# SDN Protocols, Verification and Troubleshooting

Why did OpenFlow version 1.0 include support for multiple queues per port?

Ports with multiple queue could allow for QoS and prioritisation. Depending on the specifics of a packet, it could get be treated differently. This was originally a hardware feature but because of its effectiveness it was adapted and kept in because of how important it was to make it more likely for OpenFlow to be adapted and supported.

What security is provided for OpenFlow connections between controllers and switches, and why is it not always used?

The amount of TLS channels was limited in 1.0 and non-secure communication could improve throughput. The number of secure channels increased in later versions, but secure communication always incurs a speed penalty. No need in tightly controlled environments.

Give example, explain further.

‘The connection from OpenFlow controller to switch can be in-band or out-of-band’. What does this mean?

“In-Band” means that communication between the switch and controller goes over the same channels, whereas in “Out-Of-Band” there is a special connection for handling that traffic and no other data.

Some switches have dedicated management ports especially in data-centre environments. This would increase the cost of the device.

What is the function of barrier messages in OpenFlow?

Barrier messages allow the controller to instruct a switch to immediately finish all queued requests before proceeding. Sometimes the barrier reply is not true. SDN programmers use barriers as a transactional mechanism, even though it doesn’t fully meet those requirements. To make state of network consistent. Barrier messages might for example follow a batch of requests. Requests and replies.

The OpenFlow specification categorises message types as ‘Symmetric’, ‘Async’ and ‘Immutable’ (among others). What does each of these three categories mean?

Symmetric – goes both ways, in both directions. HELLO

Async – Switch to controller unprompted. PACKET\_IN

Immutable – will not be changed in future versions. Just a subcategory. HELLO

What support does OpenFlow provide for MPLS and VLAN tags?

VLAN tags which are modern and standardized by IEEE a fully supported on multiple levels. OpenFlow allows for popping and pushing of these tags and 1.1 allows modification of the outermost tag. MPLS which is an older labelling system historically used by ISPs is also supported.

How many controllers does OpenFlow allow to be in use in a network at one time?

Prior to 1.2 only one controller per switch was able to be used at the same time but any number of controllers could be active on the same network. 1.2 introduced multiple controllers with different roles. 1.3 allowed a controller to have multiple connections to a single switch. Versions 1.4 enhanced this functionality by allowing different controllers to be responsible for a subset of rules and 1.5 enhanced multiple controller support by enabling the controller to monitor the devices connection to other controllers.

Equal and master – both allow the controller full ability to program the switch, but master enforces that all other controllers must be in slave mode.

Slave – no control, only for gathering data from the device.

How has the OpenFlow protocol evolved over its multiple versions?

Each version of OpenFlow improves on the previous one.

* 1. improves on failure handling and multiple flow tables.
  2. Implements multiple controller support and bridging VLANs.
  3. LTS Version including enforcing QoS and Auxiliary Connections.
  4. Bundles and Optical Port support.
  5. Adjacent discovery and improved interoperability.

Against what parts of a packet does OpenFlow allow matches?

12-tuple for 1.0. Later versions allow to match to any part of the packet.

Fields That Can Be Used

* Switch input port
* VLANID
* VLAN priority
* Ethernet source address
* Ethernet destination address
* Ethernet frame type
* IP source address
* IP destination address
* IP protocol
* IPType of Service (ToS) bits
* TCP/UDP source port
* TCP/UDP destination port

What are the purposes of these OpenFlow messages: FLOW\_MOD, PORT\_STATUS, PACKET\_IN, PACKET\_OUT?

FLOW\_MOD controller adds, deletes or modifies flow entries in a switch.

PORT\_STATUS is used by switch to communicate changes on switch ports to a controller. e.g. port going up or down.

The PACKET\_IN message is a way for the switch to send to the controller a data plane packet which either does not match any rules, or the rules specify to send this packet to the controller.

The controller can inject packets into the data plane of a switch. It does this with the PACKET\_OUT message, which can either carry a raw packet to inject into the switch, or in 1.0 indicate a local buffer on the switch containing a raw packet to release.

The OpenFlow specification defines ‘meters’. How are these useful in a network?

Meter: a switch element that can measure and control the rate of packets. The meter triggers a meter band if the packet rate or byte rate passing through the meter exceeds a predefined threshold. If the meter band drops the packet, it is called a Rate Limiter.

How does OpenFlow fit in the context of legacy protocols such as IEEE 802.3 and 802.1Q?

OpenFlow does not impact Ethernet, it coexists with it. They are on different layers. Ethernet could be used to carry OpenFlow messages. OpenFlow allows for rules to match against Ethernet header fields.

OpenFlow also allows to match against VLAN tags, and the pushing, popping and modification of VLAN tags.

In the context of OpenFlow, what are ‘standard’ and ‘reserved’ ports?

The OpenFlow standard ports are defined as physical ports, logical ports, and the LOCAL reserved port if supported (excluding other reserved ports). Standard ports can be used as ingress and output ports, they can be used in groups, and they have port counters.

Reserved - They specify generic forwarding actions such as sending to the controller, flooding, or forwarding using non-OpenFlow methods, such as “normal” switch processing.

The port numbers that are reserved to identify functions: ALL, CONTROLLER, TABLE, IN\_PORT, LOCAL, NORMAL, FLOOD.

Why is verification important on a Software-Defined Network, and what tools are available?

Software has bugs and networks may be misconfigured. On a large-scale deployment this almost guarantees that stuff won’t work on the first try. It’s better to verify the configuration rather than find out in production.

What approach should be taken to troubleshooting a Software-Defined Network, and what tools are available?

SDNs are composed of networking and programming, therefore troubleshooting any problems might require skill in both. One might gather symptoms, analyse information gathered and possible identify causes. The next step would be eliminating possible causes to identify real cause and putting together a solution and implementing and documenting it.

IDE and debugger for whatever language is used. Wireshark and Ping Trace Route are examples of networking tools that could be used for debugging.

An OpenFlow controller just received an OFPT\_FLOW\_REMOVED message from a switch. In what situations does this happen?

Messages are generated only if a flow was removed that had the OFPFF\_SEND\_FLOW\_REM flag set. This could be because the controller specifically requested this through a FLOW\_MOD. The idle-timeout or hard-timeout timer expiring could also cause this.

# SDN vSwitches and Controllers

In a flow-rule entry in a switch’s flow-table, what is the purpose of the IDLE\_TIMEOUT value?

IDLE\_TIMEOUT defines the time after which a rule should be dropped. This, in contrast to hard timeout counts the time since the rule was last used, as opposed to when it was instated. The rule will be dropped when either of those timers runs out, whichever does it sooner. If the idle timeout is set to zero, it will not timeout due to idleness.

In SDN, what is the difference between a southbound and a northbound interface? Give examples of each.

A northbound interface is an API available to higher-level components. It sits between the controller and SDN applications. Southbound is an interface available to lower-level components and sits between the controller and switches. OpenFlow is an example of a southbound interface and it defines the way the SDN controller should setup and adapt the network. The northbound interface is often RESTful and there are efforts to standardise an intent-based API. The client-side of the northbound interface can be implemented by applications like SaltStack.

From where does a controller receive information that triggers changes to flow-rules?

The controller might receive information from external sources like firewalls or intrusion detections systems or network managements systems. The controller also makes changes to flow rules based on the information that comes from the devices in the network in PACKET\_IN messages.

If an SDN switch loses the connection to its controller, what are the possible options for how the switch behaves?

An SDN switch can be programmed with Fail Secure or/and Fail Standalone modes.

Fail Secure mode continues to forward packets but drops all messages for the controller and Fail Standalone mode continues to operate in native switch or router mode and OpenFlow pipeline processing stops.

What are the advantages and disadvantages of a RESTful interface to an SDN controller?

REST is very popular and widely known. It is easy to program in any major language. Data is not tied to resources or methods like RMI. Can transfer data in various formats like JSON or XML. REST on HTTP cannot pre-emptively send data and is limited to one session between client and server. This changes with SPDY and HTTP2.0.

What is ‘Intent-Based Networking’?

Intent based networking is concept that aims to replace manual configuration of networks with a system where administrators provide the controller with the outcome they want, and the network achieves that state by itself through artificial intelligence and machine learning.

In the context of SDN, what are the differences between a software switch and a hardware switch?

A software switch runs on general purpose hardware while a hardware switch runs on purpose-built hardware. Hardware switches are meant for high port-density for end-users/phones/servers that a virtual switch cannot match unless everything is already virtual.

‘A network administrator should use either all software switches or all hardware switches in their organisation’s network, but not both’. Do you agree? Explain your answer.

No idea.

What functions must an SDN controller provide?

An SDN controller manages flow control to the switches/routers “below” (via southbound APIs) and the applications and business logic “above” (via northbound APIs) to deploy intelligent networks. They consolidate and mediate between different controller domains using common application interfaces.

Why does an SDN controller need to keep a flow-rule cache?

A flow-rule cache defines the action that should be taken in the event of a message arriving at a port from an address. It is instrumental in defining all the operations of a network.

A network manager you know is planning to use SDN applications running on a controller to replace some expensive hardware appliances. Give your advice on how she should proceed.

If hardware is already in place, then maybe it’s still faster. Does the network change much? Is there need for a central controller?

Compare two SDN controllers.

ONOS vs ODL.

Both written in Java.

In the event of a fault, ONOS does eventual consistency and ODL does high availability.

Scaling of both eventually leads to diminishing returns as number of activities and syncing increases. ONOS guarantees all nodes have the same data.

Both in Linux Foundation umbrella, more support for ODL.

There are several SDN controllers currently available. What lead to there being more than one? How do you expect the situation to develop in the next few years, and why?

No idea.