TR-383, TR-385, WT-451 NETCONF Server

Revision History

Revision	Date	Change Description
v1.0	25-July-2019	Initial draft
v1.1	9-Dec 2019	OB-BAA 2.0 adjustments
v1.2	15-July 2020	- Added support for new release of netopeer2 toolkit, that uses shared memory-based sysrepo instead of sysrepo-daemon - Added support for WT-451

Table of Contents

1. Scope

This document describes OLT NETCONF Server reference implementation

2. References

2.1. TR-383 Amendment 2. Common YANG Modules for Access Networks
2.2. TR-385 ITU-T PON YANG Modules
2.3. WT-451 vOMCI Specification
2.4. sysrepo - YANG-based configuration and operational state data store for Unix/Linux applications
2.5. Netopeer2 – The NETCONF Toolset
2.6. OB-BAA BBWF Demonstration Script

3. Overview

- BBF TR-385 became a standard in the beginning of 2019. To date it is the only standard for ITU PON OLT and ONU management in context of SDN. There are at least 2 open source implementations based on TR-385: VOLTHA and OB-BAA
- VOLTHA and OB-BAA use different approaches to E2E data path provisioning, from OLT NNI to ONU UNI
    - OB-BAA uses BBF TR-383 Common Yang which is already used for xDSL and DPU.
    - VOLTHA uses OpenFlow. Note that to date
        - OpenFlow pipeline is not standardized
        - There are no clear means for provisioning non-trivial QoS

Broadcom BCM686xx SDK is already integrated with VOLTHA by 3rd parties. This document describes a reference NETCONF server implementation that configures everything including data path using NETCONF (OB-BAA approach).
Broadcom NETCONF server supports a subset of TR-385 and 383 on BCM6862x (Maple) and BCM6865x (Aspen) platforms. This server can be used

- With OB-BAA
- As a stand-alone application managed by proprietary management stack developed by customers (hypothetical at this point)
- For SDN demos

In addition to TR-383 and TR-385, Broadcom NETCONF server also includes support for both NETCONF/YANG and gRPC interfaces defined in WT-451 vOMCI Specification.
The NETCONF server can either provision ONU using embedded OMCI stack, or act as WT-451 pOLT and enable dis-aggregated ONU provisioning by vOMCI using gRPC.

4. NETCONF Server Implementation

4.1. Implementation Overview

Broadcom NETCONF Server solution is based on sysrepo and netopeer2 toolkits (see references).

- sysrepo provides high-performance NETCONF/YANG database services with shared access by multiple applications. When sysrepo-managed database is about to change (e.g., by request from netopeer2-server), sysrepo notifies the relevant subscribers using a sequence of remote callbacks (validate, commit, etc.).
- netopeer2-server implements NETCONF protocol

The NETCONF server solution consists of

- netopeer2-server that implements NETCONF protocol. This is an of-the-shelve component available as open-source
- Broadcom NETCONF adapter that translates NETCONF/YANG events into BCM686xx SDK 3.6 API calls

OLT Netconf Adapter is a stand-alone application

- It is integrated with sysrepo using sysrepo client library
    - Subscribes for notifications (data base change validation and commit events for the specified xPaths)
    - Reports events (e.g., ONU discovery)
- Is integrated with BCM686xx SDK 3.6 SDK
    - Translates sysrepo events to the SDK API calls
    - Subscribes for autonomous indications
- It is integrated with BCM686xx ONU Management SDK (embedded OMCI stack)
    - Translates sysrepo events to the ONU Management SDK API calls, which in turn translates to OMCI
- It supports managing ONUs using either embedded OMCI stack or alternatively it can act as WT-451 pOLT
    - Supports multiple vOMCI connections simultaneously

    - Supports gRPC client and server endpoints
    - Supports gRPC client and server endpoint filters that automatically associate ONU with a matching vOMCI endpoint
    - gRPC client and server endpoints and filters can be provisioned via NETCONF WT-451 YANG model or using CLI
- It supports a DHCP relay functionality with ability to insert DHCP Option 82 in the upstream and strip it in the downstream

NETCONF server can be built with 2 sets of YANG models, as specified at build time (see HowTo Build and Run chapter). There is a slight difference between the 2 (OB-BAA team included some non standard models and also made a few changes).

- Standard TR-383, TR-385, WT-451 set which includes only YANG models that are actually supported by the server
- OB-BAA 3.1 bundle should be used when building the server to work with OB-BAA

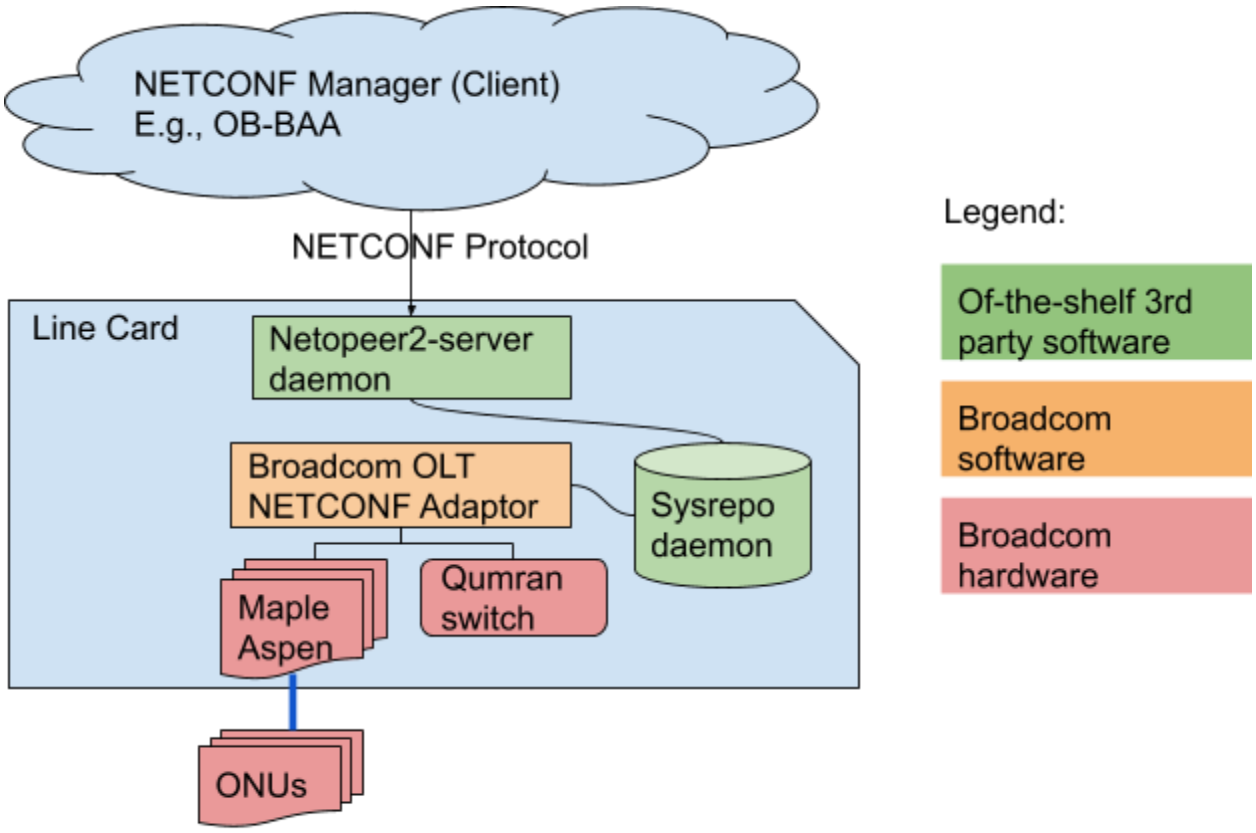

Figure1: Netconf Server on a Line Card

As shown in the diagrams above, OLT NETCONF Server is a stand-alone application that can run on a line card or a linux VM in the data centre.
Highlights:

- NETCONF server implements a number of YANG objects (containers) and their attributes. In some cases attributes are recorded in the internal database FFU, but not applied to OLT and ONU configuration
- NETCONF server uses hash tables for internal object data store and is, therefore, scalable to large systems
- The server was tested with XGS-PON only. It will probably work for GPON and XGPON1. NGPON2 requires further work
- The server supports CLI and includes API and ONU_MGMT CLI directories, same as example_user_appl included in BCM686xx SDK 3.x
- Integrated with logger and linux syslog
- Extensive error checking and detailed error reporting back to the netconf manager (client)

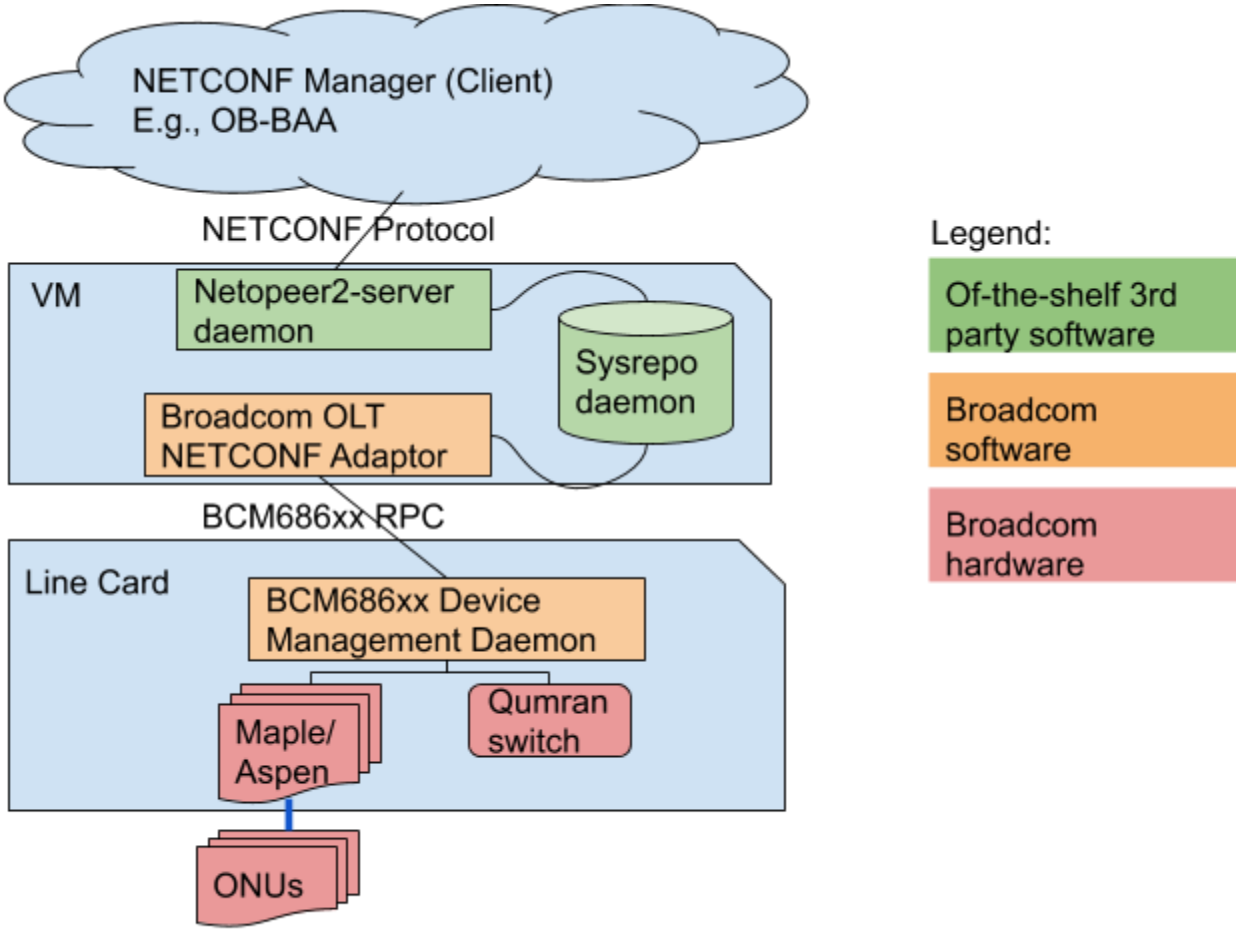

Figure 2: Netconf Server on VM in the Data Centre

4.2. YANG Objects

OLT Netconf Server supports the following YANG objects. Most of the objects are implemented only partially.

4.2.1. /ietf-hardware:hardware/component (bbf-hardware.yang)

NETCONF server records a number of different hardware classes:

- chassis
- board
- cage
- transceiver
- transceiver-link

The following leafs are recorded:

- parent
- parent-rel-pos - for transceiver and transceiver-link this attribute is used to identify 1-based PON number on the front panel
- expected-model - potentially can be used to provision transceiver type. However, it is not implemented

4.2.2. /ietf-interfaces:interface/channel-group (bbf-xpon.yang)

Supported attributes are

- polling-interval - mapped to *pon_interface.service_discovery*

4.2.3. /ietf-interfaces:interface/channel-partition (bbf-xpon.yang)

Supported attributes are:

- channel-group-ref - refers to /ietf-interfaces:interface/channel-group object

4.2.4. /ietf-interfaces:interface/channel-pair (bbf-xpon.yang)

Supported attributes are:

- channel-group-ref - refers to /ietf-interfaces:interface/channel-group object
- channel-partition-ref - refers to /ietf-interfaces:interface/channel-partition object
- wavelength-profile-ref - refers to /bbf-xpon:wavelength-profile. Currently unused

4.2.5. /ietf-interfaces:interface/channel-termination (bbf-xpon.yang) - PON interface

Supported attributes are:

- xgs-pon-id, xgpon-pon-id
- enabled : enable/disable PON interface
- port-layer-if - refers to /ietf-hardware:component of class “transceiver-link” which should contain parent-rel-pos attribute. parent-rel-pos-1 is interpreted as pon_ni.
- channel-pair-ref - refers to /ietf-interfaces:interface/channel-pair object

4.2.6. /ietf-interfaces:interface/v-ani (bbf-xpon.yang) - ONU configuration at OLT

Supported attributes are:

- enabled
- channel-partition - refers to /ietf-interfaces:interface/channel-partition object
- preferred-channel-pair - refers to /ietf-interfaces:interface/channel-pair object
- protection-channel-pair - refers to /ietf-interfaces:interface/channel-pair object
- onu-id. Note that this attribute is optional. If not set, onu_id is auto-assigned
- serial-number
- registration-id

4.2.7. /ietf-interfaces:interface/ani (bbf-xpon.yang) - ONU configuration

Supported attributes are:

- onu-id
- upstream-fec
- management-gem-port-aes-indicator
- management-gemport-id

The attributes are recorded, but NOT mapped to configuration.

4.2.8. /ietf-interfaces:interface/olt-v-enet (bbf-xpon.yang) - ONU UNI configuration at OLT

Supported attributes are:

- lower-layer-interface - refers to /ietf-interfaces:interface/v-ani

This object is used as part of data path and QoS configuration. Vlan sub-interfaces refer to using their lower-layer-interface attribute.

4.2.9. /ietf-interfaces:interface/onu-v-enet (bbf-xpon.yang) - ONU UNI configuration (Option 1)

Supported attributes are:

- ani - refers to /ietf-interfaces:interface/ani

This object is used as part of data path and QoS configuration. Vlan sub-interfaces refer to using their lower-layer-interface attribute.

4.2.10. /ietf-interfaces:interface/[enet] (ietf-interfaces.yang) - OLT NNI and ONU UNI

Supported attributes are:

- lower-layer-interface - used on the ONU side, refers to /ietf-interfaces:interface/ani object
- interface-usage - set to 'network-port' for OLT NNI

This object is used as part of data path and QoS configuration. Vlan sub-interfaces refer to using their lower-layer-interface attribute.

4.2.11. /ietf-interfaces:interface/vlan-sub-interface (bbf-sub-interfaces.yang) - OLT NNI and ONU UNI

Supported attributes are:

- subif-lower-layer/interface - references interface object (enet, v-onu-v-enet, onu-v-enet)
- inline-frame-processing/ingress-rule/rule
    - flexible-match - complex attribute with various match criteria
    - Ingress-rewrite - ingress header modification rules
- engress-rewrite - egress header modification rules
- interface-usage (network or access)
- ingress-qos-policy-profile - refers to /bbf-qos-policy-profiles:qos-policy-profiles/policy-profile

This object underpins data path configuration. See chapter Data Path Configuration,

4.2.12. /bbf-xpongemtcont:xpongemtcont/traffic-descriptor-profiles - TCONT SLA

Supported attributes are:

- fixed-bandwidth

- assured-bandwidth
- maximum-bandwidth
- additional-bandwidth-eligibility-indicator
- priority
- weight

4.2.13. /bbf-xpongemtcont:xpongemtcont/tconts/**tcont**

Supported attributes are:

- alloc-id. Optional. Auto-assigned if not set
- interface-reference - refers to /ietf-interfaces:interface/v-ani object
- traffic-descriptor-profile-ref - refers to /bbf-xpongemtcont:xpongemtcont/traffic-descriptor-profiles/traffic-dewscriptor-profile object

4.2.14. /bbf-xpongemtcont:xpongemtcont/gemports/**gemport,** /bbf-xpon:xpon/multicast-gemports/**gemport**

Supported attributes are:

- gemport-id. Optional. Auto-assigned if not set
- interface - refers to /ietf-interfaces:interface/v-ani object
- tcont-ref - refers to /bbf-xpongemtcont:xpongemtcont/tconts/tcont object
- traffic-class - traffic class 0-7. Used for GEM port selection when creating data flows
- downstream-aes-indicator
- upstream-aes-indicator
- is-broadcast

4.2.15. /bbf-xpon:xpon/wavelength-profiles/wavelength-profile

This object is relevant for NGPON2. Attributes are recorded FFU, but not applied.

4.2.16. /bbf-qos-classifiers:classifiers

Supported attributes are:

- match-criteria, filter-operation - only MATCH ANY is supported. Non-trivial match criteria require support in BAL. See "BAL QoS Extension Proposal
- scheduling-traffic-class - this attribute is used for GEM port selection

4.2.17. /bbf-qos-policies:qos-policy

Supported attributes are:

- classifiers - list of references to /bbf-qos-classifiers:classifiers objects

4.2.18. /bbf-qos-policy-profiles:qos-policy-profiles

Supported attributes are:

- policy-list- list of references to /bbf-qos-policies:qos-policy objects

4.2.19. /bbf-link-table:link-table/link-table

This table is used to link OLT and ONU - side interfaces together

- v-ani with ani
- v-onu-enet with onu-enet or enet

4.2.20. /bbf-l2-forwarding:forwarders/forwarder

Each forwarder is a bridge that ties together 2 or more OLT interface objects

- sub-interface on OLT NNI
- sub-interface(s) on v-onu-enet. Just 1 such sub-interface for 1:1 configuration, multiple for N:1

The implementation currently supports only 1:1 configuration. Attributes required for N:1 are recorded, however attempt to apply such configuration is rejected with error NOT-SUPPORTED

4.2.21. /bbf-polt-vomci:remote-nf-settings/nf-server

WT-451 pOLT server endpoint provisioning

- Endpoint name, local address and local port
- Endpoint filters: ANY, by ONU vendor, by ONU serial number

4.2.22. /bbf-polt-vomci:remote-nf-settings/nf-client

WT-451 pOLT client endpoint provisioning

- Endpoint name, local address and local port
- Endpoint filters: ANY, by ONU vendor, by ONU serial number

4.3. Code Structure

The server is implemented in

- /aspen/main/host/netconf_server - server's "main"
- /aspen/main/host/netconf_server/modules/bbf-xpon - all objects above
- /aspen/main/host/netconf_server/netopeer2-cli-scripts - example configuration scripts

The implementation depends on

- a number of 3rd party packages in the third_party directory. In particular, libyang, sysrepo, libnetconf2 and netopeer2
- yang models in third_party/yang-models. All relevant models are imported into sysrepo data store at build time
- All objects are stored in object hash, which can be manipulated using xpon_object_add(), xpon_object_delete(), xpon_object_get(), xpon_object_get_or_add() functions
- Typically there is 1 .c file per YANG object type. In some cases a single .c file implements 2-3 related objects.
    - For each object there are functions “init”, “start”, “exit”, “get_by_name”, and “delete”
    - For some objects there are also additional functions, usually for looking up an object by some criteria other than its name (e.g., gem_get_by_traffic_class(), v_ani_get_by_id(), etc.)
- Implementation behavior is described in “Typical Configuration Sequence” chapter.

4.4. Implementation Restrictions and Further Development

NETCONF server development is provided as a reference. The following incomplete list includes features that are NOT implemented

- NGPON2 support
- Protection support
- Non-trivial QoS classification rules (requires support in BAL)
- Additional classification criteria for forwarding (by Ethernet address, IP protocol)
- Additional forwarding modes
    - N:1 unicast
    - N:1 multicast
    - TLS
- Alarm reporting

5. Typical Configuration Sequence

In the below description NETCONF server actions performed in response to configuration changes are shown in this color.

5.1. OLT Side Configuration

BBF opted to use the same configuration sequence for all ITU PON flavours, although many parameters are only relevant for NGPON2.

- Create *wavelength-profile* (only needed for NGPON2 and XGS PON)
- Create *channel-group, channel-partition and channel-pair* objects. The meaning of those objects is defined in ITU G.988
- Create *channel-termination* object (that represents PON interface)

    - At this point NETCONF server provisions and activates pon_interface object
    - When channel-termination is disabled/deleted, pon_interface is deactivated/cleared
    - By default ONU discovery is ON, unless disabled by setting channel-group.polling-interval=0
    - Discovered ONUs are reported using /bbf-xpon-onu-states:onu-state-change notification
- Create OLT NNI interface(s) (usually "enet" interface object with "interface-usage==network-interface"
- Create *v-ani* object (that represents ONU)
    - At this point NETCONF server provisions and activates ONU object
    - Once ONU activation is completed, NETCONF server
        - Reports /bbf-xpon-onu-states:onu-state-change notification
        - Initiates OMCI MIB upload and initial MIB configuration using ONU management API
- Create *traffic-descriptor-profile* object (TCONT SLA)
- Create *tcont* object
    - At this point NETCONF server configures itupon_alloc_id object
- Create *gemport* object
    - At this point NETCONF server configures itupon_gem_port object
- Create *qos-classifier, qos-policy and qos-policy-profile*. Each *qos-classifier* includes traffic-class attribute that will be used for GEM identification
- Create *vlan-sub-interface* object pointing to OLT NNI interface. This sub-interface includes classification rules and actions for downstream BAL flow. The object CAN refer to *qos-policy-profile*
- Create *vlan-sub-interface* object pointing to v-onu-v-enet interface. This sub-interface includes classification rules and actions for upstream BAL flow. The object CAN refer to *qos-policy-profile*
- Create *forwarder* object referring to the downstream and upstream vlan-sub-interfaces above
    - At this point NETCONF server creates upstream and downstream BAL flows. The following sequence is executed for each vlan-sub-interface/ingres-rule
        - If DS and/or US object refers to qos-policy-profile, use this profile. Otherwise, try to locate qos-policy-profile via linked ONU side objects
            - Go over sub-interfaces on onu-enet object linked with v-onu-v-enet using link-table (see ONU configuration below). Find sub-interface such that its egress packet structure matches OLT's US vlan-sub-interface's ingress rule
        - Find matching qos-classifier on the qos-policy-profile identified in the previous step
        - Find gemport object by qos-classifier/traffic-class

        - Create downstream and upstream BAL flow using information in vlan-sub-interface/ingress-rule in downstream and upstream vlan-sub-interface respectively
            - Map vlan-sub-interface/ingress-rule/flexible match to BAL flow.classifier
            - Create “flow action” by combining vlan-sub-interface/ingress-rule/ingress-rewrite with vlan-sub-interface/egress-rewrite
            - Map the “flow-action” above to BAL flow.action
            - Create BAL flow
        - Check if v-ani-v-enet is linked with ONU-side interface and ONU has matching sub-interfaces. Create ONU flow if yes (see in the next section)

5.2. ONU Side Configuration

ONU side configuration can be done after OLT configuration, or interleaved. Netopeer2-cli examples in netconf_server/netopeer2-cli_scripts interleave OLT and ONU configuration. On the other hand, BBWF demo example script creates the entire OLT configuration first, then ONU configuration.

- Create *ani* object (that represents ONU)
- Create *link-table/link-table* entry that links the *ani* and OLT’s *v-ani* objects
- Create *enet* object (that represents ONU UNI)
- Create *link-table/link-table* entry that links the *enet* and OLT’s *v-ani-v-enet* objects.
    - At this point NETCONF server checks if all information needed for creating ONU flow(s) is present and create ONU flow(s) if yes. If not all information is available, ONU flow creation is postponed (non an error), For each *ingress-rule* on each *vlan-sub-interface* on the *enet* interface it checks that
        - The ingress rule can be matched with one of ingress-rules on one of *vlan-sub-interface*s on the linked *v-ani-v-enet* object. The matching is done by comparing egress packet structure produced by ONU’s rule with match criteria in the OLT rule
        - It is possible to find qos-classifier and gemport by qos-classifier/traffic-class

- Create *qos-classifier, qos-policy and qos-policy-profile*. Each *qos-classifier* includes traffic-class attribute that will be used for GEM identification. Note that NETCONF server supports having qos configuration on only 1 side (either OLT or ONU) and locating the relevant qos-profile via linked interfaces
- Create *vlan-sub-interface* object pointing to the *enet* interface. This sub-interface includes classification rules and actions for upstream ONU flow. The object CAN refer to *qos-policy-profile.*
    - At this point NETCONF server goes over the vlan-sub-interface/ingress-rules, does checks described in link-table bullet above and creates ONU flow(s) if all necessary information is available

6. HowTo Build and Run

NETCONF server includes a number of examples in host_reference/netconf_server/netopeer2-cli_script directory. The scripts are numbered in the suggested execution order. All numbered scripts are derived from OB-BAA OLT configuration examples.

6.1. Build

Open-source version of the NETCONF server can be built for an Edgecore Whitebox OLT that uses an Open Network Linux (ONL) operating system.
The toolchain for the Edgecore Whitebox board is maintained by the ONL as a docker image.

1. Install the binfmt-support kernel module with the following command as root:
   #> apt-get install binfmt-support
2. To install Docker on the build platform, refer to the procedure at:
   https://docs.docker.com/engine/installation/linux/docker-ce/debian/
3. Download the ONL Source CodeFrom Github
   ```
   #> git clone https://github.com/opencomputeproject/
                    OpenNetworkLinux.git
   ```
4. Pull docker container containing ONL toolchain
   #> cd OpenNetworkLinux & docker/tools/onlbuilder -8
5. Update cmake in the docker container to at least 3.5
6. Build inside the docker
   #> make BOARD=asfvolt16

It will build all NETCONF server components for asfvolt16 (Edgecore) board

Broadcom BCM686xx SDK 3.x licensees can build the NETCONF server for x86 linux or for their target board. As a prerequisite, NETCONF_SERVER and ONU_MGMT optional packages must be unpacked in the same directory as the rest of the BCM686xx SDK 3.x

- make BOARD=sim NETCONF_SERVER=y NO_BAL=y TR451_VOMCI_POLT=y
- make BOARD=<target_board> PLATFORM=maple NETCONF_SERVER=y NO_BAL=n TR451_VOMCI_POLT=y

When running on x86 linux VM, netconf server requires that the OLT run 3.x SDK in full line card mode (with BAL).
By default netconf server is built with OB-BAA YANG model bundle. It can be built with standard TR-383, TR-385 model bundle instead using build option `USE_OBBAA_YANG_MODELS=n`
For example,

- make BOARD=sim NETCONF_SERVER=y NO_BAL=y US_OBBAA_YANG_MODELS=n TR451_VOMCI_POLT=y

6.2. Running netconf server on linux PC/VM

- start svk_init.sh on the board first
- start netopeer2-server on linux PC
    - build/fs/bin/start_netopeer2_server.sh
        - Additional command line options are:
            - -d - debug mode. Stay in the foreground
            - -v3 - verbose logging
            - -h - for more options
- Start Broadcom NETCONF adapter on the same linux PC where netopeer2-server is running
    - build/fs/start_netconf_server.sh -device board-ip-address:50200
        - Additional command line options are:
            - -d - debug mode. Stay in the foreground
            - -log debug|info|error - initial log level
            - -syslog - log to syslog
            - -tr451_polt - disable embedded OMCI stack and enable WT-451 support
            - -h - for more options
- Connect to maple/aspen in XGS mode using CLI, as usual
- Start netopeer2-cli on the linux build machine
    - build/host-sim/fs/bin/sysrepotool.sh build/fs/bin/netopeer2-cli
    - Connect to the running netconf server
        - connect --host <linux VM IP address> --port 10830 --login your-user-name
- Start running netopeer2-cli scripts (see below)

6.3. Running netconf server on the line card

The following files and directories have to be copied to the target board

- All regular files in build/fs/*
- build/fs/lib
- build/fs/bin
- build/fs/sysrepo

Once it is done, start-netopeer2-server.sh and start-netconf-server.sh can be started in the same way as in the previous section, except that when starting the netconf server, '-device board-ip-address:50200' parameter is not required.

6.4. Provision system using netopeer2-cli NETCONF client

- Connect to the running netconf server
    - edit-config --target running --defop merge --test test-then-set --config=netconf_server/netopeer2-cli_scripts/1.1-add-interfaces-olt.yc
    - Continue running scripts in order

