. It currently displays logical routers, switches, ports, and networks on the edge.


1ca76354-fb2b-4497-9838-3ff9fb736411
b8cb48eb-8596-475a-ab98-a5fec24afc96
sw52_uranium
sw52
['172.31.52.1/24']
sw51_test
sw51
w51_bromine
['172.31.51.1/24']
r0
['172.31.255.2/30']
tsw0
['172.31.255.1/30']
gr0
['10.53.12.246/24']
osw0
osw0_localn
4a004789-a60a-4abf-
7-8c35-dcaed5f4ebec
08cdcc8b-e668-4f68-a511-020a8d3
6d-4cd3-416d-a443-0e73ee69f0aa
sw50
b3c49fdb-c6e4-4dff-99fa-
5-466d-a3c5-912b0189d5f9
7c994058-768d-4133-bedc-b5808f492a9d

Appendix A: References

A.1. Jupyter

- https://ucsd-ccbb.github.io/visJS2jupyter
- https://networkx.github.io/documentation/stable/index.html
- Missing links for ipywidgets, matplotlib

A.2. OVN

- https://scottlowe.org/2016/12/09/using-ovn-with-kvm-libvirt/
- https://www.pydoc.io/pypi/ovsdbapp-0.9.0/index.html
- https://github.com/oVirt/ovirt-provider-ovn
- https://github.com/openvswitch/ovs/blob/master/tests/ovn.at
- http://blog.spinhirne.com/2016/09/an-introduction-to-ovn-routing.html

A.3. Libvirt Hooks

- https://libvirt.org/formatdomain.html#elementsMetadata
- https://www.libvirt.org/hooks.html
- https://github.com/rhardouin/libvirt_hooks