WHITE PAPER

SDN Controller Testing: Part 1

Table of Contents

Introduction

The cost, complexity, and manageability of network infrastructure are challenges shared by network operators, service providers, enterprises, and government agencies. These costs inhibit the build-out of new networks and data centers while the complexity slows the time-to-market for new services and applications. Poor manageability further increases operating costs and slows responsiveness to change. Since most existing infrastructures run proprietary operating systems, deploying and managing mixed vendor-networks means operating and managing in a diverse protocol soup.

Ever since server virtualization ensued in the data center, network operators have been looking for a new way to manage and orchestrate the network layer – similar to how they are able to manage the spin-up and configuration of a virtual machine. The emergence of software defined networking (SDN) a few years ago has the promise to decrease cost and complexity by improving management and orchestration through a programmatic interface to the network layer. OpenFlow has emerged as one of the most well-adopted protocols that enable SDN.

The OpenFlow switch specification does not describe how some basic functions, like network discovery, should work and if that should be a feature of the controller or an application running on top of the controller.

OpenFlow is a protocol originating out of Stanford University, now progressed through efforts of the Open Networking Foundation (ONF), which simplifies and unifies routing and switching. Its goal, when fully implemented, is to provide lower-cost, lower-complexity switches with centralized intelligence via the controllers. This separation of control-plane/data-plane provides a common protocol to program the forwarding plane and the ability to easily program the network using applications on top of the controller.

A SDN network defines three layers: the application layer, the control layer, and the network layer. The applications use an application programming interface (API) to the controller that has a connection to each switch, providing the ability to populate the flow tables of the switch to determine how it matches incoming packets and what actions it should take on the packets.

- The OpenFlow switch specification protocol standard defines the message format as well as the function and behavior of the OpenFlow switch, yet it does not go into much detail about the controller implementation beyond the:
    - OF-Channel – a TCP session with optional encryption, used to communicate with each OpenFlow-enabled device. This consists of controller/switch asynchronous and symmetric messages.
    - Controller/switch messages include a feature capability exchange, packet_out, flow_mod, read_state, and other management functions
    - Asynchronous messages include packet_in, flow-removed, port-status, errors, and others
    - Symmetric messages are primarily the hello and echo

Upon reading through the OpenFlow switch specification, it is clear to see that it primarily defines a command set by which applications can program the network layer. However, it does not describe how some basic functions, like network discovery, should work and if that should be a feature of the controller or an application running on top of the controller. Essentially, the controller is useless without an application or integrated tools, like discovery.

Most open-source controllers that became available had a small set of functions and applications included. Some early examples were NOX, POX, and Beacon. These were followed by additional open-source projects as well as several commercial controllers, including the ProgrammableFlow Controller from NEC.

As commercial SDN controllers are coming to market, they are offering integrated application suites and seamless integration with the switches from the same company or ecosystem of partners.

Testing SDN

Testing of SDN, specifically OpenFlow, started with some open-source tools that were developed with some of the early open-source switch and controller projects. To this point, most of the testing had been focused on the OpenFlow Switch (physical or virtual) since the specification primarily defines what features and behavior the device should have. The testing of the OpenFlow-enabled devices has progressed to a conformance program, currently for v1.0, offered through the ONF, as well as some benchmarking, including Flow-Table capacity, which is also defined by the ONF. Commercial test tools, including those offered by Ixia now provide robust functional and performance testing of OpenFlow-enabled devices. Testing OpenFlow switches has revealed three primary types of implementations:

- **Software-based switches (including vSwitches) –** Typically have high functionality, supporting the majority or all of the optional OpenFlow features, but do not have hardware-based forwarding so the forwarding rate may not be high especially for small packet sizes.
- **Hybrid-based switches –** These are typically the existing routing/switching products that have implemented some OpenFlow functionality. They typically have a limited feature set and may be able to leverage hardware forwarding, yet they have limited forwarding table capacity.
- **Purpose-build switches –** These are typically built for SDN and have comprehensive OpenFlow implementations as well as hardware forwarding that supports advanced functions like packet modifications without degraded performance.

What has been lacking to this point is testing the SDN controller. Operators now beginning to get serious about deploying SDN want to ensure the controller will provide the performance, scale, and features they need for their environment. The only open-source tool in existence is mini-net, which provides switch simulation but this is pretty limited and more suited toward virtualized simulated environments for training and other purposes. Commercial tools are now available from Ixia to test and stress OpenFlow controllers.

Commercial tools are now available from Ixia to test and stress OpenFlow controllers.

Comprehensive testing of an OpenFlow controller will need to cover:

- **Southbound interface –** This is primarily the OF-Channel carrying all the OpenFlow messages
- **Northbound –** This is testing the API to ensure that the programs can leverage the API calls available
- **Applications –** This is ensuring the SDN system, which includes the applications, controller(s) and switches, achieves the desired purpose

We will break this subject matter into multiple parts and focus primarily on the "Southbound" interface and show the following two specific test cases in this Part 1 Case Study:

- Testing OpenFlow Network Topology Discovery
- End-to-End Flow Installation

Methodologies

Testing OpenFlow Network Topology Discovery

Introduction

The network discovery is a very important aspect of an OpenFlow Network. The OpenFlow controller uses packet_out to send LLDP packets to all the connected OpenFlow switches. It uses the packet_in messages from all the switches to map the network topology of all the connected OpenFlow switches with all its connection types. A most commonly-used Fat-Tree topology will be configured in the network and the following key metrics will be measured:

- Controller ability to map the correct network topology (graphical)
- Topology discovery time by Controller

These measurement will be performed for a various number of switch connections to measure the controller performance.

The network discovery is a very important aspect of an OpenFlow Network.

Test Setup

NEC ProgrammableFlow Controller PF6800 (NEC controller) and Ixia test system running IxNetwork with OpenFlow switch emulation performing LLDP-based topology discovery with OpenFlow v1.3. Ixia IxNetwork emulates up to 200 OpenFlow switches in Fat-Tree topology and NEC PFC initiates LLDP-based topology discovery.

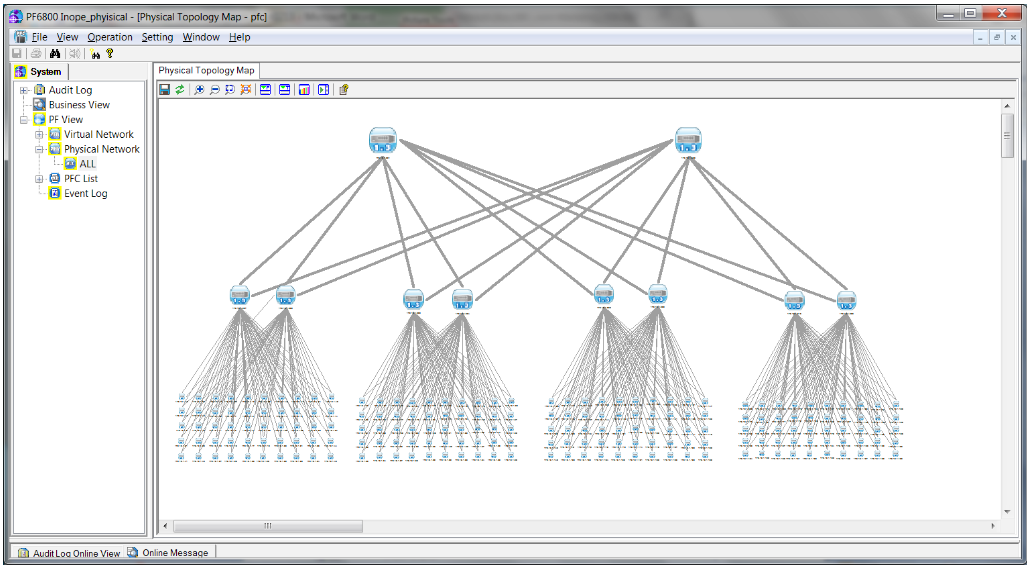
In this test, the following three performances are measured:

- OpenFlow switch's boot-up time
- Total topology discovery time of 200 switches by NEC controller
- Time difference between the above

OFPT_HELLO enables us to estimate the switch's boot-up time since OpenFlow protocol always starts from this message, and the topology discovery time will be measured based on the receiving time of last LLDP packet.

Results

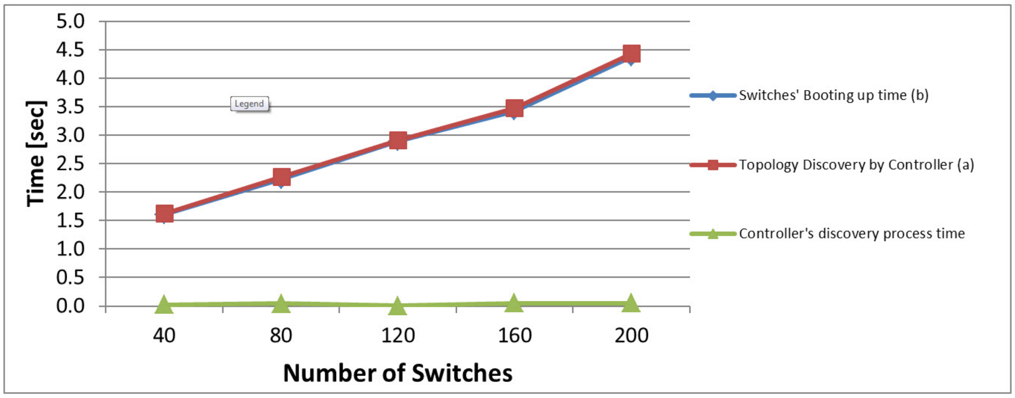
The following figure shows the physical topology discovered by the NEC Controller's GUI. Each of the 200 small icons represents an OpenFlow switch emulated by Ixia and all links between switches are automatically discovered by the NEC Controller.



Each of the 200 small icons represents an OpenFlow switch emulated by Ixia and all links between switches are automatically discovered by the NEC Controller.

The following is the result of the performance testing:

- Boot-up time for the Ixia-emulated switches is less than 100 ms per switch
- NEC Controller discovery time for 200 switches is 4.5 seconds, including switch boot-up time
- NEC Controller time to discover a new switch is 100 ms



End-to-End Flow Installation

Introduction

It is important to measure how fast the controller is able to push the flows to setup the data flows in the OpenFlow network. The network can be provisioned as either L2 or L3 flows. The controller will process the various packet_ins generated by the switch to calculate the best network path and provision the switches with data flows. The following key metrics will be measured:

- End-to-End Flow Installation Rate (ms)
- End-to-End Flow Installation Rate (pps) measurement will be performed for a varied number of data flows to measure the controller flow installation performance

End-to-End Flow Installation Rate is not only OpenFlow's flow setup rate. It includes all the necessary processing required for commercial deployments, including database synchronization for redundancy and application processing time including GUI visualization.

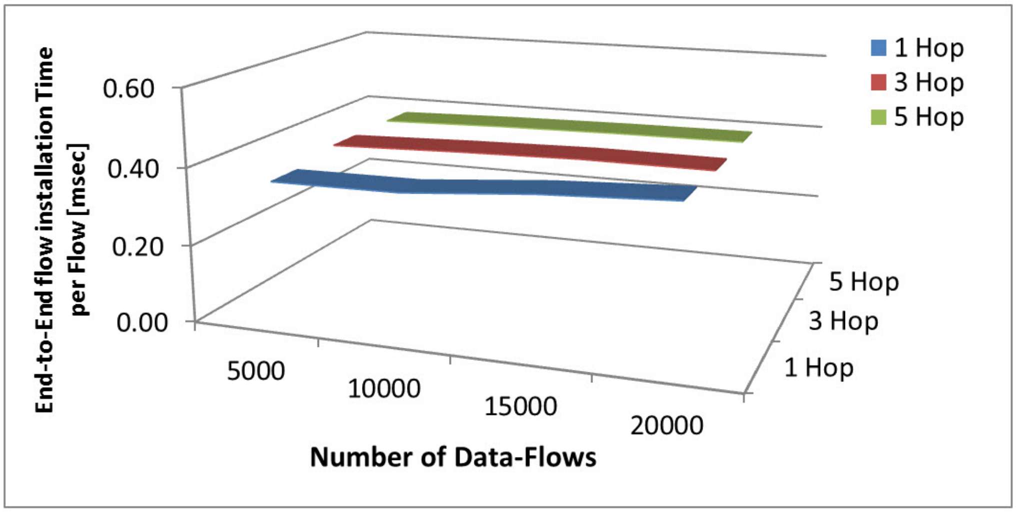
> It is important to measure how fast the controller is able to push the flows to setup the data flows in the OpenFlow network.

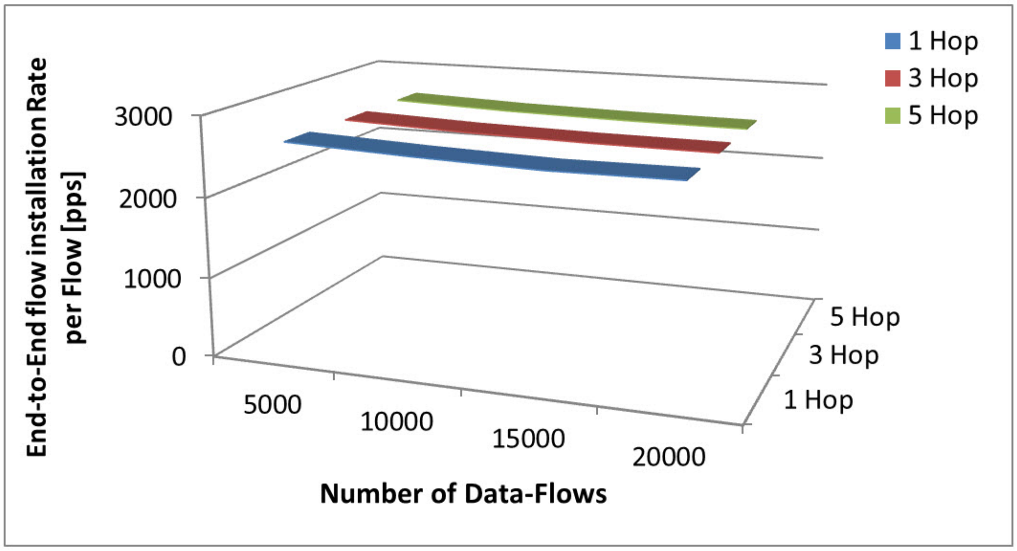
Demo Setup

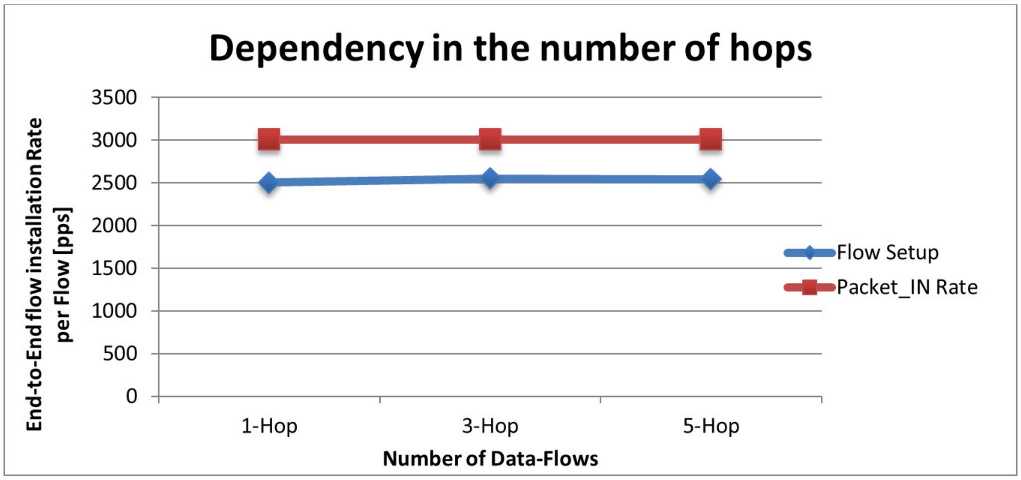
Ixia sends up to 20,000 variations of End-to-End Flow Installation request packets to the controller as OFPT_PACKET_IN message of OpenFlow v1.3, and it simulates 20,000 hosts sending a packet to a host. End-to-End Flow Installation Rate is measured in three different numbers of hops.

Results

The following shows process time to setup each flow by the controller. We observed that the controller can setup a flow in less than 0.5 ms and there is no dependency in the number of hops for flow set up. Please note that this is not only for OpenFlow processing but also includes end-to-end flow installation. More precisely, it includes processing time of applications on the controller and database processing time for controller redundancy.







End-to-End Flow Installation Rate (pps) measurement will be performed for a varied number of data flows to measure the controller flow installation performance.

Conclusion

The goal of this testing is to begin to establish the proof points and metrics to allow consumers of SDN technology to ensure it will meet their requirements of performance, scale, and reliability. This first phase focused on some key functions an OpenFlow controller would need to perform at the desired scale, in this case supporting two hundred OpenFlow switches. Ideally this type of methodology will become standardized within the ONF and can be used for apples-to-apples comparisons when evaluating controller performance.

The first test was focused on a core requirement of an OpenFlow controller, which is to perform topology discovery. The method used (propagating LLDP packets) has become the defacto standard method for OpenFlow. Not only is this used for topology discovery, but in many cases it is also used periodically for monitoring of the network topology to ensure no changes have occurred. This is quite useful since OAM for OpenFlow has not yet been defined.

The goal of this testing is to begin to establish the proof points and metrics to allow consumers of SDN technology to ensure it will meet their requirements of performance, scale, and reliability.

The concern with a large topology is that this requires the controller to send packet_out messages to every node and processing packet_in messages coming from every node for each link in both directions. In this test the key metric was to measure the time it took to process each switch coming up and then discovering the full topology.

Another component of this test was to observe the controller GUI to ensure it was usable with the number of nodes being discovered. Some controllers have not implemented GUI-based topologies with the ability to zoom-in and zoom-out to support large scale. Some also use the full DPID as the identifier of each switch. Using this approach by default does not scale well since it is 16 bytes (larger than a MAC address). At the conclusion of this test it was determined that the NEC controller performed well and could easily handle the advertised scale of supporting two hundred nodes.

The second test exercised a fundamental task of an OpenFlow controller, which is to install flow table entries that enable end-to-end paths. The key metric is measuring how fast the controller can push down flows to setup the data path. Another important factor is to observe if the hop count of the path has a measureable effect on the time. The time measured will be impacted by the application running on the controller and the database used to store the flow table and data path information. Again, this test was performed at scale with two hundred switches. This test demonstrated the end-to-end flow installation took .5 ms and was not affected by the hop count. This was scaled from 5,000 paths to up to 20,000 paths without degradation in performance. The controller was able to handle packet_in rates of 3,000 pps. This type of data shows that the controller is designed to scale and does not show degradation as the number of flow installations is scaled up.

Future phases of testing will expand on test cases for the southbound side (between the controller and switches), as well as testing the North Bound Interface (NBI) and applications running on top of the controller.

For more information on the NEC Controller go to: www.nec.com/sdn

For more information on the Ixia test solution go to: www.ixiacom.com/sdn

Appendix - Detail of Test Environment

Server Spec of PF6800 (ProgrammableFlow Controller)	
Product Name	Express5800/R120b-2
RAM memory	24GB
CPU	Xeon X5690 (12M Cache, 3.46 GHz, 6.40 GT/s Intel® QPI)
OS	Red Hat Enterprise Linux 6.4 (x86_64) (Kernel version: kernel-2.6.32-358.2.1.el6.x86_64)

Ixia Tester	
Chassis	OPTIXIAXM2-02
Module	LSM1000XMVDC8-01
Software	IxNetwork with OpenFlow Switch Emulation Option PN: 930-2105

Ixia Worldwide Headquarters
26601 Agoura Rd.
Calabasas, CA 91302

(Toll Free North America)
1.877.367.4942

(Outside North America)
+1.818.871.1800
(Fax) 818.871.1805
www.ixiacom.com

Ixia European Headquarters
Ixia Technologies Europe Ltd
Clarion House, Norreys Drive
Maidenhead SL6 4FL
United Kingdom

Sales +44 1628 408750
(Fax) +44 1628 639916

Ixia Asia Pacific Headquarters
21 Serangoon North Avenue 5
#04-01
Singapore 554864

Sales +65.6332.0125
Fax +65.6332.0127