Author Details

Mani Prashanth Varma Manthena
M.Sc. in Electrical Engineering - Telecommunications Track (2012 - 2014)
Faculty of EEMCS
TU Delft
Graduate Intern (2013 - 2014)
Service Enabling and Management Department
TNO - Delft
Email: me@prashanthvarma.com, M.P.V.Manthena@student.tudelft.nl
Ph. No.: +31684401806

NaaS Platform

NaaS Platform is a software based application platform, which is written and developed as a step towards realizing/enabling Network-as-a-Service (NaaS) cloud model for service provider networks. This platform was developed as a Proof of Concept (PoC) for my M.Sc. thesis/graduation project at TU Delft and TNO. My M.Sc. thesis/graduation project involved proposing and validating an evolutionary approach to NaaS architecture with SDN and NFV for service provider networks. In Figure 1, my PoC NaaS architecture is shown.

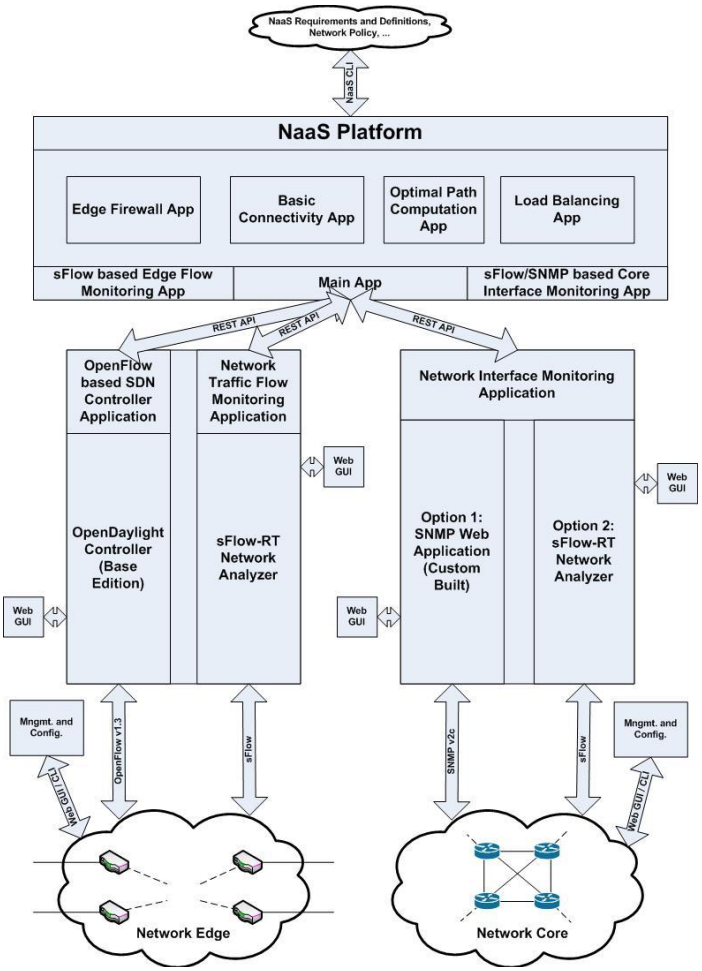

Figure 1. PoC NaaS Architecture

PoC NaaS Architecture Description

Note: This PoC is implemented mainly in UNIX/LINUX (i.e. Ubuntu) based systems/VMs. However, it can also be implemented in other OS environments without any changes to the PoC - Source Code. Further, all the required applications and dependencies for the PoC can either be implemented in a single system/VM (i.e. refer PoC - VM) or in multiple systems/VMs (i.e. in the above PoC NaaS architecture, each block along with its external interfaces represents a single system/VM) for scalability reasons.

PoC NaaS Architecture - NaaS Platform and Virtualized Network Functions:

- Basic Connectivity
- Optimal Path Computation
- Load Balancing
- Edge Firewall

PoC NaaS Architecture - OpenFlow based SDN Controller Application:

- OpenDaylight Controller (Base Edition)

PoC NaaS Architecture - Network Edge Traffic Flow Monitoring Application:

- sFlow-RT Network Analyzer

PoC NaaS Architecture - Network Core Interface Monitoring Application:

- Option 1 (e.g. Testbed Setup A): Custom built SNMP Web Application
- Option 2 (e.g. Testbed Setup B): sFlow-RT Network Analyzer

PoC - Testbed Setups:

- Testbed Setup A: Proposed solution with legacy switches at the network core
- Testbed Setup B: Proposed solution with OpenFlow/Open vSwitch enabled switches at the network core

Proof of Concept (PoC) Resources:

- For my Proof of Concept (PoC) - Architecture, go to the folder PoC - NaaS Architecture
- For my Proof of Concept (PoC) - Source Code, go to the folder PoC - Source Code
- For my Proof of Concept (PoC) - Testbed Setups, go to the folder PoC - Testbed Setups
- For my Proof of Concept (PoC) - VM, go to the folder PoC - VM
- For my Proof of Concept (PoC) - References, go to the folder PoC - References

NaaS Platform Contents

- Main Application
- sFlow based Edge Flow Monitoring Application
- SNMP/sFlow based Core Interface Monitoring Application
- Basic Connectivity Application
- Optimal Path Computation Application
- Load Balancing Application
- Edge Firewall Application
- Configuration and log files

Note: As it can be clearly seen from the NaaS Platform's source code, this application is written for my PoC - Testbed Setups. However, If you want to use this application for other testbed setups (i.e. custom testbed setups), you need to make some changes to the configuration files in the source code of this platform. Information regarding these required changes to the source code configuration files is mentioned in the section "Custom Testbed Setups" at the end of this document.

Getting Started

Note: This software platform is completely written in Python 2.7, due to its simplicity, interoperability, support, and platform agnostic nature. Thus, the NaaS Platform will run in all the systems/VMs with Python version 2.x.

Step 1) Installing and launching the required applications and dependencies

A) Installing and launching the OpenDaylight Controller (Base Edition) application (OpenFlow based SDN Controller Application):

- Before starting with the installation of the OpenDaylight Controller application, check and verify whether your system's Java is properly installed and its environment variables are also properly configured.

- Download the latest OpenDaylight Controller base edition from the OpenDaylight CrossProject: Integration Group: Controller Artifacts wiki.

- Hydrogen stable distribution artifacts (till September, 2014) of OpenDaylight Controller (Base Edition) were tested and implemented successfully for this PoC.

- In essence, any of the stable distribution artifacts of OpenDaylight Controller should work with this NaaS Platform and/or architecture, as its only requirement is that of support for OpenFlow version 1.3 and above by the OpenDaylight controller.

- Before launching the OpenDaylight Controller application, configure the network edge switches with the details of the OpenFlow based SDN Controller Application (i.e. OpenDaylight Controller application). This configuration requires the following two values:

  Controller IP Address = IP Address of the system hosting the OpenDaylight Controller application

  Port Number = 6633 (OpenFlow protocol port number)

- After downloading and un-zipping the latest OpenDaylight Controller artifact, open a new terminal window and go (i.e. "cd") to the controller's main directory, where you need to enter "./run.sh" (i.e. for UNIX/LINUX systems) or "./run.bat" (i.e. for windows systems) with administrator privileges to launch the OpenDaylight Controller application.

  $ sudo ./run.sh -Xmx1024m -of13

  Note: The "-of13" command launches the OpenDaylight Controller application with OpenFlow protocol version 1.3 support. Additionally, the "-Xmx1024m" command is optionally used in case of a MAVEN error "java.lang.OutOfMemoryError: PermGen space:" during the startup of the OpenDaylight Controller application.

- After launching the OpenDaylight controller, verify the controller's operational status by navigating to its web GUI.

  Default URL:
  http://*controller-ip***:8080**

  Default Login Credentials:
  User: **admin,** Password: **admin**

  Note: "*controller-ip*" in the above URL is the IP address of the system hosting the OpenDaylight Controller application, this URL is also the base URL for the OpenDaylight Controller's REST API. For further information and references relating to the OpenDaylight Controller application, go to the folder PoC - References\OpenDaylight Project and Controller

B) Installing and launching the sFlow-RT Network Analyzer Application (sFlow based Network Edge Traffic Flow Monitoring Application):

- sFlow-RT network analyzer application talks sFlow protocol with the underlying network devices for network monitoring, sFlow is an industry standard which stands for "sampled flow". In the network devices, sFlow samples incoming packets as per the pre-configured sampling rate and sends them as sFlow datagrams to the collector (i.e. sFlow-RT network analyzer) along with network interface counters which are sent as per the pre-configured polling interval.

- Before launching the sFlow-RT network analyzer application, configure the network edge switches with the details of the sFlow datagram collector application (i.e. sFlow-RT network analyzer application). This configuration requires the following two values:

  Collector IP Address = IP Address of the system hosting the sFlow-RT network analyzer application

  Port Number = 6343 (sFlow protocol port number)

- For installing and launching the sFlow-RT network analyzer application, open a new terminal window and enter the following commands:

  $ sudo wget http://www.inmon.com/products/sFlow-RT/sflow-rt.tar.gz

  $ tar -xvzf sflow-rt.tar.gz

  $ cd sflow-rt

  $ sudo ./start.sh

- After launching the sFlow-RT network analyzer, verify the application's operational status by navigating to its web GUI.

  Default URL:
  http://*analyzer-ip***:8008**

  Note: "*analyzer-ip*" in the above URL is the IP address of the system hosting the sFlow-RT network analyzer application, this URL is also the base URL for the sFlow-RT network analyzer application's REST API. All the exposed REST APIs by the sFlow-RT network analyzer application are well documented in the

application's web GUI. For further information and references relating to the sFlow-RT network analyzer application, go to the folder PoC - References\sFlow based Network Monitoring

C) Installing and launching the sFlow/SNMP based Network Core Interface Monitoring Application:

- Here you have two options, option 1 is using a custom built SNMP based network interface monitoring web application and option 2 is using sFlow-RT network analyzer application for network core interface monitoring.

- In essence, these network interface monitoring applications must expose interface failure and high bandwidth utilization events with configurable utilization thresholds via REST API to the NaaS platform.

- sFlow based network interface monitoring applications are preferred over their SNMP counterparts, as sFlow involves polling interface counters on pre-configured polling intervals without any pull requests (i.e. no overheads and delays). Further, sFlow-RT network analyzer application has a well documented REST API which already supports and exposes interface failure and high bandwidth utilization events with configurable utilization thresholds.

- However, most of the legacy network devices do not support sFlow, in which case simple and custom built SNMP based network interface monitoring applications must be used. As we know, SNMP is the de facto network management and monitoring standard in almost all of the legacy network devices.

- As part of the PoC, I have built a custom SNMP based network interface monitoring web application as per the above mentioned requirements. Source code for the custom built SNMP web application is available in the folder PoC - Source Code\SNMP-Web-Application.

- **Option 1:** For networks similar to Testbed Setup A (i.e. proposed solution with legacy switches at the network core), use custom built SNMP based network interface monitoring web application.
  In this option, for network interface monitoring, you can install and launch my custom built SNMP Web Application (i.e. PoC - Source Code\SNMP-Web-Application) or built your own version of the custom SNMP web application as per the above mentioned requirements.

  Installing and Launching my custom SNMP Web Application: The procedure for installing and launching the SNMP Web Application is available in the README file of its source code.

  After launching the SNMP Web Application, verify the application's operational status by navigating to its web GUI.

  Default URL:
  http://*snmp-ip***:8090**

  Note: "*snmp-ip*" in above URL is the IP address of the system hosting the SNMP Web Application, this URL is also the base URL for the SNMP Web Application's REST API. All the exposed REST APIs by the SNMP Web Application are well documented in the application's web GUI.

- **Option 2:** For networks similar to Testbed Setup B (i.e. proposed solution with OpenFlow/Open vSwitch enabled switches at the network core), use sFlow-RT network analyzer as the network interface monitoring application.

  In this option, for network interface monitoring, you can configure the network core switches to talk with the already installed and launched sFlow-RT network analyzer (i.e. Step 1 - B). However, if you want to run the network interface monitoring application (i.e. sFlow-RT network analyzer application) in a different

system/VM for scalability reasons as in the PoC - NaaS Architecture, install and launch a new instance of the sFlow-RT network analyzer in that system/VM as per the procedure mentioned in Step 1 - B.

After launching the sFlow-RT network analyzer, verify the application's operational status by navigating to its web GUI.

Default URL:
http://*analyzer-ip***:8008**

Note: "*analyzer-ip*" in the above URL is the IP address of the system hosting this instance of sFlow-RT network analyzer application (i.e. for network core interface monitoring), this URL is also the base URL for the sFlow-RT network analyzer application's REST API. All the exposed REST APIs by the sFlow-RT network analyzer application are well documented in the application's web GUI. For further information and references relating to the sFlow-RT network analyzer application, go to the folder PoC - References\sFlow based Network Monitoring

D) Installing the required dependencies for the NaaS Platform

- Requires Python version 2.x, this application was tested and implemented with Python version 2.7.

  Installing Python pip and other dependencies in Ubuntu, Python pip is a tool for installing and managing python packages. Ubuntu terminal command for installing Python pip tool:

  $ sudo apt-get install python-pip python-dev build-essential

  $ sudo pip install --upgrade pip

  $ sudo pip install --upgrade virtualenv

- As this application involves simultaneously executing several python scripts at the start of the application, the system/VM which hosts this application mush have the xterm terminal emulator installed in it. Ubuntu terminal command for installing xterm terminal emulator:

  $ sudo apt-get install xterm

- Requires a Python package called Requests for talking with REST APIs (i.e. opening URLs). Requests is a simple and elegant HTTP library for Python. Ubuntu terminal command for installing Requests package:

  $ sudo pip install requests

- Requires a Python package called NetworkX for optimal path computations using graph theory. NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Ubuntu terminal command for installing NetworkX package:

  $ sudo pip install networkx

- Requires a Python packages called NumPy & SciPy for plotting optimal paths computed by the NaaS Platform. Ubuntu terminal command for installing NumPy & SciPy packages:

  $ sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose

Step 2) Installing and launching the NaaS Platform

- After copying the NaaS Platform source code to the system/VM, open a new terminal window and go (i.e. "cd") to the platform's main directory, where you need to enter "./start.sh" with administrator privileges to launch the NaaS Platform.

  $ sudo ./start.sh

  Note: The "./start.sh" command launches all the required scripts in the source code of the NaaS Platform.

Step 3) Configuring the NaaS Platform and Architecture

- After launching the NaaS Platform, go to the running Main Application terminal window (i.e. xterm terminal) and configure the NaaS architecture as per the instructions in the terminal window.

- After the NaaS architecture is configured in the Main_App, you will be presented with a command prompt to execute manual NaaS operations. Enter "help" or "list" or "?" in the command prompt to view all the available manual NaaS operations through the NaaS platform.

Step 4) Running the Virtualized Network Functions

- When the NaaS platform is launched, along with the Main application, a single instance of all the available virtualized network functions are opened in a new terminal (i.e. xterm terminal).

  The following applications are opened as xterm terminals at the startup of the NaaS Platform:
  1) Main Application
  2) Basic Connectivity Application
  3) Load Balancing Application
  4) Edge Firewall Application

- Go to the corresponding xterm terminal, and follow the instructions to start that virtualized network function.

- These virtualized network functions are fully configurable and flexible with a wide range of NaaS requirements and definitions, network policies, etc.

- At the start of the NaaS Platform, the preferred order of configuring and starting the virtualized network functions is Basic Connectivity Application (required), Load Balancing Application (optional), and Edge Firewall application(optional). However, these virtualized network functions are independent of each other and can be executed independently without the requirement of each other.

- Further, you can run multiple instances of a single virtualized network function without any incompatibilities and policy collisions.

  For this, you need to run the setup scripts of the corresponding virtualized network functions. Here are the commands to enter in the terminal window, once you are in the platform's source code main directory:

  1) Starting a new instance of Basic Connectivity Application (i.e. virtualized network function)

  $ python Setup_Basic_Connectivity_App

  2) Starting a new instance of Load Balancing Application (i.e. virtualized network function)

$ python Setup_Load_Balancing_App

3) Starting a new instance of Edge Firewall Application (i.e. virtualized network function)

$ python Setup_Edge_Firewall_App

Similarly, if you want to reconfigure the NaaS Platform and architecture (i.e. access the Main Application), you can run a new instance of the Main Application by running the Initialize Python script in the platform's source code. Here is the command to enter in the terminal window to start the Initialize Python script:

$ python Initialize

- Additionally, if you want to visualize optimal path computations (i.e. graphical representation of computed optimal paths), you need to un-comment few lines in the Optimal_Path_Computation_App Python script in the platform's source code. These graphs/plots will pop up after each optimal path computation by the Optimal_Path_Computation_App Python script. Further, you can find the latest instances of these graphs/plots in the Path_Computation_Plots folder of the platform's source code.

  For this, you need to un-comment (i.e. remove the # tag) all the instances of the line **"#plt.show()"** in the Optimal_Path_Computation_App Python script in the platform's source code.

Custom Testbed Setups

If you want to implement this NaaS Platform for custom testbed setups (i.e. other than the PoC - Testbed Setups), you need to make few changes to the configuration files in the source code of this platform.

- At first, as per your testbed setups, you need to edit the network topology configuration files in the platform's source code. These configuration files are in simple and readable JSON format.

- Finally, as per your configured static LSPs in the network core, you need to edit the MPLS static LSP path to label bindings configuration files in the platform's source code. These configuration files are in simple and readable JSON format.

Final Note

For further information and references relating to the SNMP Web Application, go to the folder PoC - References or contact me via my email id.