OPEN NETWORKING
FOUNDATION

Third Wireless Transport SDN Proof of Concept Detailed Report

V1 released
2016-12-06

ONF Document Type: Detailed Report
ONF Document Name: Third Wireless Transport SDN Proof of Concept Detailed Report

Disclaimer

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

Glossary

CBRS Citizens Broadband Radio Service

MW microwave

NBI North-bound Interface

NE Network Element

ODL OpenDaylight SDN controller

ONF Open Network Fundation

OTWG Optical Transport Working Group

PoC Proof of Concept

SBI South-bound Interface

SDN Software Defined Network

WT Wireless Transport

WTP Wireless Transport Project

Executive Summary

This white paper provides an overview of the contents and results of Proof of Concept (PoC) conducted from 24th to 28th October 2016 by the Wireless Transport Project of the Open Networking Foundation (ONF) in North Brunswitck (New Jersey - US).

This PoC was focused on demonstrating the capabilities and benefits of utilizing a common Information Model for multi-vendor control of wireless network elements through open management interfaces, as defined in the Wireless Transport Project of ONF Optical Transport Working Group (OTWG) and documented in the technical report TR-532 [3].

The PoC included wide participation from the wireless transport industry including operator representatives, microwave equipment vendors, integrators and applications providers. It follows the second PoC organized in April 2016, where only partial microwave management model was implemented and demonstrated, moreover additional use cases have been shown.

The following new use cases were implemented and were the subject of this PoC for the purpose of demonstrating the aforementioned applications and presented in this whitepaper:

- Closed Loop automation
- Test automation
- Spectrum management

Some applications developed for the second PoC have been enhanced to support the complete WT model:

- Connection and configuration of new microwave devices
- Detection of aberrances (‘Compare’ application)
- Receiving, displaying and storing of alarm and event information

A standard OpenDaylight (ODL) version was used as SDN controller. Mediators were used for translating the information model to vendor specific configurations.

All vendors implemented the model and completed all the test cases successfully demonstrating the viability of the concept for using a common information model for configuring and management of wireless network elements using open management interfaces.

1 Introduction

The third Wireless Transport Proof of Concept (PoC) took place October 24th through the 28th and was hosted by AT&T at Rutgers University WINLAB laboratory in New Brunswick, New Jersey.

The PoC was supported by representatives from the wireless transport eco system including operators, equipment vendors, integrators and applications providers.

The following equipment vendors participated in the PoC with their microwave or millimetre wave equipment:

- Ceragon (15GHz)
- Ericsson (23Ghz & 80GHz)
- NEC (80GHz)
- Nokia (6GHz & 5.8Ghz)
- SIAE (80GHz)
- ZTE (11Ghz)

The following Integrators and Application Providers provided buildings block and applications:

- Brocade
- Frinx
- HCL
- Highstreet Technologies
- Wipro

Content and organizational support for the PoC was provided by the following Operators:

- AT&T (event host as well)
- Deutsche Telekom
- Telefónica

The third wireless transport PoC was an evolution of the second one, completing the implementation of a common information model (developed in ONF by the Wireless Transport Project) and applying it over a variety of wireless link types and architectures. In the proof of concept the vendors supplied both all-indoor, split mount, and full outdoor configurations. The provided wireless links ranged from 5.8Ghz Unlicensed band to 80GHz high capacity, millimetre wave band. Vendors used the netconf via a mediator or native support on the network element.

The primary goal of the third wireless transport PoC was to prove the effectiveness of a common defined model in a multivendor environment. The goal was achieved through the use applications that exercised multivendor control of wireless network elements through open management interfaces (netconf based).

2 SDN Network Architecture and Configuration

2.1 Overview

The SDN architecture and configuration of the test setup in the third Wireless Transport PoC is illustrated in Figure 2 below.

The architecture identifies the main objects which are target of implementation and subsequent verification: the ODL controller (Beryllium SR2), the mediators, the Netconf interface and the northbound applications.

Network deployment included a single SDN controller, an application layer which implements specific functions that are intended to operate over the network via northbound interfaces and a wireless network layer. The wireless network layer was composed of pairs of devices and mediators and interoperates with the controller via Netconf implementing the Yang model defined by ONF in the context of Wireless Transmission project (TR-532). The mediators were devices specific and proprietary to the vendors.

The developed functional information model was incorporated into OpenDaylight via Netconf/YANG plugins able to populate the MD-SAL data store.

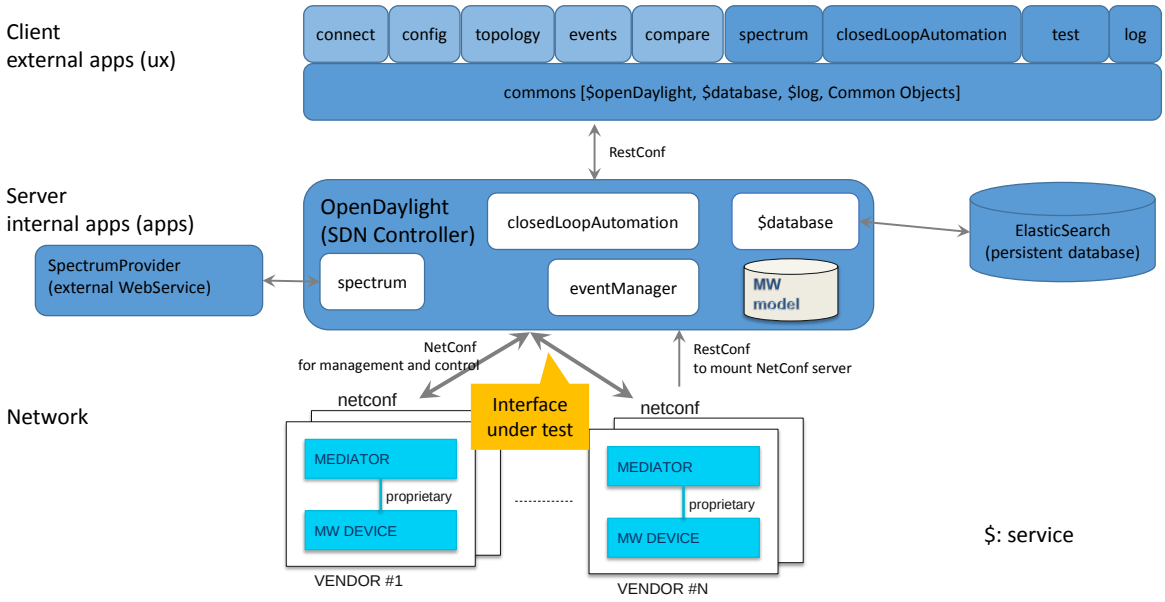


Figure 1: Overview of the SDN Architecture used in the third Wireless Transport PoC

Two ODL installations were tested: one completely open source (downloaded directly from the ODL repository), and one distributed by a partner involved in the PoC (Brocade).

In addition to the mediators required to connect the physical NEs, an NE simulator ("Default Values Mediator") has been developed completing the activity started for the second WT PoC. It behaves like a generic NE, which allows re-demonstrating the use cases and applications from the PoC without the actual need and installation of physical microwave equipment. This supports Telefonica's plans on maintaining a server running a demonstration and testing environment for future application developments.

2.2 PoC Test Network Setup

Network elements were connected via real wireless transport links with back-to-back Ethernet cabling (either electrical or optical). Traffic generator was connected to the two end points in order to verify for each use case the status of the traffic and the possible errors or packet losses due to specific commands.

From the control point of view, SDN Controller was instantiated over a virtual machine on Linux server, while 'network element mediators' were instantiated in virtual machine running either on specific hardware (e.g. PC) or in the same server hosting the SDN Controller.

Control channels for each node were out-of-band using local LAN connectivity.

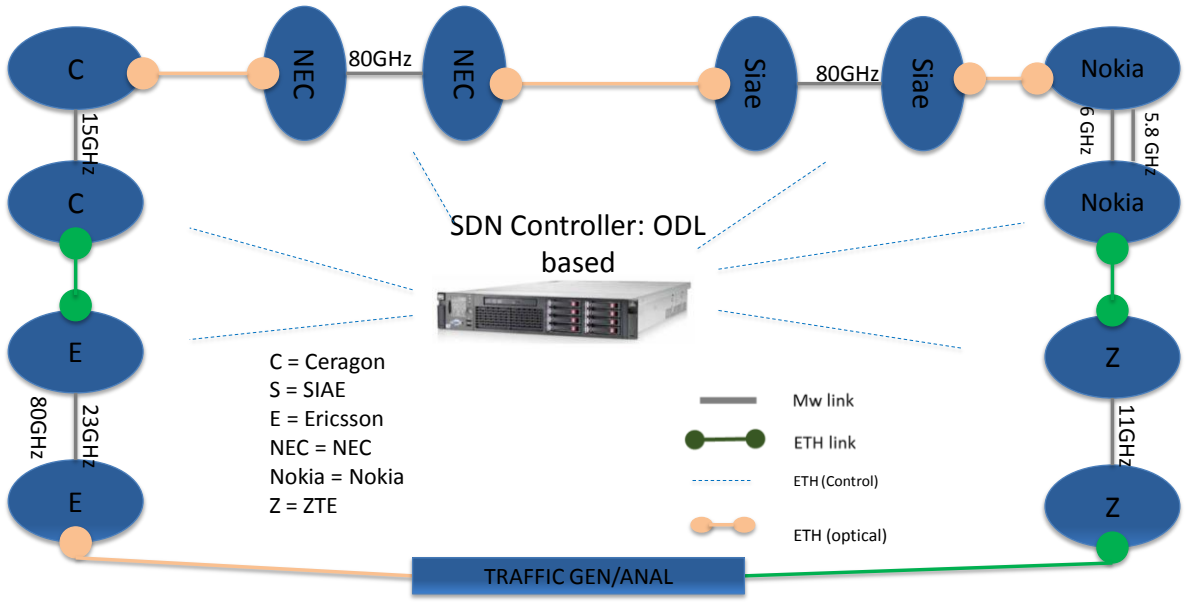


Figure 2: Test bench for the third PoC

Test bench set-up was available some weeks before the PoC execution, allowing a faster integration of the different SW components.

3 Use Cases and Applications

For the third PoC of Wireless transport SDN, some use cases from the second PoC has been refined based on the latest Microwave information model [3]. In addition, new uses cases have been developed and implemented for this third PoC in order demonstrate new wireless transport SDN applications and capabilities illustrating planning and discovery, dynamic view in real time, configuration, discrepancy monitoring and detection, and event handing.

Description of the following enhanced use cases from second PoC as used in this third PoC is detailed in [4]:

- Connection and configuration of new microwave devices
- Detection of aberrances ('Compare' application)
- Receiving, displaying and storing of alarm and event information

The following are the three added uses cases which have been developed and implemented for this third PoC:

- Closed Loop automation
- Test automation
- Spectrum management

Brief description of newly added use cases is provided in the subsections below.

Closed Loop automation
The closed Loop automation use case is developed and implemented in order to demonstrate the capability of the SDN controller to perform automated management functionalities on the network and its elements without the intervention of a human operator. For this purpose, the use case implements a simple task for which the SDN controller (OpenDaylight) executes in response to trigger events.

Three trigger events are thought for this application; external application trigger, timer based trigger or an internal network trigger (either by the SDN controller or an Network elements). The task assigned for this use case is a change of the AirinterfaceName stored in the target NE nodes with a noticeable timestamp indicating the time for which the task is executed.

The external trigger is emulated by applying a push button the GUI demonstration application implemented specifically for this PoC.

The internal trigger is emulated by implementing an application which subscribe to the SDN controller for receiving some specific notification events. In this particular demonstration, the application is registered to receive notifications when a new NetConf server with the capability “MicrowaveModel-ObjectClasses-AirInterface” is mounted into the SDN network. At the receiver of such notification the task is executed accordingly at the target NE.

Test automation
The test automation use case demonstrates an application in which the SDN controller can be used to run tests on the network by executing a set of desired test commends pre-programmed in a test script file and the test results stored in a corresponding logfile for post processing and analysis of the test results. In the PoC, a test script is written to test of a NE in the network under the control of the SDN controller for compliance to the implementation of the ONF wireless IM according to [3].

Spectrum management
The spectrum management use case is used to demonstrate the SDN controller capability in managing the spectrum allocated in its network dynamically. The main purpose of this spectrum management is to reconfigure the allocated frequency channels for each link in the network in order to mitigate high interference conditions by changing operating frequency channels to another with acceptable lower interference level. Change of frequency channel could also be required in response to an external entity (e.g. regulator agency spectrum management and allocation server or database) to evacuate certain frequency channels for other operation. Spectrum management use case is applicable to microwave network operating in unlicensed spectrum, in self-managed block assigned spectrum or in sharing spectrum such as the CBRS 3.5 GHz band in US.

The application implementing the spectrum management use case checks the current assigned and configured frequencies and polarization for each microwave/millimeterwave link and its associated NEs. The application also obtain and maintain information of the NE names and the AirInterface.radioSignalIds under its network control. The application can also access information from a SpectrumProviderService about the expected or planned frequencies. This information can then be used by the application to

compare the configured frequencies with the planned Frequencies for each link – in case of a mismatch the application configures the frequencies in the NEs accordingly. Another use scenario for the spectrum management application is if a link (or groups of links) experience high level of interference, the application will reconfigure the allocation of radio channels across the affected links to a newer clean channel that the radio NEs are capable of. The decision and the configuration are all logged in a persistent database. In this PoC the interference conditions is emulated by a push button in the GUI of the spectrum management application.

4 Test Results

Six wireless transport vendors and five integrators and application providers participated successfully in the test. The same south bound interface (Netconf protocol and YANG data models) was used between the OpenDaylight controller and the mediators of all vendors. The interfaces between mediators and network elements were proprietary interfaces which were not considered for testing in this PoC.

The use cases explained in section 3 were successfully tested by performing the following steps for the network elements of each vendor.

After having been started, the mediators announced their network elements to the Controller. The network elements were displayed as “Connected” at the GUI of the Connect application. This application also allowed network elements to be manually connected or disconnected.

The Comparison application showed the planned and actual values of all microwave attributes and highlighted the discrepancies. For the purpose of the PoC discrepancies were provoked by giving “wrong” planned values.

The configuration of the network elements was then manually modified by using the Configuration application. As an example the transmitter of a radio link was switched off. This caused the traffic flow through the chain of network elements to be disrupted (visible at the GUI of the traffic analyzer). The traffic was restored as soon as the transmitter was switched on again.

The configuration changes also triggered the network elements to report events which were forwarded as Netconf notifications to the Controller. The event handling application was used to display these notifications.

Both the closed loop automation application and the spectrum management application demonstrated the overall network view of SDN applications. The closed loop automation application used a timer (simulating the triggering condition) to periodically change the names of all air interfaces in the PoC network. The new values contained the current time stamp so that the modification could be easily monitored.

For the spectrum management application an interference situation was simulated by pressing a button at the GUI. As a consequence the application changed the frequencies of all radio links in the PoC network to new values. The resulting temporary disruption and later restoration of the traffic was again monitored with the traffic analyzer.

The test automation application verified that each mediator complied with the ONF Wireless Transport information model.

5 Conclusions

Operators
The Third PoC has proved to be a significant step towards the releasing of a standard of a common and generic information Model for SDN-enabled Wireless Transport environments, simplifying the operations and control of these network elements, and facilitating the integration of distinct multi-vendor solutions under a common and single control framework. It showed that the information model, effectively covers the different aspects of operating and controlling microwave network element in a SDN network.

It has been demonstrated a unified control and management of six wireless transport vendor products (Ceragon, Ericsson, NEC, Nokia, SIAE, ZTE) implementing the same Information Model (being defined in the context of ONF by the Wireless Transport project, with the participation of those major manufacturers), in an open environment with an OpenDaylight based controller.

It also has demonstrated that this common model can be used to manage and control legacy microwave equipment that currently does not support this information model. That can be done by using an external mediator.

Vendors
The PoC has proved that it is feasible to manage multi-vendor Network Element using the Yang Data models, has been developed by the ONF Wireless Transport project and derive from the above mentioned information model. Moreover, it has been shown that the model effectively generalizes the different aspects of Microwave/Millimeter-wave operating and controlling mechanism, and can be used to operate and control vendor specific Network Elements regardless of their unique implantations.

The PoC has demonstrated how the cooperation in a multi-vendor microwave network could be managed by a single SDN controller. This offers the possibility to operators for developing their own applications on top of an open and modern REST northbound interface, independently from the vendors providing the microwave infrastructure.

Integrators and Application Providers
The demonstrated applications has shown that all vendors implemented the full Netconf common southbound management interface based on a YANG model which includes events handling and that the relevant events are generated as a result of action made by the application.

It also showed that the yang model facilitates the development of reach and variety applications that can control the different aspects of microwave network and to collect a statuses and Performance Monitoring data.

All the applications have been successfully executed over the different vendors equipment. Moreover, some of the applications approached (Via the SDN controller) the different Networks Elements on the same run. It has strengthened the assumption that the information model enables an application that should not be aware of the differences between the Networks Elements.

The choice of OpenDayLight enables the applications to use the inherent North Bound Interface the OpenDayLight generates automatically when a YANG module is introduce to it. It exposes a Restful NBI that derives from the YANG Model. This capability enables the development of application without investing a development effort on the NBI. However, to support other SDN Controllers a development of standard NBI should be consider

6 Acknowledgments

Thank you to AT&T, in particular to Tracy Van Brakle, major sponsor of this event, and to Winlab that hosted the demo. Thank you to Thorsten Heinze (Telefonica) for its fundamental contribution to the definition of the model and of TR-532 specification and to Martin Skorupski (Highstreet Technologies) for coordinating the application development. Thank you to all the team participating to the event (AT&T, Telefonica, DTAG, WINLAB, Ceragon, Ericsson, NEC, Nokia, SIAE, ZTE, Highstreet Technologies, FRINX, Wipro, HCL, Spirent, Brocade).

Thank you to Luis Contreras (Telefonica), Dimitrios Siomos (DT), Petr Jurcik (DT), Thomas Kessler (DT) for providing continuous support to the activities of the team.

Finally, thank you to the contributors and reviewers to the various sections of this White Paper, namely:

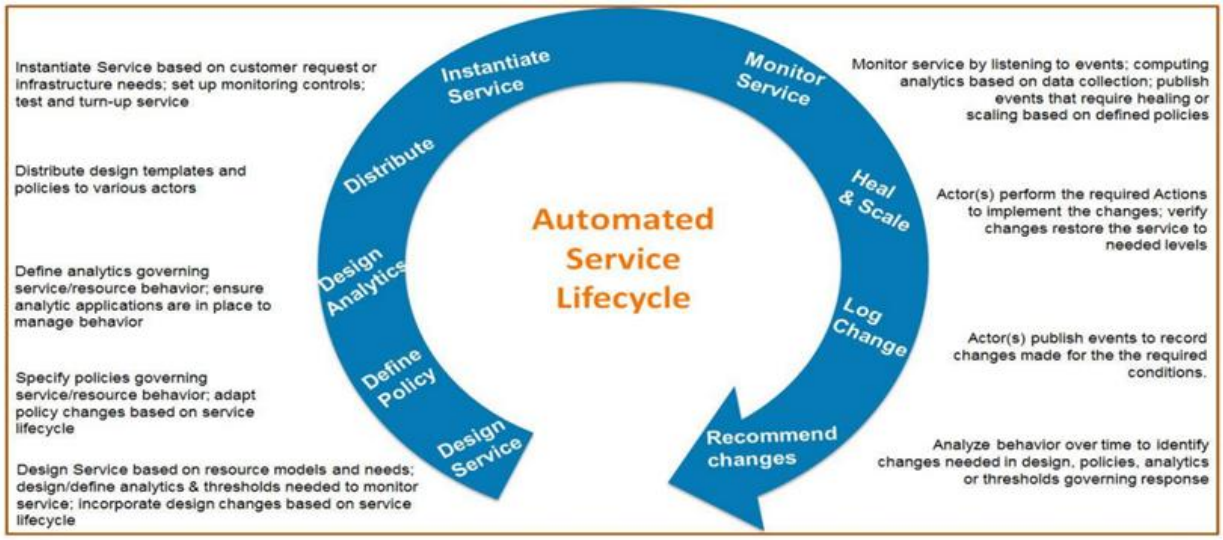
- Alexandru Stancu – Ceragon
- James Ries – Nokia
- Michael Binder – Ericsson
- Yossi Victor – Ceragon
- Martin Skuropski – Highstreet Technology
- Nader Zein – NEC
- Paolo Spallaccini – HCL
- Weihua Liu – ZTE
- Lukáš Beleš - FRINX
- Giorgio Cazzaniga – SM Optics (SIAE Microelettronica group)
- Petr Jurcik - DT
- Yang Yongang - ZTE

7 Contributions

7.1 Closed Loop Automation

Context

The Closed Loop Automation (CLA) is a pattern, defined by “ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper”, which provides the necessary automation for proactive response to network and service conditions without human intervention. A high-level schematic of the “Closed-loop automation” and the various phases within the service lifecycle using the automation is depicted in the following picture:



We prove that OpenDaylight can work automatically and independently with NetConf devices without human intervention, over point to point Wireless networks. It can automatically read and write information from/to NetConf devices. Additionally, when something is changed on the device by another person or by another process, this is noted and processed.

Implementation

In this PoC, we implemented a module which reads and writes data from/to NetConf devices. We process only NetConf devices which have the MicrowaveModel-ObjectClasses-AirInterface capability. The app reads and changes the AirInterfaceName of package MicrowaveModel-ObjectClasses-AirInterface. We simply change a name according to actually timestamp.

This action will be executed when three different events occurs:

- **External application**
  In the PoC the external trigger is represented by an “Execute Now” button on the DLUX GUI.

- **Timer**
  The timer can be switched “on” and “off” and in addition the time frequency for automatic execution can be configured by the user. The default time frequency is 5 seconds. These settings are stored in the persistently in ODL’s config datastore.

- **Notification from SDN Controller or the network elements**
  The application subscribes for notification from ODL. Such notification can be notification from

ODL itself or forwarded notification form the network. The application listens to ODL topology changes. Each time, when a new NetConf server with the capability “MicrowaveModel-ObjectClasses-AirInterface” is mounted, then the closed loop automation function is executed.

The application is developed as Karaf OSGi in Java 1.8. The frontend is based on DLUX which is mainly in written in Angular.

Resource

Additional information about the CLA, how to install it, and the source files, are available at:

https://github.com/OpenNetworkingFoundation/CENTENNIAL/tree/master/03-WTP-PoC/code/apps/closedLoopAutomation

ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper

http://about.att.com/content/dam/snrdocs/ecomp.pdf (see section 5)

7.2 Spectrum management

Context

The Spectrum management application is designed for simulating a specified scenario, which shows what COULD SDN provide when the interference occurred in microwave transportation, and what COULD SDN achieve when facing devices from multi vendors at the same time.

In this simulation, the Spectrum management is using the different “next” available frequency for every device individually according to the “running” values retrieved from a general agency which simulate the FCC database to make sure all devices can move to the next frequency and maintain the availability of the transportation.

Implementation

The Spectrum Management consists of three parts, each part can be deployed individually anywhere.

1. The FCC database simulation was implemented by a Model-Oriented-System so called MOS which was created by ZTE, which also along with a Command-line interface and web based jsonrpc interface that bases on model and primitive as ‘get, set, add, del, action’. Provides all available frequencies for every AirInterface on every device.

2. The Spectrum application coded in JAVA, provides an api in restful style named ‘execute’ which would read ‘next’ availiable frequency for every AirInterface on every device of all vendors acoording to the current value. After that the spectrum application will send the new value to each device one by one.

   The ‘next’ available frequency is decided by current value and the step, the formula looks like [next value] = [current value] + [step]

3. The UI provides a huge button with ‘alarm’ on it and a table of all AirInterfaces infomation of all online devices, includes planned value and current value of tx/rx frequencies.

   When the button is punched, it shows an ‘alarm’ icon which represents the ‘interference’ alarm has been triggered, and send the ‘execute’ command to the spectrum application, after all work done the button’s icon changes back to normal.

During the whole process, the traffic would jitter for a moment till all done.

Resource

Additional information about the Spectrum management app is available at https://github.com/OpenNetworkingFoundation/CENTENNIAL/tree/master/03-WTP-PoC/code/apps/spectrum

Additional information about the general agency so called MOS is available at https://github.com/olinchy/mos/tree/master/develop/source-code

7.3 Test Automation

Context

Following the increasing demand for verification tools which raised within the ONF Wireless-Transport Project, the working group targeted the development of an application conceived within the third Proof-of-Concepts architecture constraints and capable of providing the following automatized verification functionalities:

- Detecting and discovering the YANG model exported by the netconf server implemented on the NE mediation layer

- Parsing the whole set of attributes in the exported YANG models
- Recognizing the basic attributes features: configurability, type, defaulted values
- Being able to retrieve attribute values via northbound i/f (restconf)
    - The set of attribute values to be retrieved should be made selectable via input files
- Being able to modify attribute values via northbound i/f (restconf)
    - The set of attribute values to be modified should be made selectable via input files
- Reporting the outcome of all the performed operation in formatted logfiles/summaries, including execution timestamps

A dedicated Use Case has subsequently been defined for the Third WTP PoC.

Slightly extending the intended scope and in the perspective of future evolutions, The PoC3 Test Automation application targets a broader goals range, including:

1. The automatized verification of status, capabilities and configurability of the equipment
2. Basic automatized sanity checks of the equipment
3. Basic automatized negative cases execution

For the 3rd PoC purposes a test automation engine has been provided (TA engine); leveraging TA engine, PoC contributors can autonomously develop specific test cases in order to come to a custom realization of the Test Automation Suite. A basic front-end interface has also been provided in order to improve the user experience with the TA engine.

The TA engine capabilities extend verification benefits beyond the specific PoC3 functionality and may represent a “concept tool” for the validation of the WTP YANG model candidate for standardization, which is still under development.

A test automation framework should also, in principle, be aimed at providing early responses during the joint development integration phases for the whole PoC chain. Such PoC chain can therefore be considered the real DUT: it spans across the YANG models, the equipment mediation layer until the ODL architecture, with special reference to the newly introduced features and bundles.

In its present state of development, the test automation engine/suite is primarily consuming the restconf northbound i/f. It cannot be used for verifying the other developed apps interfacing northbound the ODL.

The test automation application might also be used to validate development contributions coming from diverse sources. Yet, marking a difference with respect to the 2nd WTP PoC, the automation of the environment setup - which would have facilitated the construction of a “continuous integration” test environment - has not been implemented, preferring to focus, in the timeframe available, on delivering the specific functionality requested in the use case. Such feature however, alongside many others, is definitely a target for a possible follow-up of the present work

Implementation

The Test Automation Engine/Suite is composed of two main parts:

- Back-End test automation engine
- Application Front-End interface

The **Back-End engine** has been developed via JavaScript code which is run atop Node.js runtime environment; the scripts are leveraging on several different technology packages and libraries, such as chai, mocha, supertest, synchronize, and others.

Three scripts have been developed for the Back-End engine, thus providing the described test automation features. Each script implements a different functionality, as described below:

1. *01-standalone-YANG-parser.js*

The script provides a flexible standalone functionality (algorithm) for YANG models discovery/attribute parsing. The script is featuring the whole parsing algorithm and can be applied to standalone YANG models which are defined into specific files. The necessary script configuration is retrieved from a separate input file.

2. *02-netconfserver-YANG-parser.js*

   The script provides NE inquiring and YANG models discovery/attribute parsing capabilities, alongside flexible attribute restconf URL construction and flexible attribute data retrieval ("GET"). The necessary script configuration is retrieved from a separate input file.

3. *03-netconfserver-YANG.js*

   The script provides NE inquiring and YANG models discovery/attribute parsing capabilities, alongside flexible attribute restconf URL construction, flexible attribute data retrieval and attribute data modification, in accord with a predetermined (hardcoded) pattern ("GET" + "SET"). The necessary script configuration is retrieved from a separate input file.

All the scripts accept input files and produce detailed logs (see Resource paragraph for more info).

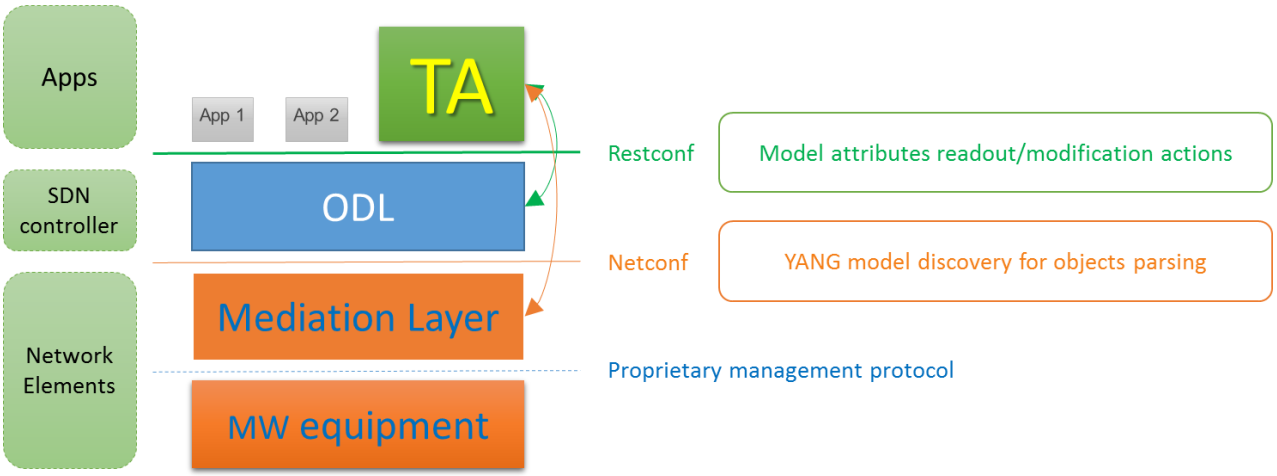
A **Front-End application interface** has been also developed. It delivers a simple GUI which gives to the user the possibility of:

1. Selecting the specific script to be run
2. Selecting the target Network Element
   a. The NE must belong to a given topology that must be statically defined in a separate file.

The GUI has been developed as a web server, specifying as HTTP server the port '3000' on the ODL; the web server can be started as a Node.js application (for this purpose it has been also provided an *npm start* script).

The GUI shows a test summary report and a sliding window reporting the test result logs for each script that is run on the specified NE.

In the figure below, a sketched depiction of the TA application deployment over the PoC3 architecture is given:



Test Automation application deployment

Resource

Additional information about the Test Automation application, alongside the developed code and more detailed instructions about installation and usage can be found at:
https://github.com/OpenNetworkingFoundation/CENTENNIAL/tree/master/03-WTP-PoC/test

7.4 Default Value Mediator

Context
The Default Value Mediator version 3 (DVM v03) is an enhancement of the DVM used in the second MW PoC.Its purpose was the same as the DVM, meaning providing the application developers the possibility of developing and testing their SDN applications without the need of connecting to a real mediator that communicates with an actual wireless transport device. DVM v03 offers the same interface and also the same behavior as any conventional mediator and represents a very useful tool in the context of SDN applications development.

Implementation
The Default Value Mediator version 3 is a Netconf server coded in C, based on the OpenYuma framework, implementing the YANG models used in the third MW PoC. It is an enhancement of the DVM used in the second MW PoC. The server provides default values for all the attributes defined the YANG models comprised in the following packages: CoreModel-CoreNetworkModule-ObjectClasses, MicrowaveModel-Notifications, MicrowaveModel-ObjectClasses-AirInterface, MicrowaveModel-ObjectClasses-EthernetContainer and MicrowaveModel-ObjectClasses-PureEthernetStructure. The values are available for getting and setting immediately after the server starts. Each of the YANG model is converted in a Netconf server module, implemented as a C shared object and can be dynamically loaded into the server, even at runtime.

All values for the attributes of the DVM v03 are loaded from an external XML configuration file. This provides better flexibility than the previous DVM. For any change in the default value of an attribute, the server does not need a recompilation. A Mediator restart is sufficient for the new default values to be reflected. This also constitutes a disadvantage. If an attribute is set in the server through a Netconf edit-config operation, its value will not be kept upon server restart, because the setting is not applied in the XML configuration file, it only affects the runtime values. Altogether, this gives SDN application developers an easy to use tool for simulating entire wireless transport networks with minimum effort.

DVM v03 is also capable of generating up to two Netconf notifications, at an interval specified in the configuration file. The notifications can be of type attributeValueChangedNotificaion or problemNotification and their contents are also customizable in the aforementioned file.

Another feature implemented in this Mediator is that the reading from the XML file is done dynamically. This means that if we change the values of some status attributes from the XML file between two consecutive Netconf get operations, we can get two different values for a status attribute. This makes the DVM v03 resemble more a real mediator which reads status attributes directly from the network element.

Resources
Additional information about the DVM v03, about how to install and use it, and the source files are available at https://github.com/OpenNetworkingFoundation/CENTENNIAL/tree/master/03-WTP-PoC/code/Default_Values_Mediator

7.5 Persistent database

Context

Typically an SDN controller stores data from the network in a memory database. However, applications on top of the SDN controller may require access to a persistent database for information which are not available in the network itself. Such persistent database could store information of passive components of the network (e.g. cabling, antennas) or administrative data (e.g. planning states, site owner). In addition a persistent database could store any kind of logs from applications, the SDN controller and network elements.

Implementation

The OpenDaylight project uses Apache Karaf as open-source OSGi framework. It is beneficial when a database also acts as OSGi bundle within Karaf. Installation and maintenance would be similar to all other OSGi bundles used by OpenDaylight. Apache Karaf organization itself provides a monitoring solution called Apache Karaf Decanter. Decanter uses as database the open-source project ElasticSearch. It provides a distributed, multitenant-capable full-text search engine with a well-documented REST interface and schema-free JSON documents.

Even the OpenDaylight project does not use ElasticSearch, its Karaf feature is distributed together with the OpenDaylight distribution. This fact makes installation and setup of persistent database easy and efficient.

Resources

Installation of the persistent database in ODL karaf:
https://github.com/OpenNetworkingFoundation/CENTENNIAL/tree/master/03-WTP-PoC/code/apps/persistentDatabase

Additional information about Apache Karaf Decanter at
https://karaf.apache.org/manual/decanter/latest-1/

Additional information about ElasticSearch at
https://www.elastic.co/products/elasticsearch

8 References

- [1] Netconf [RFC 6241]
- [2] YANG [RFC 6020]
- [3] TR-532 – Microwave Information Model
- [4] Wireless Transport SDN Proof of Concept 2 Detailed Report (June 6, 2016) - https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Wireless_Transport_SDN_PoC_White_Paper.pdf
- [5] Wireless Transport SDN Proof of Concept White Paper (2015-10-09) - https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/ONF_Microwave_SDN_PoC_White_Paper%20v1.0.pdf