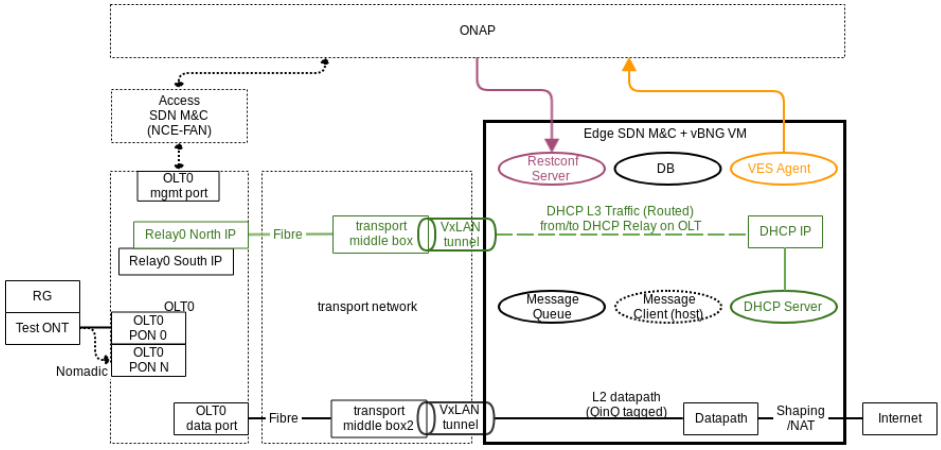
Swisscom Edge SDN M&C and virtual BNG

Introduction

The Swisscom virtual BNG and Edge SDN M&C for the ONAP BBS use-case is a demo system. It is a functional prototype and it is not meant for production use. The whole system is running in a single OpenStack VM and therefore only able to support a small number of subscribers. The forwarding dataplane is implemented fully inside the VMs networking stack and therefore not designed to support high data rates. The system is able to interact with ONAP according to the BBS use-case definition.

Below a diagram of the whole end-to-end architecture, the **Edge SDN M&C + vBNG VM** is highlighted:



As shown above in the diagram the DHCP and Dataplane traffic are brought to the VM by VxLAN tunneling. Since it is just routed L3 traffic, DHCP traffic does not really require VxLAN. For sake of not having two different tunnel technologies we use VxLAN tunneling for both DHCP and Dataplane traffic. Now the question is if the OLT supports native VxLAN tunneling. In our case the OLT does not. Therefore some VxLAN tunnel encapsulation device is required (see transport middle boxes above). Those boxes are very simple to set up. See at the very end of this document how to build such a middle box.

Installation by Heat Template

The Swisscom virtual BNG and Edge SDN M&C is installed and initially set up by a single Heat Orchestration Template. Therefore the OpenStack cloud you would like to use for testing should support Orchestration with Heat. The complete initial vBNG + Edge SDN M&C configuration is provided as Heat stack parameters. The Heat stack creates all the required OpenStack infrastructure, e.g. router, network, security group, port and vbng instance.

Once the stack is created, the vBNG configuration file is deployed to the instance to '$HOME/vbng.conf' and latest vBNG code is directly pulled from specified upstream git repository (default is the Swisscom repository) by cloud-init user-data script. The stack output shows initial vBNG configuration, the floating IP and how to connect to the instance by SSH. To create a stack in OpenStack heat you first require the Heat template from here: vbng.yaml

Option A) Upload template in Horizon

The stack can be created directly in OpenStack Horizon by:

- Navigating to 'Orchestration -> Stacks' in the sidebar
- Pressing the 'Launch Stack' button
- In 'Template Source' select to upload the template file 'vbng.yaml' and press 'Next'.
- Now heat asks for the stack input parameters, they can be set by just entering the desired values in the Horizon form.

The template defines a hopefully useful default for each parameter, therefore not much has to be changed for the Swisscom Lab installation. However, some things like e.g. image, flavor and key have to be selected in the drop-down menus. Also all the initial configuration can be changed there. For a full list of the supported stack parameters see the appendix below.

Option B) OpenStack Commandline Client

Source the openrc.sh file of your OpenStack tenant and create the heat stack the following way:

```
source Downloads/vbng-openrc.sh
Please enter your OpenStack Password for project vBNG as user bng:

openstack stack create -t Downloads/vbng.yaml vbngstack
```

In case you would like to overwrite default parameters with your custom values, try adding '--parameter <key=value>', e.g.:

```
openstack stack create -t Downloads/vbng.yaml vbngstack --parameter key='your_key' --parameter
flavor='your_flavor' --parameter image='your_image'
```

A full list of the supported stack parameter is shown in the following table:

Appendix: Stack Parameters

Key	Default Value	Description	Notes
OpenStack Settings			
key	vbng	Name of the SSH keypair for logging in into the instance	constraint: nova.keypair
image	"CentOS 7 x86_64 GenericCloud 1901"	Name of the glance image Supported are upstream cloud images for: Ubuntu 16.04 / Ubuntu 18.04 / CentOS 7	constraint: glance.image
flavor	a1.tiny	Flavor to use for the instance Can be a small one (1vCPU/4GB RAM/10GB disk)	constraint: nova.flavor
extnet	external	Name of external network	This is the existing OpenStack external network containing the floating IPs
int_cidr	192.168.1.0/24	Internal Network IPv4 Addressing in CIDR notation	Can be anything in the private IP space if your OpenStack supports overlapping IP tenant ranges.
dns1	8.8.8.8	DNS server 1 for internal network	E.g. DNS server 1 Openstack VMs will use
dns2	8.8.4.4	DNS server 2 for internal network	E.g. DNS server 2 Openstack VMs will use
vBNG Git Repository Settings			
git_repo	ssh://git@git.swisscom.com:7999/ztxgspon/vbng.git	Virtual BNG Git Repository URL (ssh://)	This repository holds the vbng code and is cloned by cloud-init
git_branch	master	Git branch to check out.	
git_sshkey	NOT SHOWN HERE	SSH Private Key for Git Repository (Read-Only Access)	For cloud-init read-only access
git_hostkey	NOT SHOWN HERE	SSH Host Key for Git Host (git.swisscom.com)	
vBNG Settings			
cust_cidr	10.66.0.0/16	Customer IPv4 Network in CIDR notation	The network for your subscribers
cust_gw	10.66.0.1	Customer IPv4 Network Gateway	The IPv4 gateway your subscribers will use
cust_dns	8.8.8.8	Customer DNS Server	The DNS severs your subscribers will use
cust_start	10.66.1.1	Customer IPv4 Range Start Address	Subscriber IP range for DHCP
cust_end	10.66.1.254	Customer IPv4 Range End Address	Subscriber IP range for DHCP
dhcp_l2	false	Global Layer 2 DHCP switch	Enable Layer 2 DHCP on datapath
dhcp_cidr	172.24.24.0/24	DHCP Server / Relay Network in CIDR notation	The network between The DHCP server and the DHCP L3 Relay on the OLT.
dhcp_ip	172.24.24.1	DHCP Server IPv4 Address	The DHCP Server is binding/listening to that address

in_tun_port	4789	UDP Port for incoming VxLAN Tunnels	For incoming VxLAN UDP packets. Used to configure OpenStack Security Groups
onap_dcae_ves_collector_url	http://172.30.0.126:30235/eventListener/v7	ONAP DCAE VES Collector URL	The URL the VES agent is streaming VES to

vBNG Initial Configuration by cloud-init

Once the stack is created by heat, cloud-init user data script checks out the vbng git repository and runs the scripts `00-installdeps.sh, 01-setupdatapath.sh, 02-setupcontainers.sh` contained in the repository. The parameters passed to them are kept in `$HOME/vbng.conf`. Once cloud-init finished its job it will create the file `$HOME/vbng_provisioning_done` on your instance. Logs are kept in `/var/log/cloud-init-output.log`. You may re-run those scripts as many times as you wish, work will only be done once. For example you have to re-run these 3 scripts on instance reboot. Keep in mind, you may require a reboot in case you have kernel updates installed.

- vbng/00-installdeps.sh
    - Update the system, install dependent packages, install and setup docker.
- vbng/01-setupdatapath.sh
    - Set up the datapath part, including shaping, routing and NAT.
- vbng/02-setupcontainers.sh
    - Create docker images and start all containers: Database, Message Queue, Restconf Server, VES Agent and DHCP Server.

OLT Configuration

OLT onboarding configuration is not done by cloud-init, since OLT parameters are normally not known at stack creation time. With the newest version there is a REST interface for configuring OLTs. See REST API documentation below: '**POST CreateOLT**', '**GET GetOLT**', '**POST DeleteOLT**'. To assist with working with the REST API, you can utilize Postman, and import the current API Collection.

ONT/Subscriber Configuration

Subscribers are usually configured by calls to bbs-edge-restconf-server directly from ONAP. In case you would like to test this functionality you can of course trigger this directly with the REST API calls defined below.

Important parameters are: **"remote_id":"AC9.000.990.001","s_vlan":10,"c_vlan":333** .Of course the values configured must match what the OLT/ONT in the Lab sends.

Currently only 4 subscribers profiles are supported (1/2/3/4), 2 * 100Mbit/s symmetrical and 2 * 20Mit/s symmetrical, respectively. This should be enough to run all test-cases for the BBS use-case.

REST API

To assist with working with the REST API, you can utilize Postman, and import the current API Collection. The information below documents each REST call that is available.

POST CreateInternetProfileInstance

Description: Creates a subscriber instance in the vBNG. This call will be used directly by ONAP. Note, the "service_id" MUST be unique, as this is used to identify the profile for updates and deletion.

Input Parameters / Body

```
{
        "remote_id":"AC9.000.990.002",
        "ont_sn":"serial",
        "service_type":"Internet",
        "mac":"00:00:00:00:00:00",
        "service_id":"2",
        "up_speed":"100",
        "down_speed":"100",
        "s_vlan":10,
        "c_vlan":334
}
```

curl example call

```
curl --location --request POST "{{host}}:{{port}}/CreateInternetProfileInstance" \
  --header "Content-Type: application/json" \
  --data "{
        \"remote_id\":\"AC9.000.990.001\",
        \"ont_sn\":\"serial\",
        \"service_type\":\"Internet\",
        \"mac\":\"00:00:00:00:00:00\",
        \"service_id\":\"1\",
        \"up_speed\":\"100\",
        \"down_speed\":\"100\",
        \"s_vlan\":10,
        \"c_vlan\":333
}"
```

GET GetInternetProfileInstance

Description: Returns list of all configured subscribers in the vBNG.

curl example call

```
curl --location --request GET "{{host}}:{{port}}/GetInternetProfileInstance"
```

POST ChangeInternetProfileInstance

Description: Updates an existing subscriber instance in the vBNG. The "service_id" parameter is used as the key to identify the specific profile to be updated.

Input Parameters / Body

```
{
        "remote_id":"AC9.000.990.001",
        "ont_sn":"serial",
        "service_type":"Internet",
        "mac":"00:00:00:00:00:00",
        "service_id":"1",
        "up_speed":"100",
        "down_speed":"100",
        "s_vlan":10,
        "c_vlan":333
}
```

curl example call

```
curl --location --request POST "{{host}}:{{port}}/ChangeInternetProfileInstance" \
  --header "Content-Type: application/json" \
  --data "{
        \"remote_id\":\"AC9.000.990.001\",
        \"ont_sn\":\"serial\",
        \"service_type\":\"Internet\",
        \"mac\":\"00:00:00:00:00:00\",
        \"service_id\":\"1\",
        \"up_speed\":\"100\",
        \"down_speed\":\"100\",
        \"s_vlan\":10,
        \"c_vlan\":333
}"
```

POST DeleteInternetProfileInstance

Description: Deletes an existing subscruber instance in the vBNG

Input Parameters / Body

```
{
        "service_id":"1"
}
```

curl example call

```
curl --location --request POST "{{host}}:{{port}}/DeleteInternetProfileInstance" \
  --header "Content-Type: application/json" \
  --data "{
        \"service_id\":\"1\"
}"
```

POST CreateOLT

Description: Creates and onboards an OLT into the vBNG.

Input Parameters / Body

```
{
        "data_dest_ip":"192.168.201.10",
        "data_dest_port":"4790",
        "data_vni":"12022",
        "dhcp_dest_ip":"192.168.201.10",
        "dhcp_dest_port":"4790",
        "dhcp_vni":"12023",
        "relay_north_ip":"172.24.24.2",
        "relay_south_ip":"10.66.0.2",
        "dhcp_l2_only":true,
        "s_vlan":300
}
```

curl example call

```
curl --location --request POST "{{host}}:{{port}}/CreateOLT" \
  --header "Content-Type: application/json" \
  --data "{
        \"data_dest_ip\":\"192.168.201.10\",
        \"data_dest_port\":\"4790\",
        \"data_vni\":\"12022\",
        \"dhcp_dest_ip\":\"192.168.201.10\",
        \"dhcp_dest_port\":\"4790\",
        \"dhcp_vni\":\"12023\",
        \"relay_north_ip\":\"172.24.24.2\",
        \"relay_south_ip\":\"10.66.0.2\",
        \"dhcp_l2_only\":true,
        \"s_vlan\":300
}"
```

GET GetOLT

Description: Returns list of all the OLTs currently configured in the vBNG.

curl example call

```
curl --location --request GET "{{host}}:{{port}}/GetOLT" \
  --header "Content-Type: application/json"
```

POST DeleteOLT

Description: Deletes an OLT instance from the vBNG. Note, "olt_id" parameter is auto-generated by *CreateOLT* API call and can be retrieved via the *GetOLT* API call.

```
{
        "olt_id":1
}
```

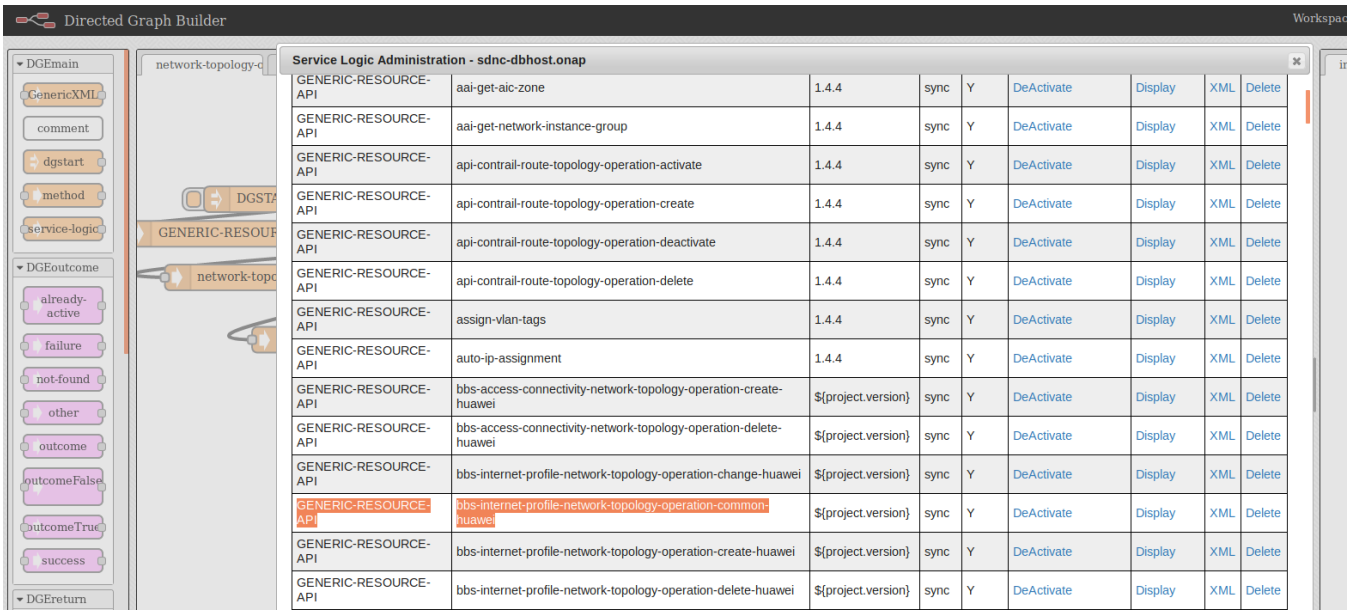
curl example call

```
curl --location --request POST "{{host}}:{{port}}/DeleteOLT" \
  --header "Content-Type: application/json" \
  --data "{
        \"olt_id\":1
}"
```

ONAP Configuration

The installation and initial configuration of Edge SDN M&C + vBNG is done by an Heat stack template, see above. The parameters which must be modified in ONAP are the following:

- The IP of Edge SDN M&C in order to be accessed from SDN-C is currently hardcoded in the DG -> **GENERIC-RESOURCE-API_bbs-internet-profile-network-topology-operation-common-huawei.json (<parameter name='prop.sdncRestApi.thirdpartySdnc.url' value='http://172.30.0.121:5000' />).** The Edge SDN M&C external controller is not registered in ESR for this release. Note: The IP above is provided by Heat stack output, it is the Floating IP of the vBNG instance in Swisscoms Lab.
- To update the IP of Edge SDN M&C in the corresponding DG, one must export the relevant DG mentioned earlier, update the IP, import back and finally enable the DG.



GENERIC-RESOURCE-API	aai-get-aic-zone	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	aai-get-network-instance-group	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	api-contrail-route-topology-operation-activate	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	api-contrail-route-topology-operation-create	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	api-contrail-route-topology-operation-deactivate	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	api-contrail-route-topology-operation-delete	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	assign-vlan-tags	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	auto-ip-assignment	1.4.4	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-access-connectivity-network-topology-operation-create-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-access-connectivity-network-topology-operation-delete-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-internet-profile-network-topology-operation-change-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-internet-profile-network-topology-operation-common-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-internet-profile-network-topology-operation-create-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete
GENERIC-RESOURCE-API	bbs-internet-profile-network-topology-operation-delete-huawei	${project.version}	sync	Y	DeActivate	Display	XML	Delete

Setup Transport Middle Box for VxLAN Tunneling

We built our middle-boxes on top of CentOS 7. Ubuntu and other distributions will work in a similar way. The commands shown here refer to CentOS 7. The middle box can be any x86 server with two 10Gbit/s NICs. One NIC will be facing the OLT on L2, the other NIC will be in the external network to communicate with the vBNG. To set up such a middle box use these commands to configure on top of a minimal CentOS 7 installation:

- Copy the team members SSH public keys and disable SSH password auth:

```
cat > authorized_keys << __EOF
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCjD
/+BIg4c28lHlHw464vbfUYjfDJ1sSKgrEYcMkL+qO6LagkDAWkWdelmAmpcUJlOPYjxDwmKj8Bu6/fd+WfVzk6y33YVmAFN4jAmv
/87dYCNuAMr4gDWc3cU5lsNdpsPzQqGUCFfJCvldyUZeu21YZ2rkYB1+Q9VObUSaa5Z74sKNYQJi0AgnZh63cYOyqVDCwIloWd2FzC+4o
04cVL3P1R+COGRq1EUUmy5LSI9rsCO59mLCt8Wm4h5OiY84nEbQVZUH3QyYw/ihmGm2qtklkbNMPOPZ7+8ZN5+of4u
/7bpEiZk3FcMh7lYwi6dMyUzwv47Il633JP6GDgOxuCH Daniel Balsiger SSH
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDH8lM+qleGIvXI3wgqIp73pKZwwxKfr9BDCdoVP3/zWRQ
/7zpw98nvx7gqfVLlt+P2TjxHbSJqGrSECSmKFCHsYzuA+khmg/aca
/IQa2FYFpUR1sT4czWQC14PiGGIoSbMukeUZvddZwZlalNZmOKjzY1Flz3w7+W+XHyFuwy6qfaIt1hIBKkqTUxECYq0O6OkdK6gzouKuA
Y/4AM+VvcIkdHMm9x3LCXWBAH24QzCG
/IzydqXfi4FkVtmGJv2AgEMyR0seSoU3drCXvpY91WjXT8i6m7EMB739hw0V32UaqslY3qHtuNTGake5JFWJn9zYF6lZwGXpU94Bw7YjQ
L1 Michail Salichos SSH
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCRxCsboa1ERMgiJCP2iA8Zcm2LuAOALQHIZIQEvbcwMifdeXMTawC0tDnU6qy35q+cr5W3+4HJD
yBLSAKmDosZepm1a/27cRlgXK/vtkxM5UlDk+lZsF/YGXBzZvWepM4XhozzCMNfvWWxkz5SnEl
/ZYfdN2H5psXReNTgBX33ax2cI+aOBZxsX2Y0FYBuqlJFT7htgblGjHLq43nL
/cF9w9cXkMv+mPUQJN4wNf1HU5JBjX6sKl6Y3IIPxEVGFohu8c9tDHa8JoWxIzKZz3z9Zd8KkfTTsRtXh3MH7mMRZkVTgHHVU3NA4
/psEVMJHFtXI6R/laOv8Lpytdky7tkv taapeda0@UM01183

__EOF
mkdir .ssh
chmod 0700 .ssh
cp authorized_keys .ssh # copy not move (selinux)
chmod 0600 .ssh/authorized_keys
rm -f authorized_keys
sed -e 's|^PasswordAuthentication yes|PasswordAuthentication no|' -i /etc/ssh/sshd_config
systemctl restart sshd
```

- Disable NetworkManager, Firewalld and Postfix services, enable legacy networking:

```
systemctl disable NetworkManager
systemctl stop NetworkManager
systemctl disable firewalld
systemctl stop  firewalld
systemctl disable postfix
systemctl stop postfix

systemctl enable network
systemctl start network
```

- Create Network Interface Configuration Files in /etc/sysconfig/network-scripts/ :
  - ifcfg-bridge:

    ```
    DEVICE=bride
    TYPE=Bridge
    MTU=1400
    ONBOOT=yes
    BOOTPROTO=none
    IPV6INIT=no
    IPV6_AUTOCONF=no
    ```

  - ifcfg-nic1 (facing OLT):

    ```
    DEVICE=nic1
    TYPE=Ethernet
    MTU=1400
    ONBOOT=yes
    BOOTPROTO=none
    IPV6INIT=no
    IPV6_AUTOCONF=no
    BRIDGE=bridge
    ```

  - ifcfg-nic2 (in external network, facing vBNG):

    ```
    DEVICE=nic2
    TYPE=Ethernet
    MTU=1450
    ONBOOT=yes
    BOOTPROTO=none
    IPV6INIT=no
    IPV6_AUTOCONF=no
    IPADDR=172.30.0.252
    PREFIX=24
    DEFROUTE=yes
    GATEWAY=172.30.0.1
    DNS1=8.8.8.8
    DNS2=8.8.4.4
    ```

- Create VxLAN Tunnel Interface on bridge creation:

```
cat > /sbin/ifup-local << __EOF
#!/bin/sh
if [[ "\$1" == "bridge" ]]
then
  ip link add vxlan0 type vxlan id 88888 local 172.30.0.252 remote 172.30.0.121 dstport 4789 dev nic2
  ip li set up dev vxlan0
  ip link set master bridge dev vxlan0
fi
__EOF

chmod 755 /sbin/ifup-local
restorecon -Fv /sbin/ifup-local
```

Once those files are in place the configuration is reboot persistent. To have a sane state, please reboot the box once, after having created those files.