Points to mention:

* Southbound protocols supported by ONOS. <https://wiki.onosproject.org/display/ONOS/Southbound+protocols?src=contextnavpagetreemode>

<https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networking-controller>

An SDN controller is an application in a software-defined networking (SDN) architecture that manages flow control for improved network management and application performance. The SDN controller platform typically runs on a server and uses protocols to tell switches where to send packets.

SDN controllers direct traffic according to forwarding policies that a network operator puts in place, thereby minimizing manual configurations for individual network devices. By taking the control plane off of the network hardware and running it instead as software, the centralized controller facilitates automated network management and makes it easier to integrate and administer business applications. In effect, the SDN controller serves as a sort of operating system (OS) for the network.

The controller is the core of a software-defined network. It resides between network devices at one end of the network and applications at the other end. Any communication between applications and network devices must go through the controller.

The controller communicates with applications -- such as firewalls or load balancers -- via northbound interfaces. The Open Networking Foundation (ONF) created a working group in 2013 focused specifically on northbound APIs and their development. The industry never settled on a standardized set, however, largely because application requirements vary so widely.

The controller talks with individual network devices using a southbound interface, traditionally one like the OpenFlow protocol. These southbound protocols allow the controller to configure network devices and choose the optimal network path for application traffic. OpenFlow was created by ONF in 2011.

**Pros and cons of SDN controllers**

The SDN controller resides between the application layer and infrastructure layer.

One major benefit of SDN controllers is that the centralized controller is aware of all the available network paths and can direct packets based on traffic requirements. Because of the controller's visibility into the network, it can automatically modify traffic flows and notify network operators about congested links.

Companies can -- and should -- use more than one controller, adding a backup for redundancy. Three seems to be a common number among both commercial and open source SDN options. This redundancy will enable the network to continue running in the event of lost connectivity or controller susceptibility.

The controller acts as a single point of failure, so securing it is pivotal to any software-defined network. Whoever owns the controller has access to the entire network. This means network operators should create security and authentication policies to ensure only the right people have access.

**SDN controller vendors**

Vendors that offer SDN controllers include the following:

Big Switch Networks

Cisco

Cumulus Networks

Hewlett Packard Enterprise

Juniper Networks

Nuage Networks

Pica8

Pluribus Networks

VMware

Open source SDN controllers:

SDN controllers are available in a variety of open source options, including:

Ryu

Floodlight

Open Network Operating System (ONOS)

OpenDaylight (ODL)

OpenContrail

Also visit: <https://en.wikipedia.org/wiki/List_of_SDN_controller_software>

SD-WAN controllers:

Traditionally, SDN controllers are used in data center networks. As SDN technology evolved, however, the WAN became a compelling use case, driving the growth of software-defined WAN (SD-WAN) technology. An SD-WAN controller performs many of the same duties as an SDN controller, following policy configurations to direct WAN traffic over the most efficient route. The SD-WAN market has fewer notable open source options than SDN, as most SD-WAN controllers typically come tied together with the vendor's proprietary SD-WAN platform.

https://wiki.onosproject.org/display/ONOS/The+Network+Configuration+Service

The Network Configuration Service also provides the ability to add information into ONOS's network view. This makes it possible to write a program that reads the device and topology inventory, as well as other information, from a site-specific database and then provides it directly to ONOS, without having to rely on dynamic device and topology discovery. Additionally it may be used to add supplemental information that ONOS does not or cannot discover automatically, but may require for correct operation.

<https://wiki.onosproject.org/display/ONOS/NETCONF>

A picture containing diagram

Description automatically generated

Table

Description automatically generated

Q: Some of the traffic that I’d expect my OpenFlow controller to see doesn’t actually appear through the OpenFlow connection, even though I know that it’s going through.

A: By default, Open vSwitch assumes that OpenFlow controllers are connected “in-band”, that is, that the controllers are actually part of the network that is being controlled. In in-band mode, Open vSwitch sets up special “hidden” flows to make sure that traffic can make it back and forth between OVS and the controllers. These hidden flows are higher priority than any flows that can be set up through OpenFlow, and they are not visible through normal OpenFlow flow table dumps.

Usually, the hidden flows are desirable and helpful, but occasionally they can cause unexpected behavior. You can view the full OpenFlow flow table, including hidden flows, on bridge br0 with the command:

$ ovs-appctl bridge/dump-flows br0

to help you debug. The hidden flows are those with priorities greater than 65535 (the maximum priority that can be set with OpenFlow).

The **Documentation/topics/design** doc describes the in-band model in detail.

If your controllers are not actually in-band (e.g. they are on localhost via 127.0.0.1, or on a separate network), then you should configure your controllers in “out-of-band” mode. If you have one controller on bridge br0, then you can configure out-of-band mode on it with:

$ ovs-vsctl set controller br0 connection-mode=out-of-band

Diagram

Description automatically generated

Intent Compilation

Table

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text

Description automatically generated