Software-Defined Networking: Guidelines for Experimentation and Validation in Large-Scale Real World Scenarios

Joao Goncalves[1], David Palma[1], Luis Cordeiro[1], Sachin Sharma[2], Didier Colle[2], Adam Carter[3], and Paulo Simoes[4]

[1] OneSource, Consultoria Informatica, Lda.
{john,palma,cordeiro}@onesource.pt
[2] Department of Information Technology (INTEC), Ghent University, iMinds
{sachin.sharma,didier.colle}@intec.ugent.be
[3] Edinburgh Parallel Computing Centre (EPCC), University of Edinburgh
adam.carter@ed.ac.uk
[4] CISUC, Department of Informatics Engineering, University of Coimbra
psimoes@dei.uc.pt

Abstract. This article thoroughly details large-scale real world experiments using Software-Defined Networking in the testbed setup. More precisely, it provides a description of the foundation technology behind these experiments, which in turn is focused around OpenFlow and on the OFELIA testbed. In this testbed preliminary experiments were performed in order to tune up settings and procedures, analysing the encountered problems and their respective solutions. A methodology consisting of five large-scale experiments is proposed in order to properly validate and improve the evaluation techniques used in OpenFlow scenarios.

Keywords: OFELIA, OpenFlow, Software-Defined Networking.

1 Introduction

The research towards the Future Internet raises many challenges that require demanding infrastructures and setups to allow their assessment and clarification. However, testing new protocols and services in existing network facilities is not typically considered as an option. Consequently many sectors are not willing to experiment with new protocols and applications that might affect their services.

Software-Defined Networking (SDN) has been acknowledged as suitable answer to provide necessary functionalities and flexibility. This is achieved by decoupling the networking decisions from the hardware enforcing them, new routing protocols or on-demand routing decisions can be easily introduced in production networks, mitigating the effects of typical changes or updates in existing networks. Moreover, the definition of an open protocol such as OpenFlow breaks the barriers of hardware dependant routing mechanisms, enabling a larger dissemination of well-structured networks and services.

In this work a detailed compendium of the preliminary experiments performed on The OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) testbed [1] is presented, that ultimately led to the definition of the five large experiments, which are set out to expose the properties and limitations of a SDN employing the OpenFlow standard [2]. More specifically, this OpenFlow network is coupled with a Quality of Service (QoS) system towards an emulated city of one million inhabitants.

In Section 2 of this document, all the tools and mechanisms used throughout the experiments are explained in detail. Followed by the definition of the relevant Key Performance Indicators (KPI) and of the proposed experimentation process which are described in Section 3, with particular attention given to the preliminary tests and the problems identified while executing them. In Section 4 an analysis of the expected results is given and finally in Section 5 the main conclusions are drawn and future steps considered.

2 Experiment Tools and Mechanisms

The OFELIA project consists of a collaborative creation of an experimental research facility, involving several testbed islands. This large-scale facility is based on OpenFlow, allowing its users and researchers a unique opportunity to dynamically configure, control and experiment the whole network infrastructure using different technologies without limitations.

The architecture followed in these experiments, sets out a goal for future OpenFlow-based networks, enabling them with the possibility to dynamically configure paths with guaranteed traffic characteristics. The separation between data and control planes followed by the SDN paradigm fits this goal, allowing additional business intelligence to be included on top of the control plane, which in turn enforces the necessary decisions on the data plane.

2.1 OFELIA Testbed Islands

The motivation for this work also resulted from previous experiments performed resorting to OFELIA. On these particular experiments, six different interconnected testbeds were used, each one with very specific characteristics. The defined scenarios were exploited so that an extensive use of these islands could provide meaningful results and insights for future large-scale experiments [1].

In particular the following OFELIA islands were and will be used in the performed tests:

- **i2Cat (Barcelona):** Providing L2 switches and optical equipment, enabling power aware OpenFlow experiments;
- **ETHZ (Zurich):** Integration within the perimeter of the campus network and also provides the possibility of an operational research track in parallel with an experimental track; available there is the provision of connections to OneLab and GENI;

- **CNIT-Cantania (Cantania):** Based on NetFPGA and Open vSwitch (OvS) technologies while focusing on Information Centric Networking;
- **Create-Net (Trento):** Citywide distributed island based on Layer 2 switches and NetFPGAs, with opt-in users via heterogeneous access technologies;
- **TUB (Berlin):** A few OpenFlow switches are integrated in the campus network;
- **iMinds (Ghent):** Central hub of the federated facility that allows for large scale simulation with at least 100 interconnected servers, with a total of 10 NetFPGA 1Gb cards.

Each of these pan European islands possess their own characteristics but are seen as a whole within these experiments, building large-scale scenarios for the performed evaluations, using the OFELIA provided facilities.

2.2 Main Components

A whole set of different components needed to be defined for further use in the desired large-scale experiments. The detailing of each component appears in the following sections.

Open vSwitch. OvS [3] is a production quality, multilayer virtual switch. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols. OvS is targeted at multi-server virtualization deployments, a landscape for which the current stack is not well suited. These environments are often characterized by highly dynamic end-points, the maintenance of logical abstractions, and – sometimes – integration with or offloading to special purpose switching hardware.

Floodlight. OpenFlow is an open standard managed by Open Networking Foundation. It specifies a protocol through which a remote controller can modify the behaviour of networking devices through a well-defined "forwarding instruction set"; and the Floodlight Open SDN Controller [4] is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. Floodlight is designed to work with the growing number of switches, routers, virtual switches, and access points that support the OpenFlow standard.

Queue Installer. With the intention of providing an interface inside the Floodlight controller to ease the process of queue creation within an OF enabled switch, in particular the OvS, a queue installer was created. This entity presents itself as an extension to Floodlight, a module per se, rather than as a standalone application; following this approach allows this module to be easily embedded in every Floodlight installation and also to enable Floodlight to handle all the event processing, avoiding the creation of additional overheads on the controller communication.

The required communication for queue configuration, between an OpenFlow Controller and its corresponding switches, is not foreseen in the current implementation of the OvS used, therefore we have designed an architecture capable of generating the appropriate queue configuration messages, following the Open vSwitch Database Management Protocol (OVSDB) standard [5]. It should be noted also that the Open Networking Foundation has recently created a working group for the Configuration and Management of queues and other hardware related configurations.

RouteFlow. RouteFlow [6] is an open source project to provide virtualized IP routing services over OpenFlow enabled hardware. Mimicking the connectivity created by a physical infrastructure and running IP routing engines, RouteFlow is composed by an OpenFlow Controller application, an independent RouteFlow Server, and a virtual network environment.

The main issue of RouteFlow for large-scale experiments is that an administrator needs to devote a long time in manual configurations: (1) creating a virtual environment, (2) creating mapping between a virtual environment and the physical infrastructure, and (3) writing routing configuration files for each machine of the virtual environment. Therefore, for large-scale experimentation, we implemented a framework to configure RouteFlow automatically. In our framework [7], we use an additional module that gathers configuration information by sending probe messages in the physical infrastructure. The configuration information is then sent to RouteFlow using configuration messages. Using these messages, RouteFlow configures itself.

QoS Platform. The QoS platform is service-based, meaning that everything being done within the platform is about setting up services to provide for clients. This will be achieved in the following manner:

1. Setting up basic data connectivity for every equipment – *Routers*;
2. Setting up basic service models – *Trunk, Interconnected Trunk, Node*;
3. Adding equipment to establish the service;
4. Setting up service specific configurations for the aforementioned equipment;
5. Publishing the information to the appropriate path table, or to an interconnected partner;
6. Setting up session services for service instantiation;
7. Publishing session services to the appropriate clients

The aforementioned clients are mostly applications that use a provided Application Programming Interface (API) to request service operations, like triggering, stopping or modifying existing services.

2.3 Supporting Tools

The following tools were mainly designed and created to be of service to the experimentation process, predominately aiding in the generation and collection of data.

Pulse Generator. The Pulse Generator is a standalone application that was defined and implemented in order to realistically emulate a large number of users. It generates both data plane and control plane traffic and uses a customizable profile to allow for varying traffic characteristics.

The Pulse Generator reads a traffic profile and schedules each connection profile described therein to take place at some point in the future. A connection profile is a set of parameters that describe a repetitive connection to the QoS Platform (in the control plane) and optionally also a data-traffic generator (in the data plane). The period of this repetitive connection can be as low as tens of milliseconds. The connection profile also indicates the duration of the repetitive connection (or if only a single connection is required), the lifetime of each service and if data-traffic is required. As there are two parts to a service event on the control plane (invocation/trigger and stop/termination), the pulse generator handles both.

Measurement System. A proper data collection and processing mechanism must to be required in order to present concrete and detailed results from pre-established metrics within the experiments. Although mainly focused on Java technology, existing profilers are not optimized for performance using the Java Virtual Machine, therefore creating an unwanted overhead if used to assess the capability of a given module.

Striving to provide deterministic and accurate methods for measuring data within the software being used, a tailored library was designed and developed for distribution and integration with any software that may be used in the testbed. The decision to produce a library in lieu of a standalone application, for the most part, comes from the requirement of measuring actions that take place inside the core of the used software for each scenario, in order correctly understand the impact of each component and its performance.

Proprietary formats were used to store the information so as the development time could be reduced to a minimum while providing some degree of modularity to the whole system, facilitating integration into newer or different applications.

3 Experimentation

Within the paper, the focus has been on the assessment of large-scale OpenFlow networks and in particular take into account a QoS platform in a dynamic control environment, where a high event arrival-rate is expected as we intend to emulate 1 million inhabitants. Figure 1 depicts the defined large-scale topology for reaching the experiment's goals.

Whether developing a new platform, or constructing and assessing a new set of components, a validation process must take place. This will be guaranteed by defining relevant KPIs and different experiments that targeted at several distinct goals.

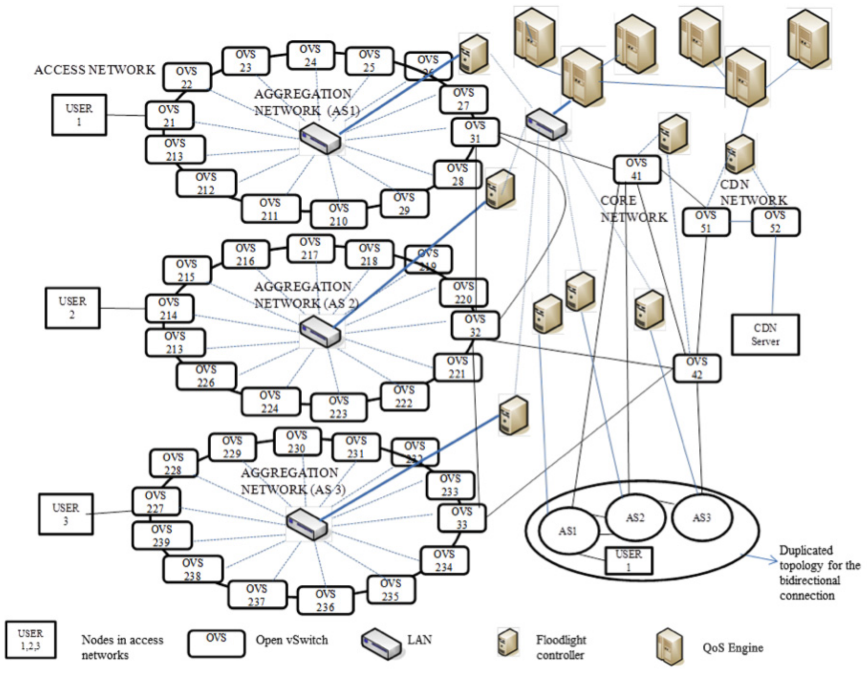

Fig. 1. Large-scale topology

3.1 Key Performance Indicators

The selection of appropriate Key Performance Indicators is crucial for a suitable evaluation of the defined scenarios. Both qualitative and quantitative KPIs may be considered in the selection process as long as they are relevant for the assessment being conducted. The overall set of KPIs should also consider all of the aspects of the solution under consideration. Moreover, the identified KPIs should not be redundant and their measurement must be possible either through objective or subjective (like when assessing Quality of Experience) techniques [8].

In addition to the selection of an appropriate set of KPIs, it is also important to understand what are the expected outcomes of the performed experiments and what constitutes a successful experiment. By assessing the performance results obtained for the selected indicator, the experiment will be considered successful if they are within the pre-defined lower and upper limits. These KPIs reflect not only the competence of each constituent module but also the performance of both voice and video traffic.

These and other network related parameters are further detailed in the following bullet points that present the metrics to be considered and their expected values. One considers a parameter to be successful whenever it is within these limits, presented in italic together with the component responsible for the measurement.

CPU and Memory Usage. Overall CPU and Memory usage on the machines being monitored – *Applicable to each component;* $\leq$*75% (on average).*

Used Bandwidth. Total amount of bandwidth required by the component being monitored – *Applicable to each component.*

Losses. Overall percentage of losses registered on a given interface – *Applicable to each component;* $\leq$*5%.*

Delay. Overall end-to-end delay of the packets being exchanged between different components – *Applicable to each component;* $\leq$*300ms.*

Jitter. Variation of the delay – *Experiment framework;* $\leq$*75ms.*

Packet Loss. Maximum percentage of losses of voice traffic – *Experiment framework;* $\leq$*3%.*

Peak Signal-to-Noise Ratio (PSNR). Percentage of the impact of noise in the corruption of the fidelity of video frames (pixel-based comparison) – *Experiment framework;* $\geq$*35%.*

Video Quality MOS. Assess the mean opinion score of the overall received video – *Experiment framework;* $\geq$*3.7.*

Host Discovery Time. Time taken to find the endpoints in the network via the OpenFlow Controller – *QoS Platform & Measurement system.*

Host Path Discovery Time. Time taken to find the path between endpoints or between a single endpoint and the core network (in case of inter-domain requests) via OpenFlow Controller – *QoS Platform & Measurement system.*

Queue Installation Time. Time taken to install a queue in a switch via the OpenFlow Controller – *QoS Platform & Measurement system.*

Queue Removal Time. Time taken to remove a queue in a switch via the OpenFlow Controller – *QoS Platform & Measurement system.*

Flow Installation Time. Time taken to install a flow in a switch via the OpenFlow Controller – *QoS Platform & Measurement system.*

Flow Removal Time. Time taken to remove a flow in a switch via the OpenFlow Controller – *QoS Platform & Measurement system.*

Route Creation Time. Time taken by RouteFlow to add a new route in a switch – *Route Flow; It depends on the defined hello interval.*

Route Deletion Time. Time taken by RouteFlow to delete a route in a switch – *RouteFlow; It depends on the defined router dead interval.*

By having a very specific set of KPIs, a more detailed study of the whole platform can be accomplished, enabling an easier and faster improvement over each batch of trials towards the platform optimization.

3.2 Preliminary Tests

Before setting into automated testing, a proper platform validation was required, so a scenario was set up in order to assess the inner workings of the selected tools and how they would cope in these scenarios.

A set of simple tests at low scale will be undertaken. The purpose of this experiment is to assess the functionality of all components and their interoperability, therefore validating the existing platform for the upcoming experiments. These experiments would be executed manually, always taking into consideration the feedback received from each individual run before proceeding with the next set of trials. For the first experiment, to validate the designed topology and verify if the packets where travelling through the expected paths, single triggers on the network were manually issued.

The results will be obtained via a script that parses the logs to create a list of average response times based on the 85%, 90%, 95% and 100% best results. These results have to be considered very preliminary, as their main intent is to further validate the platform. Having said, the first run will have a duration of 10 seconds, with triggers spaced by one second. Provided the platform copes well with the previous validation tests, these will be extended to a total duration of one hour, with triggers spaced by 200 milliseconds.

3.3 Main Experiments

Following the preliminary tests, a larger set of experiments to understand the performance of the components employed was designed. In total five experiments were designed, with the initial four aimed at stress testing the platform and verifying its resiliency. In the fifth and final experiment, the previous scenarios will be extended as to encompass as many islands of the OFELIA testbed as possible.

1st Experiment. The main purpose is to improve over the preliminary experiments, solving any problems that might have risen from an experimental operation point of view. This experiment can be perceived as an extension to the integrated experimental platform validation, where a whole assessment of the system is performed in order to prepare it for the next five, fully featured, experiments. It will aim to confirm that the OpenFlow connectivity and bandwidth guarantee can be achieved, and that each of the 50 OvS can be set-up and brought down again.

2nd Experiment. This experiment sets out to stress test the signalling in the context of high signalling load. Single and multiple domain scenarios will be stress tested. Variations regarding connection duration will be introduced. The maximum performance will be obtained under various network conditions. The experiment will be repeated following any improvements identified and implemented during the testing.

3rd Experiment. Here the resilience of the unidirectional communication will be stress-tested to support right of way for World Wide Web HD video service in the presence of background traffic (i.e., assessing if the system maintains a sustainable bandwidth level for the invoking application). The stress test will be conducted in the context of various network conditions of background traffic load, focusing on unidirectional traffic from a Content Delivery Network (CDN) to an end user, and later on using instead bi-directional video-to-video traffic from one end-user to another.

4th Experiment. This experiment will assess how the system behaves under several different failure scenarios. A framework will be designed so that the controller will detect data plane failures, with data plane traffic (high-priority and best-effort) being rerouted on failure free paths. In this experiment, three failure recovery scenarios will be evaluated. In the first scenario, enough bandwidth will be assumed so that neither high-priority nor best-effort traffic will be affected after all the traffic has been re-routed to a failure-free path. In the second scenario, the available data path bandwidth will be restricted so that all priority traffic can be rerouted as soon as all flows are recovered. However, some of the best-effort traffic flows will experience packet loss in order to meet the requirements of high-priority traffic. In the third scenario, the capacity is squeezed further so high-priority traffic will also be starved for bandwidth after all traffic is redirected to a failure-free path.

5th Experiment. This experiment is based on expanding the previous learning experience of the experiments described above, in order to expand the acquired knowledge to a deployment based on a set of federated distributed island testbeds. It will take a version of one of the small-scale experiments designed, and implement it on the other available islands from the OFELIA project. The experiment will try to assess the behaviour of the control plane in a multi Autonomous System (AS) scenario where each AS is located in a different OFELIA island. It will be addressed the question of how operational issues faced in a heterogeneous environment impact the components in the software stack.

3.4 Methodology

Through the controlled generation of pulses – where a pulse is everything between the trigger creation, call duration and trigger removal – with the Pulse Generator, their number in a given interval will be gradually increased between runs. The value will be increased until a breaking point is reached, i.e., where the platform stops responding to the trigger creation requests due to the amount of currently instantiated triggers.

These pulses will be distributed within a two-hour interval according to a Poisson distribution, providing a more accurate representation of a real world scenario. Upon identifying the breaking point, a series of repetitions will be performed; later assessing the validity of the results obtained by the Measurements framework and the Pulse Generator logs.

4 Results

As previously mentioned preliminary results were obtained from performing a small scale evaluation on top of the used OpenFlow testbed. From these results we established a methodology that enables a thorough and rigorous method of assessment for each of the developed and used components. These components and structure definition will allow a thorough assessment of realistic future networks resorting to OpenFlow.

Despite the defined approach, the actual performance evaluation and results collection for the large-scale testbed are scheduled but not presented in this work. The results obtained in upcoming iterations will further validate the proposed methodology which currently stands as the main contribution and outcome from this work, which includes not only the definition of 5 realistic large-scale scenarios but also the necessary tools and components for the desired evaluation.

5 Conclusion

In order to establish a methodology and performance analysis of a real large-scale Software-Defined Networking testbed a set of preliminary tests was performed. During these preliminary tests, some issues arose and were promptly identified and circumvented. They provided insights and directions towards the presented final scenarios, which allow emulating future smart cities with millions of inhabitants.

In addition to the defined scenarios, a set of necessary components was also identified and implemented as explained in the paper, being now ready to be used in order to undertake the totality of the experiments described above. The utmost purpose will be to perform these tests at the higher possible performance levels, identifying bottlenecks on the current setup by analysing the results meticulously, while also validating the proposed solution and contributing for the development of more robust and innovative solutions for the assessment of the Future Internet.

References

1. OpenFlow in Europe: Linking Infrastructure and Applications, `http://www.fp7-ofelia.eu`
2. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: Enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 69–74 (2008)
3. Open vSwitch: An Open Virtual Switch, `http://openvswitch.org`
4. Floodlight OpenFlow Controller, `http://www.projectfloodlight.org/floodlight/`
5. Open Networking Foundation, Configuration and Management working group, `https://www.opennetworking.org/working-groups/configuration-management`
6. Nascimento, M.R., Rothenberg, C.E., Salvador, M.R., Corrêa, C.N.A., de Lucena, S.C., Magalhães, M.F.: Virtual routers as a service: The routeflow approach leveraging software-defined networks. In: Proceedings of the 6th International Conference on Future Internet Technologies, CFI 2011, pp. 34–37. ACM, New York (2011)
7. Sharma, S., Staessens, D., Colle, D., Pickavet, M., Demeester, P.: Automatic configuration of routing control platforms in openflow networks. SIGCOMM 43, 491–492 (2013)
8. Schatz, R., Hossfeld, T., Janowski, L., Egger, S.: Datatraffic monitoring and analysis, pp. 219–263. Springer, Heidelberg (2013)