Emulated and Software Defined Networking Convergence

Marcos F. Schwarz[1], Marco A. T. Rojas[1], Charles C. Miers[1,2], Fernando F. Redigolo[1], Tereza C. M. B. Carvalho[1]
[1]Escola Politécnica, University of São Paulo (USP), São Paulo, Brazil
{mschwarz, matrojas, cmiers, fernando, carvalho}@larc.usp.br
[2]Santa Catarina State University (UDESC), Joinville, Brazil

Abstract— **Emulab is a network testbed that provides network topologies defined by the user, in a controllable, predictable, and repeatable environment. It includes PC nodes with full access to the experimenter, running an operating system of his/her choice. Emulab works as a programmable patch panel to create an arbitrary topology by dynamically configuring network switches. The current approach that provides the switches programmability has strict requirements. These restrictions limit the models of switches to build an Emulab facility and demand a significant amount of work in the process of adding new switch models. This paper proposes the implementation of OpenFlow as an interface to configure the Emulab switches; this provides a new approach to add new switch models and improves the list of supported switches in Emulab with OpenFlow-compliant models.**

Keywords- SDN; OpenFlow; Emulab; Future Internet; testbeds

I. Introduction

Emulab [1] is a network testbed developed at the University of Utah that provides an experimentation environment, allowing researchers to configure and access networks composed of emulated, simulated, and wide-area nodes / links. Emulab works as a programmable patch panel that allows the creation of an arbitrary topology, only limited by the number of nodes and interfaces available in the facility. Emulab uses SNMP (Simple Network Management Protocol) to dynamically configure the switches and emulates the network topology described in the experiment. The current implementation approach based on SNMP uses proprietary MIBs (Management Information Base) to configure the switches. This approach requires a significant amount of work to support new switch models in Emulab infrastructure. In addition, it has restricted the options of switches that can be deployed to build an Emulab facility.

The OpenFlow protocol provides an open and standard approach to the switch to communicate through an external controller [2]. OpenFlow specifies a standard interface through which entries in the switch forwarding table can be defined externally. This approach provides similar results in comparison to SNMP usage in Emulab. OpenFlow provides access to the data plane of a switch over the network and is a component of Software Defined Networking (SDN). In SDN the control plane and data plane of network equipment is separated. The control plane runs on external servers and the network equipment is only responsible for data plane forwarding. The forwarding tables (FIB) are programmed by the OpenFlow API [3].

This paper is organized in 6 sections. Section II describes the Emulab infrastructure. Section III presents the restrictions and limitations of the current Emulab implementation. After that, Section IV presents the related work. Section V introduces our proposal to use OpenFlow on Emulab infrastructure aiming to overwhelm the aforementioned limitations. Section VI details the implementation features required to achieve OpenFlow support in Emulab infrastructure. Finally, section VII shows the benefits and the limitations of the presented proposal and relates the future work.

II. Emulab Infrastructure

There are more than twenty Emulab facilities around the world, from testbeds with some few nodes up to testbeds with hundreds of nodes. The main Emulab facility is maintained by the Flux Group at the University of Utah [4]. Emulab is widely deployed in research of computer networking and distributed systems. Emulab infrastructure is composed by:

- **Master server**: it runs all the critical software components like database, web server, name server and access to node power cycling.
- **Users server**: it corresponds to the fileserver. This server is accessible to users to provide shared home directories and to store project files, such as, experiments results and measurements.
- **Local nodes**: each Emulab facility provides a variety of computers that are available to users. They can run a variety of operating systems images (e.g., FreeBSD, RedHat, Fedora, and MS-Windows). Researchers can also customize the provided OS images or load their own image to run other OS.
- **Experimental network switches**: the experimental network is exclusive for the experiment traffic. All local nodes are connected to these switches. The experimental network is used to create arbitrary and isolated topologies

978-3-901882-50-0

that are deployed using VLANs. The VLANs are used to allow multiple experiments that deploy different network topologies to coexist simultaneously without interference.

- **Control network switches**: The control network interconnects the master server, the users server, and local nodes. It also connects the testbed to the Internet. Each local node is connected to a separate control network, physically isolated from the networks that are used for experimental traffic.

Emulab uses SNMP to dynamically configure the switches from the experimental network to emulate the network topology described in the experiment. For each experiment, the requested nodes are allocated and the links attributes between these nodes are configured. Emulab works as a programmable patch panel and it is possible to create an user defined topology, only limited by the number of nodes and interfaces available in the facility. To connect nodes located at different experimental switches it is necessary to create VLAN tunnels between these nodes to transmit the packets of this experiment's VLAN from one node to the other. Emulab can also configure node's interfaces rate and mode (full/half-duplex). The Experimental network switches have additional requirements according to the Emulab documentation [4]:

- **Number of ports**: The number of ports depends directly on the number of experimental interfaces in the nodes. The nodes need to have at least two experimental interfaces.
- **VLAN**: VLAN support is required and its configuration has to be done dynamically via SNMP.
- **VLAN trunking** (IEEE 802.1q): Multiple experimental switches require a way to trunk VLANs between them. It is recommended the trunk links to be at least an order of magnitude faster than node links.
- **Vendor**: Emulab uses proprietary MIBs to create the VLANs dynamically. These MIBs are, at times, implemented differently by vendors in different switch models. There is a list of officially supported switches [4], which are known to work properly in an Emulab testbed.

III. Problem Definition

The current approach in Emulab to provide the switches programmability has strict requirements. It is required a customized backend for each family of supported switches. This customizations demand a significant amount of work in the process of adding new switch models.

Most features used by Emulab to dynamically configure the switches through SNMP are only available in vendors proprietary MIBs, due to the lack of standardized MIB to manipulate VLANs and VLANs trunks. Usually, each supported switch model in Emulab demands the effort to discover the correspondent MIB OID assigned to the desired resource. For example, about twenty different MIB OIDs were used in the Emulab implantation just with Cisco switches.

As each switch vendor implements different MIB OID, Emulab uses customized scripts for each new switch vendor/model supported in Emulab infrastructure. The amount of work required to create this customization adds complexity and limits to the number of supported switches. It was reported in Emulab administration mailing list [5] that the amount of time required to implement a new switch support is estimated on two weeks for people already familiar with the SNMP customized scripts in Emulab and up to two months for someone unfamiliar with them.

Emulab officially supports the following switches models from five different vendors [4]: Cisco Catalyst 65xx, 55xx, 40xx, 35xx and 29xx series, Nortel 1100 and 5510, Foundry 1500 and 9604, HP Procurve 5400, 3500 and 2800, and Intel 510T.

Even though there is a considerable amount of switches officially supported by Emulab, some of these models have already reached End-of-Sale or End-of-Life status and are no longer being manufactured [6, 7, 8, 9]. Furthermore, support to new switch models is not very common, for instance, the last switch announced as officially supported by Emulab was the HP ProCurve series on September 11, 2008 [4].

IV. Related Work

It is already possible to use OpenFlow in Emulab, but for different purposes. There are software implementations of OpenFlow switches, like Open vSwitch [10], that can be deployed in local nodes to be used in experiments.

There is an implementation of an OpenFlow switch for NetFPGAs [11], which the Emulab software supports and is available at Utah Emulab [4]. This implementation is only supported to be used in experiments.

The HP ProCurve 5400 and 3500 series switches, which are already supported by Emulab, have OpenFlow support. In the current implementation these switches models are managed through SNMP. Meanwhile, OpenFlow is exclusively available to be used in experiments.

V. Proposed Solution

To increase the number of supported switches in Emulab and to simplify the processes of adding new switch models, it has been proposed the creation of an OpenFlow backend, to perform the same features currently implemented by SNMP using proprietary MIBs. OpenFlow provides a standardized interface to manipulate the forwarding table of an OpenFlow-compliant switch [2]. The OpenFlow backend implementation intends to provide a wider option and newer switch models to build, scale and maintain Emulab facilities. OpenFlow-compliant switches need to meet the requirements of Experimental network switches, listed on Section II. We identified how each Emulab requirement can be achieved using Openflow features already available on the current Openflow switches:

- **Number of ports**: There are OpenFlow switches models with a variety of numbers of ports, starting from 24 up to 1536 Gigabit Ethernet ports. This assures that enough ports are available to either small or large sized Emulab.

- **VLAN**: The required VLAN support is present in OpenFlow through the manipulation of VLAN tags on packages. This enables the dynamic configuration of VLANs and provides the same functionality available on the SNMP implementation.
- **VLAN trunking** (IEEE 802.1q): OpenFlow can implement the VLAN trunking feature by setting multiple VLANs flows through the ports between two switches.
- **Vendor**: The OpenFlow backend will add support to switch models that are compliant with OpenFlow. At the same time the SNMP based implementation will continue to be supported to keep the current list of compatible switches.

OpenFlow-compliant switches can be implemented in two different ways: OpenFlow-only or OpenFlow-enabled. OpenFlow-only switch, a.k.a. Pure OpenFlow switch, contains just the OpenFlow protocol, as defined in the OpenFlow 1.0 Specification [12]. OpenFlow-enabled switches, a.k.a. OpenFlow-hybrid, supports simultaneously the OpenFlow Protocol and the "normal" forwarding action, which processes packets using traditional forwarding path of that switch. The traditional forwarding runs standard network protocols, e.g., OSPF, BGP. In OpenFlow-enabled switches the OpenFlow protocol can be enabled in desired ports or VLANs. This aspect makes possible for an OpenFlow-enabled switch to be part of a traditional network and an OpenFlow network simultaneously, e.g., a production network and a research network.

From the various vendors that offer OpenFlow firmware images for their switches, some are experimental, which are only available for testing and researching purposes; and others are already production images. We compiled an initial list of OpenFlow-compliant switches commercially available, which is presented in Table I.

TABLE I. OPENFLOW-COMPLIANT SWITCHES

	OpenFlow Enabled	***OpenFlow Only***
Production OpenFlow Distribution	Brocade CER/CES Brocade MLX Series IBM G8264 NEC PF5240 NEC PF5820	Netgear GSM7328SO Netgear GSM7352SO Pica8 Pronto 3290 Pica8 Pronto 3780 Pica8 Pronto 3920
Experimental OpenFlow Distribution	HP ProCurve 3500 HP ProCurve 5400 HP ProCurve 6600 HP ProCurve 8200 Juniper MX Series	n.a.

VI. Implementation Proposal

The OpenFlow backend intends to simplify the process of creating an Emulab facility. It will offer more switch options and provide a standard implementation that will be consistent through different switch models and vendors. Figure 1 illustrates the experiment workflow including the new elements added to the Emulab architecture by this proposal, the OpenFlow backend and the OpenFlow switch models.

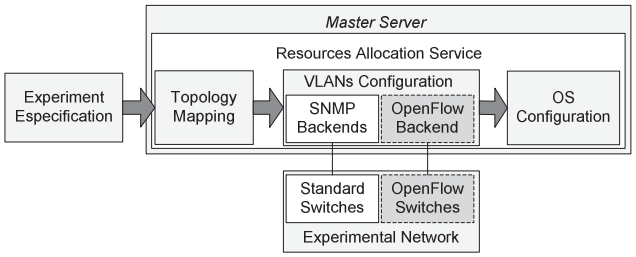

Figure 1. Experiment Workflow with OpenFlow Backend.

Each SNMP command used in Emulab will be implemented with OpenFlow. The dynamic configuration of the switches is done only at the beginning of each experiment and will only require the forwarding entries to be pushed proactively. To control the switch forwarding table the dpctl tool is going to be used, it is part of the OpenFlow reference software [13]. This tool provides commands to remotely or locally manipulate the switch forwarding table, making possible to dynamically read and modify the configuration of the switch.

In Emulab, the SNMP functions to control the experimental switches for VLAN, trunking, and link manipulation are implemented in the snmpit utility. Each vendor has its own backend that extends the snmpit utility implementing the same functions using the specific MIB OIDs for each model [4]. In order to create an OpenFlow backend the dpctl commands will be used to implement each function. Through the analysis of the snmpit backends source code, we identified the similar actions on Table II.

TABLE II. EMULAB FUCTIONS IMPLEMENTED WITH OPENFLOW

Function	***Details***	***dpctl Command***
portControl	Set port rate, port link type, enable/disable ports	mod-port
findVLAN(s)	Find the corresponding VLAN numbers by VLAN ids	dump-flows
vlanNumberExists	Check if the given VLAN number exists on the switch	dump-flows
createVlan	Create a VLAN on switch with the given VLAN id	*Not needed*
setPortVlan	Put the ports into the given VLAN	add-flows
delPortVlan	Remove and disable the given ports from the given VLAN	del-flows
removePortsFromVlan	Remove ports from the VLANs	del-flows
removeVlan	Remove the given VLAN	*Not needed*
vlanHasPorts	Check if the VLAN has any ports	dump-flows
listVlans	List all VLANs on the switch	dump-flows
listPorts	List all ports on a switch device	dump-ports
getStats	Get statistics for ports on the switch	dump-ports
getChannelIfIndex	Get the interface index, for trunking	dump-flows
setVlansOnTrunk	Enable or disable ports on a trunk	add-flows
enablePortTrunking	Enable trunking on a port	*Not needed*
disablePortTrunking	Disable trunking on a port	del-flows

Table II lists the required functions to create a backend. Corresponding dpctl commands were chosen to implement each respective snmpit function. Description of dpctl commands in the OpenFlow backend:

- **dump-ports**: it prints statistics for each interface monitored by the switch in the console. If the port number is specified, statistics are generated only for the interface corresponding to port number.
- **mod-port**: it modifies features of an interface monitored by the switch.
- **dump-flows**: it prints to the console all flow entries in switch's tables that match the flows provided by the user.
- **add-flows**: it adds the flow entry as described by flow to the switch's tables.
- **del-flow**: it deletes entries from the switch's tables that match the flow provided by the user.

There was one function that could not be completely implemented with OpenFlow 1.0, the portControl function. OpenFlow can enable and disable the desired port, but there was no feature found in OpenFlow specification to set the port rate, and the port link type. To implement these features it is possible to use a feature already available in Emulab that uses traffic shaping on the local nodes to set the link speed rate and type, among others such as delay [14].

VII. Considerations

OpenFlow is a feasible alternative to the Emulab SNMP implementation with proprietary MIBs, since it meets the requirements to perform the functions needed by Emulab experimental switches. Our approach to use OpenFlow could provide a contribution to Emulab infrastructure; most evidently the addition of new equipment models, including low cost and high performance switches. Additionally, the OpenFlow backend will provide a more uniform interface to support different switch models, requiring less effort to support newer switch models.

The approach based on OpenFlow requires no additional customization between each OpenFlow switch models/vendors and should reduce the necessary amount of work, as described on section III.

All the analyses in this paper were performed in a current stable version of Emulab (release 10/15/2012). The OpenFlow backend will be available as an additional mechanism to configure switches and will maintain the SNMP based implementation. This provides a better compatibility and offers a broader range of switches.

A. *Future Work*

The proposed OpenFlow backend will be implemented using resources available at USP. We have three years of experience with Emulab and a facility with 24 nodes, additionally there are two OpenFlow switches dedicated to this project. Another approach is to evaluate the scalability of the proposed solution, through the analysis of the OpenFlow switches performance degradation and the flow entries growth according to the number of simultaneous experiments and their topology.

To evaluate the implementation, the following metrics based on non-functional requirements were chosen: The time to configure a given experiment to Emulab switches (this is the process that uses the switch configuration features) and the operational time to add a switch to the Emulab infrastructure.

This work can also be used as a starting point to a service which offers virtual slices of the OpenFlow switches for experiments. FlowVisor [15] could be used to create the isolation between slices and the virtualization of the switch networks.

ACKNOWLEDGMENT

This work was supported by CNPq (Brazilian Council for Scientific and Technological Development) - FIBRE project (590022/2011-3).
This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil.

References

[1] B. White et al. An integrated experimental environment for distributed systems and networks. In Proc. OSDI, pages 255–270, Boston, MA, December 2002.

[2] N. McKeown et al. OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer. Communication Review, 38(2):69–74, April 2008.

[3] Open Networking Foundation - Software-Defined Networking: The New Norm for Networks. ONF White Paper. April 13, 2012

[4] Emulab. Documentation Wiki. https://users.emulab.net/trac/emulab/wiki

[5] Emulab Administrators Mailing list. https://groups.google.com/forum/?fromgroups=#!forum/emulab-admins

[6] Cisco Systems. End-of-Sale and End-of-Life Products webpage. http://www.cisco.com/en/US/products/hw/switches/prod_category_end_of_life.html

[7] Brocade Comm. Systems. Statement of Supported Software Releases for Brocade IP/Ethernet ADP Products. April 2012. http://www.brocade.com/downloads/documents/customer_advisory_notice/supported-software-ma.pdf

[8] HP Networking. End of sale – Switches. http://h17007.www1.hp.com/us/en/products/eos/switches.aspx

[9] Avaya Inc. End of Life products report. http://downloads.avaya.com/css/P8/documents/100113214

[10] B. Pfaff et al. Extending Networking into the Virtualization Layer. In Proc. HotNets, October 2009.

[11] J. Naous et al. Implementing an OpenFlow switch on the NetFPGA platform, Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, November 06-07, 2008, San Jose, California.

[12] B Heller, OpenFlow Switch Specification Version 1.0. 0 (Wire Protocol 0x01), 2009. http://www.openflow.org/wp/documents/

[13] OpenFlow Switching Reference System - http://www.openflow.org/wp/downloads/

[14] Emulab Wiki - End Node Traffic Shaping and Multiplexed Links. http://users.emulab.net/trac/emulab/wiki/linkdelays

[15] R. Sherwood, G. Gibby, K.-K. Yapy, G. Appenzellery, M. Casado, N. McKeowny, and G. Parulkary. Flowvisor: A network virtualization layer. Technical Report TR-2009-01, OPENFLOW, 2009.