Quantum Networks
NETCONF introduction

Prepared on 2024-02-09

Table of Contents

1. What is NETCONF/YANG?

The Network Configuration Protocol (NETCONF) is a network management protocol developed and standardized by the IETF. The NETCONF base protocol was officially published as RFC 4741 in 2006, with a later revision published as RFC 6241 in 2011.

NETCONF is used to perform management functions, mainly targeted at provisioning but also capable of monitoring certain configuration and operational state information. It is based on a client-server architecture where one or more client applications reside within an ISPs operations center and many servers are deployed in the network embedded within network devices.

Some of the key NETCONF configuration management support features include:

- Configuration and state data distinction
- Multiple configuration data stores
- Configuration change validations
- Configuration change transactions
- Selective data retrieval with filtering
- Event notifications streaming and playback
- Extensible RPC mechanism

YANG is a data modeling language for the NETCONF network configuration protocol. It was developed by the NETMOD working group in the IETF and was published as RFC 6020 in October 2010. YANG features include:

- Human readability
- Hierarchical configuration data models
- Reusable groupings and types
- Augmentation mechanisms
- Versioning rules
- Supports defining operations
- Modularity of data (through modules and sub-modules)

The standardization of the NETCONF management protocol and YANG data modeling is one of IETF's ways of improving Internet architecture. Standardization is key in this, as it improves overall protocol compatibility and ensures that all communicating software understands each other.

2. What is Sysrepo?

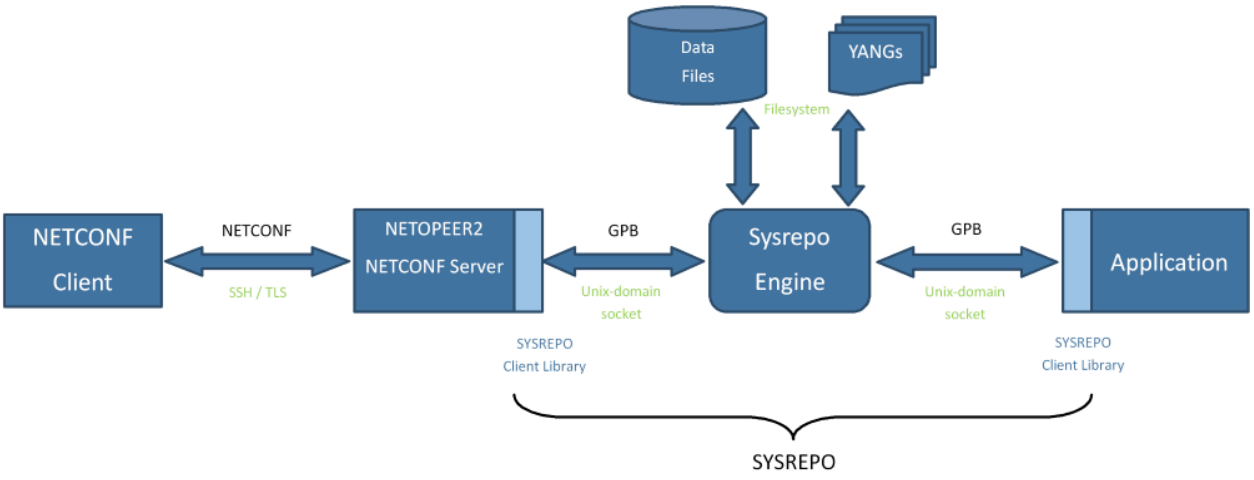
The Sysrepo project provides YANG-based datastore functionality to Unix/Linux applications. In doing so, Sysrepo enables applications to store their configuration in the form of provided YANG models instead of e.g flat configuration files. All interfaces using YANG conform to a standard definition common to all platforms that are manageable by the NETCONF standard. Furthermore, Sysrepo will enforce the data constraints defined by the YANG model, as well as ensure data consistency of all data stored in the datastore.

The Sysrepo project combines a YANG-based configuration and operational datastore (Sysrepo datastore) with a 2nd generation open source NETCONF server (Netopeer 2 NETCONF server).

3. Sysrepo Datastore

Sysrepo is a datastore for configuration and state data of Unix/Linux applications, which provides the following features:

- Centralized repository of YANG-modeled configuration files and state data
- Simple (XPath -based) API for accessing the configuration from applications
- Startup, running and candidate datastores support
- Full transaction and concurrency support, conforming all ACID properties
- Automatic data consistency and constraints enforcement according to the YANG models
- No single point of failure – applications do not need any other process to be running to access their configuration



4. Netopeer2 NETCONF Server

Netopeer2 is the 2nd generation of the well-known open-source NETCONF server that uses Sysrepo as the datastore implementation. Together they provide a complete NETCONF management framework for Linux.

Developed by the Tools for Monitoring and Configuration department of CESNET, the Netopeer2 is based on the new generation of NETCONF and YANG libraries — libyang and libnetconf2.

5. NETCONF / Netopeer2 integration into the QN Switches

Sartura has reviewed the Netopeer2 package and the related Sysrepo package stack which includes the projects:

- Netopeer2
- Sysrepo
- libyang

The entire stack has been integrated on Quantum Networks switches in an all-in-one package which includes Sysrepo and libyang. The switch device by default runs the Netopeer2 server which listens for NETCONF data requests. For example, Sartura's integration of the Netopeer2 package currently specifies Secure Shell (SSH) access which allows clients to connect on the switch port `830`.

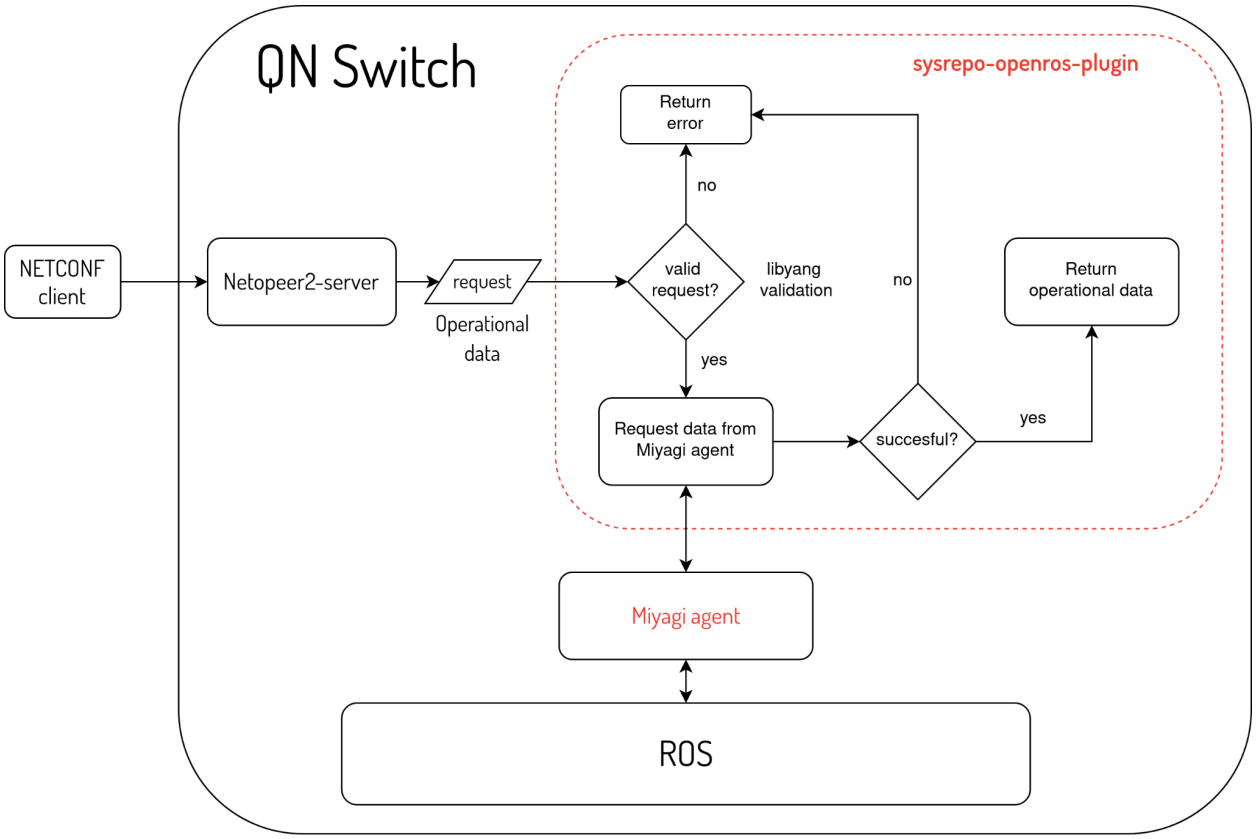
The barebones version of the packages does not allow a user to do actual changes on the system, as that part has to be implemented in a Sysrepo plugin.

As part of the integration process, Sartura has developed a demo Sysrepo plugin named `sysrepo-openros-plugin` that lets the user query the state of all the interfaces via the operational datastore (`name`, `if-index`, `description`, `type`, `oper-status`, `phys-address` and `speed`). This plugin makes use of the existing Miyagi agent infrastructure which has also been developed by Sartura in order to obtain interface data from ROS.

Because Sysrepo datastore requires a model of the data to be defined using YANG modeling language, Sartura has created an initial example YANG data model named `quantumnet-interfaces.yang`. This model has been created by Sartura from scratch in order to define a relatively simple representation of data from the switch. It is possible to use and extend standardized YANG models such as those published by IETF or OpenConfig.

While the possibility to change the system configuration is not yet present in the initial demo version of the plugin, it is possible to add such features with further development efforts.

A visual representation of the interaction between switch components including NETCONF, Miyagi agent and ROS is given below:



As can be seen from the above diagram, a NETCONF client initiates a connection to the Netopeer2 server enabled switch via a configured NETCONF transport layer (by default SSH). A request for operational data is then forwarded by the Netopeer2 server towards the `sysrepo-openros-plugin` which has an operational datastore subscription for the nodes defined in the `quantumnet-interfaces.yang` YANG model. The plugin requests the actual switch interface data from the Miyagi agent over the Miyagi UNIX socket. Miyagi agent grabs the data from ROS and passes it back to the `sysrepo-openros-plugin` which then returns it back to the user via Netopeer2 server.

6. Basic NETCONF client usage

To interface with the Netopeer2 server on the switch it is recommended to use a NETCONF client. For the purposes of this document we will be using the Netopeer2 client implementation `netopeer2-cli`.

To start the CLI execute the following command:

```
user@workstation ~ $ netopeer2-cli
>
```

The CLI has a lot of available commands which can be viewed by executing `help`:

```
> help
Available commands:
  auth           Manage SSH authentication options
  knownhosts     Manage the user knownhosts file
  cert           Manage trusted or your own certificates
  crl            Manage Certificate Revocation List directory
  outputformat   Set the output format of all the data
  searchpath     Set the search path for models
(...)
```

To initiate a connection to the switch execute the `connect` command as follows:

```
> connect --ssh --host 192.168.101.177 --login root
The authenticity of the host '192.168.101.177' cannot be established.
ssh-rsa key fingerprint is
c8:95:5e:a6:7c:cc:82:3b:59:6a:ba:3c:10:76:f4:8e:38:c7:61:54.
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.101.177 password:
(...)
>
```

Where `192.168.101.177` is the switch IPv4 address. Please note that the password for NETCONF login is the same as one required for Linux login.

Following a successful connection we can confirm the current status of the NETCONF session:

```
> status
Current NETCONF session:
  ID       : 2
  Host     : 192.168.101.177
  Port     : 830
  Transport   : SSH
  Capabilities:
      urn:ietf:params:netconf:base:1.0
(...)
```

To get the actual switch interface data it is recommended to use the command `get` with `--filter-xpath` as follows:

```
> get --filter-xpath //interface[name="te1/0/2"]
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <switch xmlns="urn:ietf:params:xml:ns:yang:quantumnet-interfaces">
      <interface>
      <name>te1/0/2</name>
      <oper-status>up</oper-status>
      <if-index>2</if-index>
      <phys-address>8c:ae:49:d0:20:22</phys-address>
      <speed>1000</speed>
      </interface>
  </switch>
</data>
```

Here we have used the XPath filtration to single out a specific port `te1/0/2`. To get the data for all interfaces do not use the XPath `name` matching:

```
> get --filter-xpath /switch
DATA
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <switch xmlns="urn:ietf:params:xml:ns:yang:quantumnet-interfaces">
      <interface>
      <name>te1/0/1</name>
      <oper-status>down</oper-status>
      <if-index>1</if-index>
      <phys-address>8c:ae:49:d0:20:21</phys-address>
      <speed>10000</speed>
      </interface>
      <interface>
      <name>te1/0/2</name>
      <oper-status>up</oper-status>
      <if-index>2</if-index>
      <phys-address>8c:ae:49:d0:20:22</phys-address>
      <speed>1000</speed>
      </interface>
      <interface>
      <name>te1/0/3</name>
      <oper-status>down</oper-status>
```

```
      <if-index>3</if-index>
      <phys-address>8c:ae:49:d0:20:23</phys-address>
      <speed>10000</speed>
      </interface>
      <interface>
      <name>te1/0/4</name>
      <oper-status>down</oper-status>
      <if-index>4</if-index>
      <phys-address>8c:ae:49:d0:20:24</phys-address>
      <speed>10000</speed>
      </interface>
      <interface>
      <name>te1/0/5</name>
      <oper-status>down</oper-status>
      <if-index>5</if-index>
      <phys-address>8c:ae:49:d0:20:25</phys-address>
      <speed>10000</speed>
      </interface>
(...)
```