Mantoo - a set of management tools for controlling SDN experiments

Stefano Salsano[(1)], Pier Luigi Ventre[(2)], Francesco Lombardo[(1)], Giuseppe Siracusano[(1)], Matteo Gerola[(3)], Elio Salvadori[(3)], Michele Santuari[(3)], Mauro Campanella[(2)], Luca Prete[(4)]
(1) CNIT / Univ. of Rome Tor Vergata - (2) Consortium GARR - (3) CREATE-NET – (4) ON.Lab

Abstract – OSHI – Open Source Hybrid IP/SDN networking is a hybrid approach allowing the coexistence of traditional IP routing with SDN based forwarding within the same provider domain. In this demo, we will show a set of Open Source management tools for the emulation of the proposed solution over the Mininet emulator and over distributed testbeds. We refer to this suite of tools as Mantoo (Management tools). Mantoo includes an extensible web-based graphical topology designer providing different layered network "views" (e.g. from physical links to service relationships among nodes). The framework is able to validate a topology, to automatically deploy it over a Mininet emulator or a distributed SDN testbed, to access nodes by opening consoles directly via the web GUI.

Keywords - Software Defined Networking, Open Source, Network management tools, Emulation.

I. MANTOO: MANAGEMENT TOOLS FOR SDN EXPERIMENTS ON MININET AND DISTRIBUTED SDN TESTBEDS

Mantoo is a set of Open Source tools meant to support SDN experiments over emulators and distributed testbeds. Mantoo is able to drive and help the experimenters in the different phases that compose an experiment: design, deployment, control and measurement, as described in the next subsections. Mantoo includes: a web based GUI called Topology3D (Topology and Services Design, Deploy and Direct, Figure 1), a set of scripts to configure and control emulators or distributed testbeds; a set of scripts for performance measurements. The overall Mantoo workflow is represented in Figure 2. Using the Topology3D, the user can design its experiment in terms of physical topology and services, start the deployment of the topology and run the experiments exploiting the provided measurement tools. The design of Mantoo and of its components is modular and it can be easily extended to support scenarios that go beyond the use cases of our interest. The first implementation of our management tools has been described in [2].

A. *Design Phase*

The Topology3D offers a web GUI to design a network topology and to configure the services for an experiment (see Figure 1). It consists in a JavaScript client and a Python back-end. A link to a public instance of the Topology 3D can be accessed from [1]. The Topology3D is meant to be an extensible framework that can support different *models* of topology and services. A model corresponds to a technological domain to be emulated and is characterized by the set of allowed node types (e.g. routers, switches, end-hosts), link types, service relationships and related constraints. As shown in Figure 2 the Topology3D takes in input a textual description of the model. The model description is used to configure the topology designer page, to enforce the constraints when the user is building the topology and/or during the validation of the topology. So far, we have provided two models: 1) the OSHI topology domain, including two types of OSHI nodes (Core Routers, Provider Edge routers, see [2]), the Customer Edge routers which are also used as traffic source/sinks and the SDN controllers; 2) a generic layer 2 network with OpenFlow capable switches, end-nodes and SDN controllers. Each model is decomposed in a set of *views*. A view is a perspective of a model, which focuses on some aspects hiding the unnecessary details. For example, the OSHI model is decomposed in 5 views: data plane, control plane and 3 views for the 3 services (IP VLLs, Pseudo Wires and Virtual Switches). In the data plane view, users can design the physical topology in terms of nodes (OSHI CR and PE, Controllers, and CEs) and links; in the control plane view, users define the association of OSHI nodes with controllers; in the service views users can select the end points of the services.

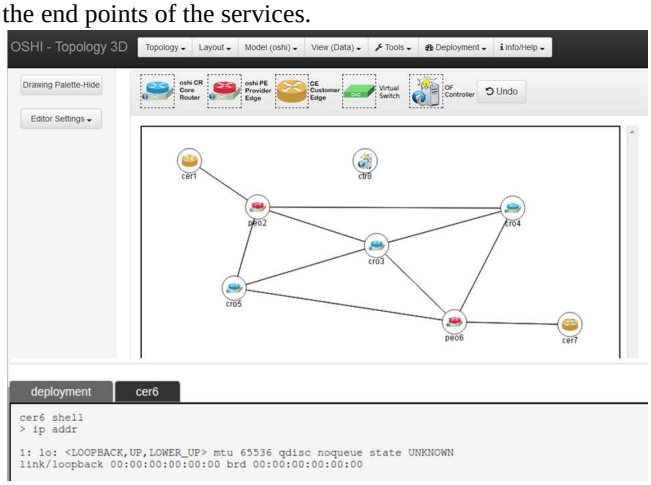

Figure 1. The Topology3D (Design, Deploy & Direct) web GUI

Topology3D integrates Networkx, a Python package for the creation/manipulation of complex networks, making it also possible to randomly generate a data plane topology with certain given characteristics. The Topology3D exports the representation of the views (topology and services) in a JSON format, which becomes the input for the deployment phase.

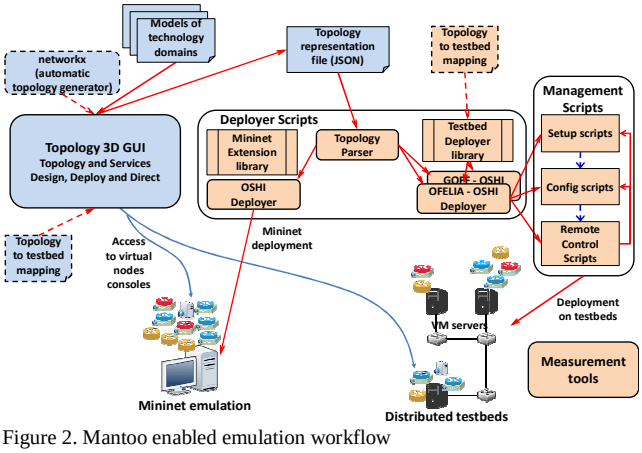

Figure 2. Mantoo enabled emulation workflow

B. *Deployment phase*

The deployment phase translates the designed topology into the set of commands that instantiate and configure the nodes and the services for a given experiment. This phase can target different execution environments for the experiments, by means of a specific "Deployer". So far, we targeted one emulator (Mininet) and two distributed SDN testbeds (the OFELIA testbed [3] and the GÉANT OpenFlow Facility - GOFF). By default, Mininet only provides the emulation of hosts and switches. We extended Mininet introducing an extended host, capable of running as a router and managed to run the Quagga and OSPFD daemons on it. Then we have added Open vSwitch to it, as needed to realize the OSHI node. The details on the specific Mininet deployment architecture can be found in [4]. The Mininet Extensions library is able to automate all the aspects of an experiment. This includes the automatic configuration of IP addresses and of dynamic routing (OSPF daemons) in all nodes, therefore relieving the experimenter from a significant configuration effort. The Mantoo framework is modular so that an experimenter can add its own deployer targeting a different specific execution environment.

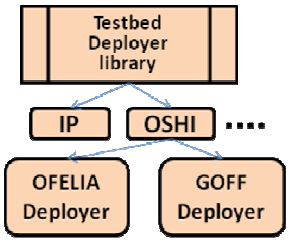

Figure 3. Testbed Deployer

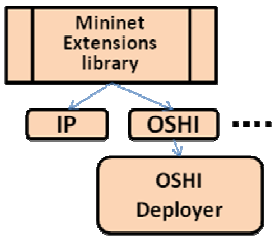

Figure 4. Mininet Extensions

We implemented and tested a Deployer for the OFELIA testbed and one for the GOFF testbed. These two testbeds share a similar architecture as they are based on the OCF (OFELIA Control Framework) [3]. The Management Scripts automate and facilitate the setup, configuration and the deployment of an experiment. They relieve the experimenter from tedious and error prone activities. A management host coordinates the overall process, usually also executing the Deployer scripts. The management host and the VMs communicate over a management network. The configuration files generated by the Deployers scripts are uploaded on a repository reachable by the VMs (e.g. a webserver running on the management host) and these files are downloaded by each VM belonging to the experiment.

In order to replicate an experimental topology emulating the network links an overlay of Ethernet over UDP tunnels is created among the VMs, based on VXLAN tunnels provided by Open vSwitch. The design of the VXLAN tunneling solution for OSHI over a distributed testbed is reported in Figure 5. Overlay VXLAN tunnels are associated to ports of the OpenFlow capable switch (Open vSwitch). The nice thing is that the configuration of the switch is the same as if its ports were physical ports. Therefore it is possible to have a realistic emulation of the node operations concerning its control plane.

C. *Control phase (running the experiments)*

In the Mininet based experiments it is possible to open consoles on the emulated nodes using the web GUI of the Topology3D. The consoles show the output generated by the ssh processes connected to the nodes (deployed in the Mininet emulator). The generated output is conveyed to the terminal shell running in the experimenter browser, leveraging the WebSocket API, where each terminal has a separate WebSocket channel.

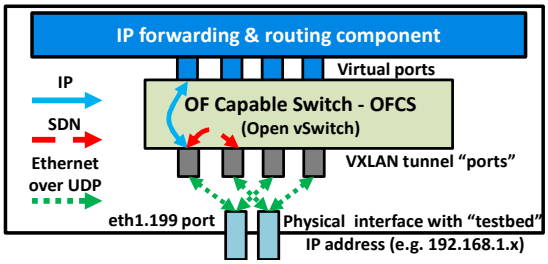

Figure 5. Implementing VXLAN tunnels using Open vSwitch (OVS)

D. *Measurement Phase*

In order to automate as much as possible the process of running the experiments and collecting the performance data over distributed testbeds we have developed an object oriented multithreaded Python library called Measurement Tools. The library offers an intuitive API that allows the experimenter to "program" his/her tests. Using the library we can remotely run the traffic generators (iperf) and gather load information (CPU utilization) on all nodes (VMs).

II. DEMO DESCRIPTION

In the demo we show how to graphically design a topology and a set of services and then to automatically deploy the experiment over the Mininet emulator and collect performance measurements. Open vSwitch is used to realize the hybrid IP/SDN nodes. Two types of services have been implemented: the IP "Virtual Leased Line" (IP VLL) and the Layer 2 "Pseudo-wire" (PW). Both services are offered between end-points in Provider Edge routers, the end-points can be a physical or logical port (i.e. a VLAN on a physical port) of the PE router connected to a Customer Edge (CE). The tunneling of the services is realized in the core hybrid IP/SDN network using either VLAN tags or MPLS labels.

III. ACKNOWLEDGMENTS

This work was partly funded by the EU in the context of the projects: GÉANT GN4 Phase 1 (GN4-1) [5], FP7 NETIDE [6], DREAMER [7] (one of the beneficiary projects of the Open Call research initiative of GN3plus [8]).

IV. REFERENCES

[1] OSHI homepage http://netgroup.uniroma2.it/OSHI

[2] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, E. Salvadori, "Open Source Hybrid IP/SDN networking (and its emulation on Mininet and on distributed SDN testbeds)", EWSDN 2014, 1-3 September 2014, Budapest, Hungary

[3] Marc Suñé et al., "Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed", Computer Networks, Vol. 61, March 2014

[4] P. L. Ventre et al. "OSHI technical report" available at [1]

[5] http://www.geant.org/geantproject/About/Pages/GN4_Phase_1.aspx

[6] http://www.netide.eu/

[7] DREAMER home page - http://netgroup.uniroma2.it/DREAMER/

[8] http://www.geant.net