Demonstrator for Experimental Evaluation of Large-Scale Distributed SDN Deployments

Hemanth Kumar Ravuri, Maria Torres Vega, Jeroen van der Hooft, Tim Wauters, Filip De Turck
Ghent University - imec, IDLab, Belgium
Email: hemanthkumar.ravuri@ugent.be

***Abstract*—The software-defined networking (SDN) paradigm has gained widespread popularity due to its ability to ease network management. However, the traditionally used centralized SDN architectures are restricted by scalability issues. To cope with these, several distributed alternatives have been proposed. The performance evaluation of such distributed solutions is limited to simulation-based methodologies, where virtualization technologies are used (e.g., virtual machines (VMs) and containers to distribute the control plane). In this context, large-scale testbeds offer a platform to validate the applicability of an architecture to real-world network conditions. The Virtual Wall is one such large-scale generic experimentation facility for advanced networking research and testing. This work proposes a demonstrator for large-scale distributed SDN deployment using the Virtual Wall. It provides a thorough description of the steps involved to deploy experiments, by evaluating the scalability of a hierarchically distributed control plane.**

***Index Terms*—Controller, Distributed, Scalability, SDN**

I. Introduction

Over the last decade, the software-defined networking (SDN) paradigm has gained attention from both industry and academia. Its evolution can be traced to problems posed by traditional networks in terms of rigidity and inability to dynamically adapt to changing conditions [1]. The SDN paradigm, where the control plane is decoupled from the data plane, supports management, automation and application-specific routing policies. This has allowed researchers to envision new networking architectures [1]. So far, several mechanisms ranging from simulators, emulators, and prototype testbeds have been deployed for performance evaluation of such architectures. In this context, experimentation facilities offer a possibility to examine the performance of the architecture under near real-world conditions. Furthermore, they provide experimental resources in the range of hundreds of physical nodes [1]. This becomes particularly important when evaluating distributed SDN architectures designed to deal with the shortcomings of the centralized SDN. The control plane needs to be distributed onto several nodes to analyze the gains obtained by the architecture. Thus far, works focusing on performance evaluation of SDN controllers have used simulators or virtualization technologies such as virtual machines (VMs) and containers to deploy the controllers [2].

In this paper, we present a demonstrator for large-scale evaluation of distributed SDN control plane architectures. To illustrate its potential and capabilities, we evaluate the scalability of a hierarchically distributed SDN control plane implemented based on the ZeroSDN controller [3]. This architecture supports independent instantiation of modules, which aids us in showcasing the relevance of the experimentation facility. In order to perform experiments, we use the Virtual Wall[1], a large-scale generic experimentation facility for advanced networking, scalability research, and testing. In the demonstrator, we set up the data plane on a physical node using Mininet[2] and evaluate the scalability performance of the controller when it is distributed on several physical nodes. In this demo paper, we provide a thorough explanation on the experimental design using the jFed[3] experimenter graphical user interface (GUI), resource provisioning, and execution. Finally, we present the results in a GUI and analyze them.

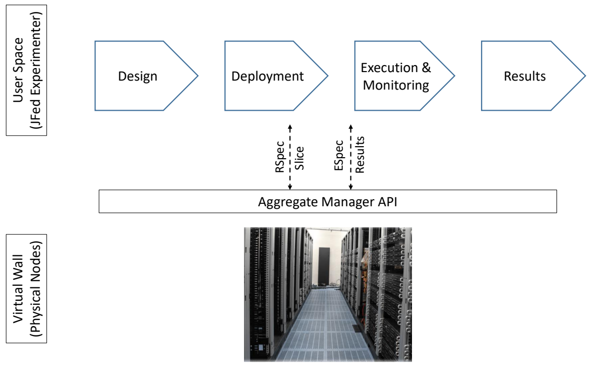

Fig. 1: Deployment of experiments on the Virtual Wall experimentation facility.

II. Experiment Lifecycle Management

This section presents an overview of the architecture of the demonstrator implemented on the Virtual Wall. Fig. 1 shows how the specific infrastructure resource management (bottom) is abstracted from the experiment lifecycle (top). The Virtual Wall is a large-scale generic experimentation facility, hosting as many as five hundred bare-metal servers. All nodes are completely configurable in terms of software installation and network connections. The userspace comprises the jFed experimenter GUI, which aids in requesting resources on the Virtual Wall to conduct experiments. The Aggregate Manager API acts as an interface between the Virtual Wall and the jFed experimenter GUI. Various steps involved in the experiment lifecycle management are as follows:

[1]https://doc.ilabt.imec.be/ilabt/virtualwall/overview.html
[2]http://mininet.org/
[3]https://jfed.ilabt.imec.be/

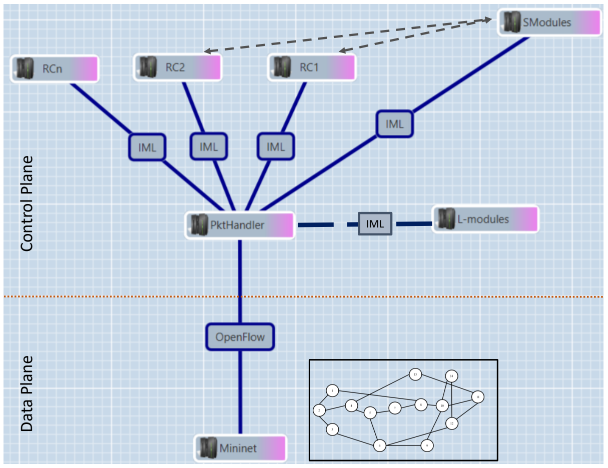

Fig. 2: Illustration of an experimental setup with a hierarchically distributed control plane.

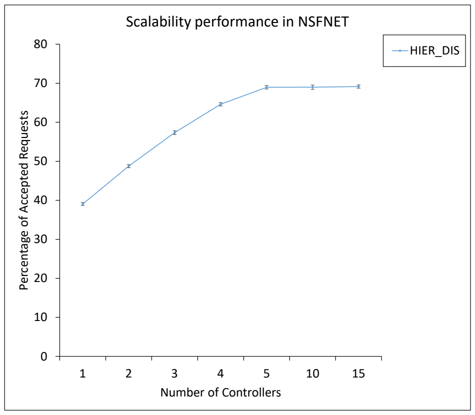

Fig. 3: Obtained scalability evaluation results, with gradual increase of the degree of the distribution.

Design: The jFed experimenter GUI comes with a topology editor that is useful in designing the experimental setup and provide configuration details ranging from the type of nodes to the Operating System (OS) to be installed on a node. The editor also generates an extensible markup language (XML) file following the resource specification (RSpec) format that can be manually edited to add additional features, such as the installation of software on specific nodes. Once the design is complete, the experiment may be run directly from the editor.

Deployment: The jFed experimenter GUI supports an advanced feature for experiment description in the experiment specification (ESpec) format. The ESpec usually contains a RSpec in combination with other files. It allows to bootstrap an experiment by uploading the required software onto different nodes and specifying the order of execution.

Execution: Each step involved in the execution of an experiment can be monitored through the progress window of the GUI. Errors generated during any step are presented in the error window and in the error logs of the specific node.

Results: Once the experiment is complete, all logs and results can be downloaded from the nodes for further analysis.

III. Demonstration Setup

A. *System Under Evaluation*

While the traditionally used centralized SDN architecture suffers from scalability issues, alternatively proposed architectures such as the flat distributed ones do not fare better due to the inherent inter-controller overhead [4]. In this direction, we envision a hierarchically distributed control plane architecture which supports independent modularity, linear scalability and on-demand instantiation. While independent modularity refers to factoring out individual control functions into independently deployable modules, linear scalability refers to the ability to replicate one or more modules. Finally, on-demand instantiation refers to instantiating different modules based on the network load, priority and network requirements.

The demonstrator presents an evaluation of the scalability performance of the hierarchically distributed SDN control plane architecture (Fig. 2). The data plane is implemented using the Mininet emulator, which is connected to the control plane using the OpenFlow protocol[4]. The control plane consists of various modules, also called controllets. The lower tier of the control plane comprises the packet handler (PktHandler) and local modules (L-modules). The PktHandler acts as the point of contact between the data plane and other controllets in the control plane. It is responsible for handling all incoming packets from the data plane and sending them to the responsible modules. Furthermore, it is also responsible for implementing the decisions taken by those modules. The L-modules are responsible for handling events that do not require a global network view. Among others, such events include handling of address resolution protocol (ARP) requests and simple hop-by-hop forwarding. The upper tier consists of the routing controllet (RC) and state modules (SModules). The SModules are responsible for maintaining the global network state and sharing it with all other modules. The network state information includes data about switches, hosts, links, network graphs, and traffic statistics. The RC is responsible for taking all the routing-related decisions using algorithms such as the shortest path algorithm [4]. It is possible to deploy multiple instances of the RC on separate nodes and distribute the incoming requests amongst them. All nodes on the control layer are connected using the internal messaging link (IML). Such an architecture aids in incorporating local-level decision-making and scalability by distributing the load. It is implemented using the ZeroSDN controller as described in one of our previous works in which the architecture has been proposed [5]. Furthermore, the detailed request handling mechanism of the architecture can be also be found there.

B. *Demonstration*

Experiment Design and Deployment: The demo starts by introducing the jFed experimenter GUI, followed by designing the experiment with the topology editor. Furthermore, we explain the provisioning of resources using a RSpec. Then,

[4]https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

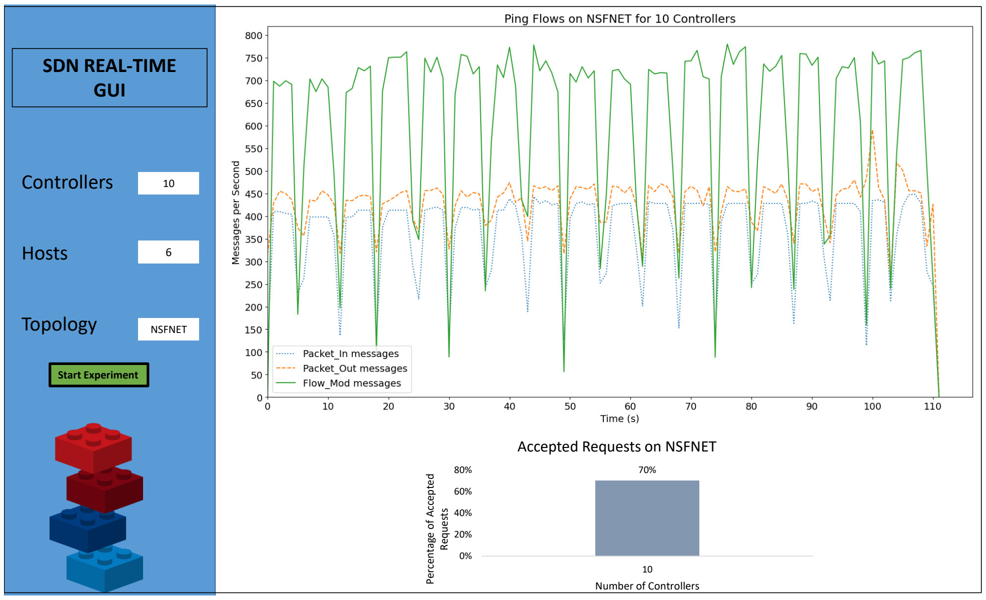

Fig. 4: Real-time GUI to perform scalability experiments.

we configure the provisioned resources using an ESpec. Thus, we deploy the control plane as shown in Fig. 2.

Execution: Then, we execute the experiment by starting the Mininet-based data plane on a separate node. We use an out-of-band controller connection. As part of the experiment, we emulate the data plane using an extension of the NSFNET topology. Fig. 2 presents the core network of the topology. In the experiment, each core switch has two edge switches, to which the emulated hosts are connected. Then, we generate the flow requests by making the hosts ping each other simultaneously. Each ping request has a default timeout of 500 milliseconds, and a successful ping request is considered as an accepted flow request by the controller. We log all the messages exchanged in the process of setting up flows. For ease of use, we present a GUI (Fig. 4) where the user can directly enter the configuration parameters to start a new experiment and check the resulting message flows dynamically over time.

Results and Analysis: The results of the flow requests can be verified in the log files. The GUI (Fig 4) presents all the messages exchanged between the data plane and the control plane, along with the scalability in terms of the percentage of accepted requests. The number of Flow_mod messages is higher compared to the rest because the controller has to install flow rules on multiple switches per request. Fig. 3 shows the scalability performance for an increasing number of RCs. The scalability increases until five RCs and further the curve flattens due to the PktHandler maxing out.

The demonstration highlights how distributed SDN-based experiments can be performed seamlessly on a number of bare-metal nodes. Furthermore, it provides insights into various aspects of the hierarchically distributed control plane architecture, ranging from its scalability to possible bottlenecks such as the PktHandler maxing out.

IV. Conclusions

In this paper, we propose a demonstration of an experimentation setup based on the Virtual Wall experimentation facility and Mininet for large-scale SDN experiments. The presented setup is validated by performing a scalability evaluation experiment for a hierarchically distributed controller. In the demonstration, we show the various steps involved in the experiment lifecycle, ranging from design to deployment. Finally, we present and analyze the results in real-time with a GUI. Our approach can be adapted to various other architectures and use cases of SDN, ranging from a data center network to content delivery networks.

Acknowledgment

This research is the result of a collaborative project between Huawei and Ghent University, and funded by Huawei Technologies, China. Maria Torres Vega and Jeroen van der Hooft are funded by the Research Foundation Flanders (FWO), with grant numbers 12W4819N and 1281021N, respectively.

References

[1] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, “A survey on large-scale software-defined networking (SDN) testbeds: Approaches and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 891–917, 2016.
[2] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, “SDN controllers: Benchmarking & performance evaluation,” *arXiv preprint arXiv:1902.04491*, 2019.
[3] T. Kohler, F. Dürr, and K. Rothermel, “ZeroSDN: A highly flexible and modular architecture for full-range distribution of event-based network control,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1207–1221, 2018.
[4] H. K. Ravuri, M. Torres Vega, J. van der Hooft, T. Wauters, B. Da, and F. De Turck, “On routing scalability in flat SDN architectures,” in *2020 11th International Conference on Network of the Future (NoF)*. IEEE, 2020, pp. 23–27.
[5] H. K. Ravuri, M. Torres Vega, J. van der Hooft, T. Wauters, and F. De Turck, “A scalable hierarchically distributed architecture for next-generation applications,” *Springer's Journal of Network and Systems Management*, 2021 [Accepted for Publication].