Copyright © 2012 Open Networking Foundation

Disclaimer

Contact the Open Networking Foundation at www.opennetworking.org for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.
Contents
1 Introduction .................................................................................................................................. 4
2 Open Networking Foundation (ONF) .................................................................................. 6
   2.1 Testing-Interop Working Group .................................................................................. 6
3 Interoperability Event ........................................................................................................... 7
   3.1 Participants .................................................................................................................. 9
   3.2 Network ....................................................................................................................... 10
   3.3 Basic Testing .............................................................................................................. 11
   3.4 Test Case – Network Discovery LLDP ........................................................................ 13
   3.5 Test Case – Layer 2 Circuit Provisioning .................................................................... 14
   3.6 Test Case – Layer 3 (IP) Learning with Dynamic Provisioning and Failover ............ 15
   3.7 Test Case – Load Balancing ....................................................................................... 17
   3.8 Test Case – L2 MAC Learning .................................................................................... 18
   3.9 Test Case – Slicing the Network with FlowVisor ....................................................... 18
4 Summary and Conclusion .................................................................................................. 19
Appendix A: Revision History ............................................................................................. 20
1 Introduction

OpenFlow and Software Defined Networking (SDN) originated in an academic environment at Stanford and U.C. Berkeley. Much of the early work was done through the Clean Slate program. OpenFlow grew in popularity within universities and research institutions and was even deployed on the Internet2 backbone. The OpenFlow protocol was developed at Stanford, with v1.0 published at the end of 2009 and v1.1 at the beginning of 2011. In March of 2011, the Open Networking Forum (ONF) was created and the intellectual property rights of OpenFlow were transitioned to it. The ONF was tasked at this time to control and commercialize OpenFlow.

An OpenFlow network is made up of two primary components, a controller and a switch, which communicate via the OpenFlow protocol. OpenFlow separates the control plane from the data plane and defines a standard open programming interface, previously not available. The programming interface enables the controller to populate the flow tables of the switch, which determines what it matches on and what action the switch takes on each packet. This open interface enables SDN where software can now customize how networks meet the needs of application and users. Applications are now either built into controllers or talk to a controller via what is referred to as the SDN API, or northbound API. This API is typically well defined by each controller, but not yet standardized.

![Figure 1. SDN vision features applications running on top of a controller using OpenFlow to control packet forwarding](image-url)
OpenFlow v1.0 defined OpenFlow-only switches. OpenFlow v1.1 added the definition of a
OpenFlow-hybrid switch which supports both OpenFlow operation and normal Ethernet
switching operation via traditional L2 Ethernet and VLAN as well as L3 routing, ACL and QoS
processing. Many early implementations from traditional routing/switching products are
OpenFlow-hybrid.

The original application of OpenFlow was to enable testing and test activity on a production
network. The barrier used to be that the network, although based on standard protocols, is run
on proprietary systems that do not offer APIs for development. That was, until OpenFlow.
Once a switch or router enables OpenFlow, then the standard API is available and new and
interesting ways to solve network problems can be developed in software and quickly deployed
on the network. Of course, to prevent breaking the current network, hybrid implementations are
used so that OpenFlow can run on dedicated physical or virtual ports.

The pressure on network equipment vendors to implement OpenFlow was strong and by May of
2011 the Las Vegas Interop show featured an OpenFlow showcase with 14 vendor
implementations, some of which were commercially available.

OpenFlow offers many benefits:

- The potential to reduce CAPEX cost by decreasing the cost and complexity of
  networking hardware
- The potential to reduce OPEX cost by decreasing network management complexity
- The potential to speed up application development and deployment through the use of
  SDN
- A smooth transition from legacy equipment through the use of hybrid devices and the
  ability to work with legacy devices and protocols
- Possibly the greatest benefit of OpenFlow is the flexibility that it has to address current
  and future network problems in many areas of the network, from the enterprise to the
  data center to the service provider edge or core

Example applications of OpenFlow include:

- Traffic engineering
- On-demand path provisioning
- Dynamic path provisioning
- Load balancing
- Distributed firewall
- Extending the network to the virtual switch for VM connectivity
- Enabling test/experimental protocols on the network

OpenFlow provides the command set, or primitives, on top of which controllers, or software
running on top of controllers, implements applications.
2 Open Networking Foundation (ONF)

The ONF publicly launched in March of 2011. The ONF’s mission is to promote the development and use of software-defined networking technologies. One of the ONF’s first tasks was to update and continue to define OpenFlow. At the end of 2011 the ONF produced the OpenFlow v1.2 standard, which is now ratified and vendors are beginning to implement support for it.

The ONF is made up of member companies who join and participate in the working groups. Any company that shares the ONF’s goals is welcome to join. Information is at:

http://www.opennetworking.org

The ONF has five Working Groups including:

- Extensibility
- Hybrid
- Config
- Testing-Interop
- Futures

The ONF has also kicked off a marketing committee that works on promoting the ONF, OpenFlow and SDN.

It is important to note that the ONF does not generate or maintain open-source code. The ONF specifies and publishes standards, starting with the OpenFlow Protocol. It does not create, maintain, support or endorse any open-source code.

2.1 Testing-Interop Working Group

The ONF Testing-Interop Working Group is focused on three key areas:

- Conformance of compliance to ONF standard
- Interoperability
- Benchmarking

Conformance of Compliance

The Testing-Interop Working Group initiates projects to define test suites for certification of compliance for OpenFlow devices. There is a current project open for OpenFlow v1.0 switches. The test suite is currently being defined. In addition to defining the test suites, the Testing-Interop WG will select and work with partner labs that will be approved to conduct ONF certification. This selection process is currently in development. The pilot phase for the v1.0 switch certification is expected to start in the calendar Q2 timeframe and be generally available by Q3. After the v1.0 switch project, it is likely that the next project will be v1.2 and/or v1.3 switches since there are minimal v1.1 implementations.
**Interoperability**

Interoperability of OpenFlow in a multi-vendor environment is critical to its success. This prevents users from getting locked into a single vendor solution and enables them to choose the best solution based on price, performance and features.

Conformance/compliance testing to a standard is not enough. It is critical to test real use-cases of OpenFlow in a multi-vendor environment. While the most critical component is OpenFlow running between a controller and a switch, it is also very important to understand how different switches behave and work together with various applications of OpenFlow.

The Testing-Interop Working Group performs the planning and execution of ONF interoperability events. These events are open to all ONF member companies. They are private events and covered by the NDA in the ONF Member agreement. The participating companies benefit through the ability to test their implementation with other vendors, identify issues, get help from a community of experts and form relationships with peers.

Another benefit of the testing and collaborative work is to ensure that the standard is clearly written and implemented in a standard way. In some cases there may be areas that need clarification or improvement. This type of event can identify weaknesses in the specification.

The conclusion of this event will result in better implementations of OpenFlow and greater confidence for the member companies and their customers. Vendors can also move on to public demonstrations of their products and features after testing them in the private event.

**Benchmarking**

The third area of focus for the Testing-Interop WG is to define standards-based test methodologies to establish performance benchmarks for OpenFlow devices. Methodologies will include testing OpenFlow controllers and switches. Some examples include:

- Flow table capacity of a switch
- Learning rate of an OpenFlow system
- Maximum message rates supported (packet_in from each switch to the controller)
- Maximum flow stats handling
- Forwarding performance benchmarking (throughput, loss, latency)

3 **Interoperability Event**

The March 2012 event was the first-ever ONF interoperability event. It took place March 5th – 9th and was hosted at the Ixia iSimCity lab in Santa Clara, CA. The event was open to all ONF member companies and was planned and managed by the Testing-Interop WG. This event was focused on the v1.0 standard, which the majority of the ONF members have implemented first. Many are skipping v1.1 and moving to v1.2 next.

Although this paper contains a high-level summary, this is considered a non-disclosure event covered under the NDA within the ONF Member agreement. No specific vendor features or
issues will be disclosed in this document. Many of the products tested are not commercially available yet.

The original proposal was to focus on test cases that applied to service provider, data center and enterprise use-cases. Vendors of controllers and applications running on controllers really determine what can be tested. For this event we were able to test:

- Topology discovery (LLDP method)
- Layer 2 Ethernet/VLAN path (circuit) provisioning (primary and backup)
- Layer 3 (IP) learning (shortest path primary and backup path)
- Layer 3 (IP) load balancing
- Enabling multi-controller connectivity using FlowVisor to slice the network

Each one of these applications requires the switches to support the OpenFlow v1.0 protocol.
3.1 Participants
The ONF Testing-Interop events are open to all ONF members and partner research institutions. Participants in the March 2012 event included:

**OpenFlow Controllers**
- Big Switch Networks (Floodlight open-source version)
- NEC
- NTT Data
- NOX (Open-source) with OESS application brought by Indiana University

**OpenFlow Switches**
- Big Switch Networks (Indigo open-source switch)
- Broadcom (Reference design)
- Brocade
- HP
- IBM
- Intel/WindRiver (Reference design)
- Juniper Networks
- NEC

**Test Equipment**
- Ixia
- Spirent

**Research Institutions**
- Indiana University – Using NOX with the OESS application
- Open Networking Lab (Stanford / U.C. Berkeley) – supporting the FlowVisor
3.2 Network
The network used for the interoperability testing was based on standard network design with an access layer and a core layer. The hosts, emulated by test equipment, were connected using 1GE copper. The uplink from the access to the core was 1GE or 10GE, copper or fiber. The interconnection within the core was primarily 10GE but some links were 1GE. All of the OpenFlow switches had an out of band (dedicated port) IP connection to an OpenFlow controller. A single controller was used for each test case until the FlowVisor phase of testing.

Figure 2. Interoperability Event Test Bed
3.3 Basic Testing
Before moving straight into a multi-vendor test case it was important to first perform some basic testing of the OpenFlow 1.0 feature set to ensure functionality. An open-source tool called OFTEST was developed at Stanford alongside the development of the OpenFlow protocol. OFTEST is written in Python and runs on Linux. It emulates an OpenFlow controller and performs functional tests on an OpenFlow switch. There are test cases developed for v1.0 and v1.1. The v1.0 suite is made up of 43 test cases and although it is not comprehensive, it is a good starting point to validate functionality before moving on to more complicated test cases. To run all of the v1.0 test cases OFTEST requires four connections to the OF Switch under test. One connection is for the IP connection over which the OF Channel is established and three Ethernet connections for data plane validation.
Figure 3. Example OFTEST Scenario

Test cases for v1.0 are listed at:

http://www.openflow.org/wk/index.php/OFTestListPage#Test_Cases_for_OpenFlow_1.0

Test results for each switch were provided to the vendor and, if there were issues, experts were available to help explain why a test case failed.

Lessons learned from running OFTEST

- It is not recommended to run OFTEST in a Virtual Machine. This introduces more variables and adds the virtual switch between the VM and the physical network interface cards. This usually causes problems with the data-plane tests.
- OFTEST requires network interfaces cards that support VLAN tagging and VLAN field modification.
- It is highly recommended to turn off all other protocol traffic (STP, IPv6, LLDP...etc.) on the data plane links since they can cause intermittent test failures due to unexpected packets.
- OFTEST is useful to identify OpenFlow features that are not supported.
  - There were some issues with packet_in support; in some cases this was not functioning properly, in others this was not enabled by default.
  - There were some issues with support for test cases requiring data packets to be sent back out on the ingress port. There are work-arounds for this and these will be communicated within the ONF.
- There were a few issues found with OFTEST which have been resolved and checked into the latest code base.
- A basic Linux and Python skill set is required to use OFTEST.
3.4 Test Case – Network Discovery LLDP

The OpenFlow protocol does not specify a network topology discovery mechanism however it has the tools to perform discovery with the support of several OF messages and actions. A commonly implemented discovery mechanism uses the propagation of Link Layer Discovery Protocol (LLDP) packets. There is nothing really specific to LLDP, but it was chosen since it is a standard packet used to discover neighbors. The way it works in OpenFlow is as follows:

- A switch establishes a connection to the controller and responds to a feature_request with a feature_reply enabling the controller to learn the datapath_id (unique to each switch)
- The controller will install a flow entirely using the flow_mod that has a rule to match an LLDP packet and send the packet to the controller over the OF Channel via packet_in
- The controller then sends an LLDP packet containing the datapath_id and port number to the switch instructing it to send out all ports
- When the adjacent switches receive the LLDP packet they will send them up to the controller
- Receiving the LLDP packet back enables the controller to learn connectivity between switches and ensure there is a connection in each direction

![Figure 4. Example Network Map after Running LLDP-based Discovery](image)

Lessons Learned from Testing Topology Discovery

- Each controller implements discovery slightly differently, some requiring manual acceptance of each learned path. Not all of the controllers had graphical depiction of the learned topology. Some controllers would accept unidirectional links.
• Some switches did not support the required feature set, which had to be worked-around or resolved.
• Some switches running in hybrid mode were sending packets by default (like STP, IPv6 Neighbor Discovery and LLDP), causing more traffic to the controller in some cases.
• The use of LLDP from routers and switches combined with LLDP for topology discovery caused some issues and confusion. One consideration is to define an OpenFlow Discovery Protocol packet.
• It cannot be assumed that the default actions of the switches operate as defined in the v1.0 standard.
• It was observed the topology discovery is usually refreshed at a specific interval that was unique to the controller.
• There were other specific compatibility issues that will be addressed within the ONF.

3.5 Test Case – Layer 2 Circuit Provisioning
Layer 2 circuit provisioning is a common task of a large enterprise or service provider. This test case focused on the ability to dynamically-provision point-to-point Layer 2 paths across an OpenFlow network. These endpoints are switch ports, behind which one or more end-hosts are located. Between these endpoints, a primary and secondary (backup) virtual circuit is created.

![Diagram showing Layer 2 Circuit Provisioning Test]

**Figure 5. Example Layer 2 Circuit Provisioning Test**

**Circuit Creation Description**

At the ingress switch, in_port and vlan_id are matched to user-configured values. Matched packets will have their vlan_id translated to the lowest-numbered free vlan_id available on the domain as determined by the application VLAN DB. Intermediate switches will match this new vlan_id and forward the packet on towards the egress switch. When packets arrive at the egress switch, they are matched against the application assigned vlan_id, translated to a user
configured vlan_id and forwarded out the egress port. This is not necessarily the same vlan tag used on ingress.

The secondary path is preinstalled on every switch that is not part of the primary path. It gets activated by ofp_port_status messages if a port along the primary path goes down. For rerouting by path failure, the primary flows are removed and the secondary flows are injected on all non-preconfigured switches. The system will not automatically revert back to the primary path. A failback will occur only by an ofp_port_status message for a port down along the active secondary path.

LLDP-based network discovery is run first to learn the topology before the circuits are provisioned.

Once the circuit was provisioned, test traffic was generated from the emulated hosts to verify the path. After the primary path was verified, failure scenarios were tested and again validated with test traffic.

**Lessons Learned from the Layer 2 Circuit Provisioning Test**

- LLDP topology discovery problems affected this test.
- Lack of support of some required features like packet_in and barrier_reply prevented some switches from working.
- Fail-over scenarios had issues due to different switch default behavior, which included some waiting for a time-out of the flow table entries when the OF Channel was lost. Some hybrid switches defaulted to L2 forwarding upon a failure, which caused flooding.
- There were controllers that installed alternative paths after detecting the path failure.

### 3.6 Test Case – Layer 3 (IP) Learning with Dynamic Provisioning and Failover

After performing network topology discovery as described in test case 3.4, the Layer 3 (IP) dynamic provisioning application performed learning of the emulated IP hosts by forwarding the ARP to the controller for each host. This Controller was then able to run a Dijkstra-based shortest path algorithm and provision a primary path and an alternate path. Once the path is established it can be verified using test traffic from the emulated hosts. Once in a good, working state, the primary path is failed (several options to cause failure). Upon a failure, traffic should be moved to the alternate path by the controller. The data path is validated with test traffic between the emulated Layer 3 hosts.
Lessons Learned from Testing Layer 3 Learning with Dynamic Provisioning

- LLDP topology discovery issues will cause problems in this application
• The ARP must be forwarded to the controller. The controller cannot assume all switches follow the default rule to send unmatched packets to the controller; should install specific flow table entry to match on ARP and send to controller.
• Learning uni-directional paths will also cause issues.
• Failure can be triggered in many ways including loss of OF Channel connection, down port, and unplugged cable.
• Inconsistent failure behavior was observed. Hybrid switches can default to Layer 2 forwarding if they lose the OF Channel connection. Default and failure behavior of each switch should be clearly understood by the controller. Switches can also rely on a timer to clear its flow table, which can slow down the convergence to the alternate path.

3.7 Test Case – Load Balancing

There are two ways to perform load balancing on flows. The first case is static load balancing. Operators can pre-configure routing policy to the Controller (i.e. HTTP traffic is always forwarded via OFSC2 to the destination, but all other traffic to the same destination is not forwarded to OFSC2). The second case is Equal Cost Multi-Path (ECMP). The Controller dynamically calculates the path-cost of the link and each flow is forwarded to the destination via a path depending on the cost. In this case, hashing with MAC/IP addresses add others are used.

![Figure 8. Example of Load Balancing using ECMP](image)

**Lessons Learned from Testing Load Balancing**

• LLDP topology discovery issues will cause problems in this application.
• Due to the discovery issues, multiple paths could not always be determined for load balancing. This was related to issues previously mentioned, including lack of support for barrier_reply and a switch using port 0 in its features_reply message.
3.8 Test Case – L2 MAC Learning

The Layer 2 MAC learning was one of the more basic test cases that essentially turned the OpenFlow network into an Ethernet switch. Once traffic was sent from the emulated hosts, the default switch behavior should be used to send the packet up to the controller. The controller used this information to learn where the hosts were located and then instructed the switch to forward the packet out to all ports (like the flood behavior of an Ethernet Switch). Once the controller learned where the source and destination MAC addresses were, it could install explicit forwarding entries in each switch.

*Figure 7. Example Topology with Controller Performing L2 MAC Learning*

**Lessons Learned from Testing Layer 2 Learning**

- LLDP topology discovery issues will cause problems in this application
- Hybrid switches flooding by default will cause issues with L2 learning and forwarding

3.9 Test Case – Slicing the Network with FlowVisor

FlowVisor is a proxy controller that sits in between the switches and the controllers. It enables slicing of the network in many ways, including based on port, VLAN, MAC, IP, etc. It also enables a seamless connection to multiple controllers, providing the ability to run multiple applications each on a separate controller. The switches think they are connected to a single controller.

The FlowVisor was not added early-on in the testing since it manipulates messages and would have added additional variables to the interoperability tests. A FlowVisor has many applications, one of them being ideal for test environments to virtualize the network and run many concurrent tests.

FlowVisor is developed and maintained by the Open Networking Lab.
Lessons Learned from Testing with the FlowVisor

- The FlowVisor served its purpose and enabled slicing of the network and the ability to run multiple tests in parallel
- Many of the same problems experienced in the earlier test cases will still affect the test when adding the FlowVisor
- One significant benefit in a test environment is that it enables multiple simultaneous tests to run since multiple controllers can be enabled only seeing their “slice” of the network

4 Summary and Conclusion

The goal of this interoperability event was to test applications of OpenFlow in a multi-vendor environment. The benefit to the participants was to gain the experience of working with multiple controllers in each of the use-cases. Although OpenFlow is a protocol between controller and switch and not run between switches, there was significant benefit to learn how the multi-vendor network functioned in each use-case.

Overall the interoperability test event was a success with strong ONF member participation. Vendors participated with released product, open-source product and un-released products. Being the first interoperability event, it was important to first perform basic functional testing, which allowed each vendor to see what areas of the standard they were supporting and what areas they had issues with. This provided them a chance to resolve issues ahead of attempting the use-cases.

The high level goal of the use-cases was to address various markets like the data center, service provider and enterprise networks. Ultimately the applications are limited to what the controller vendors support, or what applications have been developed to run on top of the controller. With the four participating controllers there were some very interesting and useful applications. One common feature each controller used was the defacto standard for network...
topology discovery using LLDP. It was good that each of them has this in common, but the downside was that issues with discovery affected every test.

The goal of the participating vendors was to identify issues, and at the conclusion of the event there were around twenty issues documented. These issues ranged from lack of support for specific features, to interoperability issues related to older implementations, to incorrect default behavior and issues with hybrid implementations. Technical details have been captured in a second document which will be available on the opennetworking.org website. One of the biggest challenges during the event was troubleshooting since there were limited debugging tools built into the controllers and switches for OpenFlow. This resulted in resorting to capture and decode to determine the issue.

Going forward, the expectation is to include newer versions of OpenFlow including the now-released v1.2 and the nearly-completed v1.3. Future testing will depend on what the vendors are implementing. The expectation is to also test additional applications, expanding more into specific data center and service provider use-cases.

**Appendix A: Revision History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>3/20/12</td>
<td>Ixia</td>
</tr>
<tr>
<td>0.2</td>
<td>4/12/12</td>
<td>Updates from NTT, NEC, IBM, Juniper, HP</td>
</tr>
<tr>
<td>1.0 (Final)</td>
<td>4/18/12</td>
<td>Updates from NEC and Ixia</td>
</tr>
</tbody>
</table>