chapter 1

SDN overview

2020-04-23

# what is SDN - the history

## Network device evolution

Since early 1990 network device manufacturer made a lot of innovation in order to increase router speeds. They started from a router node in which everything was computed into the central CPU to reach a situation where the central CPU is less and less used due to a distributed architecture in which lots of action are done in “line cards”.

![image](data:image/png;base64;base64,)

These progresses have been made thanks to the use of proprietary ASICs (Application-Specific Integrated Circuit), TCAM (Ternary Content-Addressable Memory) which have been designed to process data packets at high speed.

In early 2000, the Virtualization for x86 computers support has led to lots of innovation into systems domain. Compute virtualization and High-Speed network devices evolution have enabled the **Cloud** creation.

Later, It appears it was not convenient to manage several isolated network devices having each their own configuration language. Following needs have emerged:

* Single point of configuration
* Configuration protocol standardization
* Network feature support on x86 servers
* Extensibility and ability to scale

And these desires called for the cloud and SDN technology development.

## Early age of SDN

In Stanford University (US - CA) Clean Slate Research Projects program has been initiated in order to think about how to improve the Internet network architecture. ETHANE project was part of this program. Its purpose was to “ Design network where connectivity is governed by high-level, global policy”. This project is generally known as the first implementation of SDN:

In 2008, a white paper has been proposed by ACM (Association for Computing Machinery) to design a new protocol (OpenFlow) to be able to program network devices from a network controller.

In 2011, ONF (Open Networking Foundation) has been created to promote SDN Architecture and OpenFlow protocols.

## SDN startups acquired by major networks or virtualization vendors

First companies working on SDN have been founded around 2010. Most of them have now been bought by main networks or virtualization solution vendors.

In 2007, Martin Casado, who was working on Ethane project has founded Nicira to provide solutions for network virtualization with SDN concept. Nicira has been aquired by vMware in 2012 to develop VMare NSX. In 2016, VMWare also bought PLUMGrid a SDN startup founded in 2013.

In 2010, BigSwitch networks has been founded: BigSwitch is proposing a SDN solution. In early 2020, BigSwitch has been acquired by Arista Networks.

In 2012, Cisco has created Insieme Networks, a spin-in start-up company working on SDN. In 2013, Cisco take back control on Insieme in order to develop its own SDN solution called ACI (Application Centric Infrastructure).

In early 2012, Contrail Systems Inc has been created and aquired at the end of the year by Juniper Networks.

In 2013, Alcatel Lucent has created Nuage Networks, a spin-in start-up company working on SDN. Nuage Networks is now an affiliate of Nokia.

The road of SDN development and its history is never straighforward and looks more nuanced than a single storyline might suggest, it’s actually far more complex to be described in a short section here. This diagram from [[sdn-history]](#sdn-history) shows developments in programmable networking over the past 20 years, and their chronological relationship to advances in network virtualization.

![sdn-history](data:image/png;base64;base64,)

* <https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf>
* <http://yuba.stanford.edu/cleanslate/research_project_ethane.php>
* <http://yuba.stanford.edu/ethane/pubs.html>
* <https://dl.acm.org/doi/10.1145/1355734.1355746>

# SDN definition

## What is SDN?

The concept of SDN, and the term itself, are both very broad and often confusing. There is no real accurate definition of SDN, and vendors usually take it very differently. Initially it was used to in Stanford’s OpenFlow project, and later it has been extended to include a much wider area of technologies. Discussion about each vendor’s SDN definition is beyond the scope of this book. but we generally consider that a SDN solution has to provide one to several of following characteristics:

* a network control and configuration plane split from the network dataplane.
* a centralized configuration and control plane (SDN controller)
* a simplified network node
* network programmability to provide network automation
* automatic provisioning (ZTP zero touch provisioning) of network nodes
* virtualization support and openness

According to [[onf-sdn-definition]](#onf-sdn-definition), **Software-Defined Networking (SDN)** is:

The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices

![SDN layer[onf-sdn-definition]](data:image/png;base64;base64,)

SDN layer[[onf-sdn-definition]](#onf-sdn-definition)

In this diagram, you can see that SDN allows simple high-level policies in the "application layer" to modify the network, because the device level dependency is eliminated to some extent. Now the network administrator can operate the different vendor-specific devices in the "infrastructure layer" from a single software console - control layer. The controller in control layer is designed in such a way that it can view the whole network globally. This controller design helps a lot to introduce functionalities or programs as they just needs to be talk to the centralized controller. All details communicating with each device is hidden from the applications.

Several expectations are behind this new model:

* **cost reduction**: using standardized network nodes. The costly part of the network equipment (CPU) beeing moved and shared onto a central node.
* **openness**: using some standardized protocols like REST, OpenFlow, XMPP, NetConf
* **automation**: through the API interfaces provided by the SDN controller.
* **features rich**: with the ability of the SDN Controller to reprogram each controlled device using flow tables

in this diagram, "openflow" is marked as the protocol between control layer and infrastructure layer. This is to give an example about the "south bound" interface. As of today there are more choices available and standardized in the SDN industry, which will be covered later in this chapter.

* <https://www.opennetworking.org/sdn-definition/>
* <https://www.rfc-editor.org/rfc/rfc7426.txt>

## Traditional Network Planes and SDN layer

**traditional network device planes.**

traditionally, A typical network device (e.g. a router) has following planes:

![traditional network device planes](data:image/png;base64;base64,)

traditional network device planes

* **Configuration** (and management) **plane**: used for network node configuration and supervision. Widely use protocols are CLI (Command Line Interface), SNMP (Simple Network Management Protocol) and NetConf.
* **Control plane**: used by network nodes to take packet forwarding decision. In traditional networks most widely used network control protocols are OSPF, ISIS and BGP for IP protocol and LDP; RSVP-TE for MPLS.
* **Forwarding** (or data or user) **plane**: This plane is responsible to perform data packet processing and forwarding. This forwarding plane is made of proprietary protocols and is specific to each network equipment vendor.

First two planes (configuration and control) are located into router main processor card. The last one is located into the router line cards.

**SDN layer.**

SDN architecture is built with 3 layers:

![SDN architecture](data:image/png;base64;base64,)

SDN architecture

* **Application Layer**: is containing all the application provided by the SDN solution. Generally a Web GUI dashboard is the first application provided to SDN users. Other very common applications are Network infrastructure interconnection interfaces allowing the SDN solution to be plugged to a Cloud Infrastructure or a Container orchestrator.
* **Control Layer**: is containing the SDN controller. This is the smartest part of a SDN solution. The SDN controller is made up of:
  + one or several Northbound interfaces that are used to interconnect SDN application with the SDN infrastructure. The most used northbound interface protocol is HTTP REST.
  + one or several Southbound interfaces that are used to control SDN network nodes. Most used southbound interface protocols are OpenFlow and XMPP.
  + the SDN engine, made up of SDN Control Logic and some databases.
* **Infrastructure Layer**: is containing the SDN network nodes. This is the working part of a SDN solution. SDN network nodes are either physical or virtual nodes. On each SDN node are located:
  + a SDN agent: which is handling the communication between each SDN network node and the SDN controller.
  + A flow/routing information table filled by the SDN Agent.
  + A forwarding plane engine

## the primary changes between SDN and traditional networking

In a traditional infrastructure, the route calculation is made on each individual router. Routing path is the result of routing information exchange, and of a distributed calculation.

![Component in a traditional router](data:image/png;base64;base64,)

Component in a traditional router

Traditional networks are very robust but very hard to manage due to the high number of points to configure. Traditional network nodes are requiring expensive components because they are implementing high end routing protocols.

Control and Configuration functions are gathered into a "SDN controller" which is controlling SDN Network devices. This new architecture intends to provide a new way to configure the network using a centralized configuration and control point.

New Cloud infrastructures are requiring:

* a single configuration point
* the ability to distribute at a higher scale network elements, at least in each Cloud compute, and not only at the network infrastructure level.
* a simplified network node in order to be able to implement it into each compute node.

In order to get a single configuration point, a centralized network controller is proposed by the SDN Architecture. In order to be able to simplify network nodes, the smartest part has been moved onto a controller.

![Comparison between tradition network devices and SDN devices](data:image/png;base64;base64,)

Comparison between tradition network devices and SDN devices

A southbound network protocol is the last piece needed to allow routing information between the SDN controller and each controlled element. A network infrastructure is allowing the communication between SDN controller and SDN network nodes, and data packet transfer between SDN nodes. This underlay network infrastructure is playing the same role that the local switch fabric is doing inside a standalone router between the control processor card and lines cards.

In a SDN infrastructure route calculation is done centrally onto the controller and distributed into each SDN network node. It makes the controller the weakest point of this new kind of infrastructure.

Lots of efforts are done by each SDN solution supplier to make this centralized point:

* highly resilient: using clustered architecture to build the controller
* highly scalable: using distributed compute and storage architectures

## underlay vs overlay

**underlay.**

In SDN architecture, each network node is connected to a physical network infrastructure. This physical network which is providing connectivity between network nodes is called the underlay network infrastructure.

**overlay.**

Today the industry began to shift in the direction of building L3 data centers and L3 infrastructures, mostly due to the rich features coming from L3 technologies, e.g, ECMP load balancing, flooding control, etc. However, the L2 traffic does not disappear and most likely it never will. there are always the desire that a group of network users need to reside in the same L2 network, typically a VLAN. However, In today’s virtualization environment, a user’s VM can be spawned in any compute located anywhere in the L3 cluster. Even if 2 VMs happen to be spawned in the server, there is often a need to move them around without changing their networking attributes. These requirements to make a VM always belonging to the "same VLAN" calls for an overlay model over the L3 network. In other words, this new mechanism needs to allow you to tunnel L2 Ethernet domains with different encapsulations over an L3 network.

The overlay network is a logical network that runs on top of the underlay L3 IP network. The overlay is formed of tunnels to carry the traffic across the L3 fabric. The underlay also needs to separate between different administrative domains (tenants), switch within the same L2 broadcast domain, route between L2 broadcast domains, and provide IP separation via VRFs.

![image](data:image/png;base64;base64,)

Indeed, without such an encapsulation mechanism, traditional segmentation solutions (VLAN, VRF) would have to be provided by the physical infrastructure and implemented up to each SDN node, in order to provide an isolated transportation channel for each customer network connected to the SDN infrastructure.

Encapsulation protocols used in SDN networks have to provide:

* network segmentation: ability to build several different network connectivity between 2 SDN network nodes.
* ability to carry transparently Ethernet frames and IP packets
* ability to be carried over an IP connectivity

Several encapsulation protocols are used into SDN networks; they are:

* VxLAN
* Geneve
* STT
* NVGRE
* MPLS over GRE
* MPLS over UDP

These encapsulation protocols are providing Overlay connectivity which is required between customers workload connected to the SDN infrastructure.

![image](data:image/png;base64;base64,)

Each SDN node is call a VTEP (Virtual Tunnel End Point) as it is starting and terminating the overlay tunnels.

## interfaces between layers

We’ve seen "openflow" marked as one of the possible interfaces in the "SDN layer" section. Now we’ll introduce the concept of "southbound" and "northbound" interface and other available choices in today’s industry.

**southbound interface.**

The "southbound" interface resides between the controller in "control layer" and network devices in "infrastructure layer". Basically what it does is to provide a means of communication between the 2 layers. Based on the demands and needs, a SDN Controller will dynamically changes the configuration or routing information of network devices. For example, a new VM will advertise a new subnet or host routes when it is spawned in a server, this advertisement will be delivered to SDN controller via a southbound protocol. Accordingly, SDN controller collects all routing updates from the whole SDN cluster, decides the most current and best route entries and it may "reflect" these information to all other network devices or VMs. this ensures all devices will has the most uptodate routing information in real time. the two most well-known southbound interface in the industry is openflow and OVSDB.

**openflow.**

OpenFlow is one of the most widely deployed southbound standard from open source community. It first made its appearance in 2008 by Martin Casado at Stanford University. The appearance of OpenFlow was one of the main factors which gave birth to Software Defined Networking.

OpenFlow provides various information for the Controller. It generates the event-based messages in case of port or link changes. The protocol generates a flow based statistic for the forwarding device and passes it to the controller.

OpenFlow also provides a rich set of protocol specifications for effective communication at the controller and switching element side. Open Flow provides an open source platform for Research Community.

Every physical or virtual OpenFlow-enabled switch in the SDN domain needs to first register with the OpenFlow controller (OpenDaylight). Our Mininet switches, which are actually OVS switches, will then be registered with OpenDaylight and will pop up in the OpenDaylight topology view instantly.

The registration process is completed via an OpenFlow HELLO packet originating from the OpenFlow switch sent, to the SDN controller. This controller (OpenDaylight) can authenticate the request and check whether the switch is allowed to be part of OpenDaylight’s SDN domain.

Remember, openflow is not the only choice for the southbound interface.

**OVSDB.**

unlike openflow, OVSDB is a southbound API designed to provide additional management capabilities like networking functions. With OVSDB we can create the virtual switch instances, set the interfaces and connect them to the switches. We can also provide the QoS policy for the interfaces.

**northbound interface.**

The northbound interface provides connectivity between the controller and the network applications running in management plane. As we already discussed that southbound interface has OpenFlow as open source protocol, northbound lacks such type of protocol standards. However with the advancement of technology now we have a wide range of northbound API support like ad-hoc API’s, RESTful APIs etc. The selection of northbound interface depends on the programming language used in application development.

## SDN, openstack, NVF and data center

**openstack.**

OpenStack is one of the IaaS open source implementation solutions, providing basic services like computing service, storage service, networking service, etc. It also provides advanced services like database, container orchestration and other advanced services. SDN, and its ecology, in contrast, mainly focus on the networking. Therefore, from the perspective of technical ecological coverage, the ecological aspects of OpenStack are much wider, because networking is just one of its services that is implemented by its Neutron component and it’s various plugins.

**NFV: Networking Function Virtualization.**

NFV/VNF sounds like new buzzwords, but those technologies have been around for years. NFV means "network function virtualization", according to ETSI it stands for an "operation framework for orchestrating and automating VNFs". And VNF means "virtualized network function", such as virtualized routers, firewalls, load balancers, traffic optimizers, IDS or IPS, web application protectors, and so on. When you read today’s documents about virtualization technology, you will see the terms in such a pattern like "vXX" (e.g. vSRX) very often. that letter v indicates it is a "virtualized" product. Among others, firewalls and load balancers are the two most common VNFs in the industry, especially for deployments inside data centers.

**data center.**

Flexibility is the main driver for any visualization platform. The data center network itself is also part of the virtualization revolution. SDN and network overlays are the key drivers for virtualizing networks in data centers.

* <https://portal.etsi.org/NFV/NFV_White_Paper.pdf>

# SDN Dataplane

## kernel

## dpdk

## sriov

## smartnic

## vDPA

## eBPF

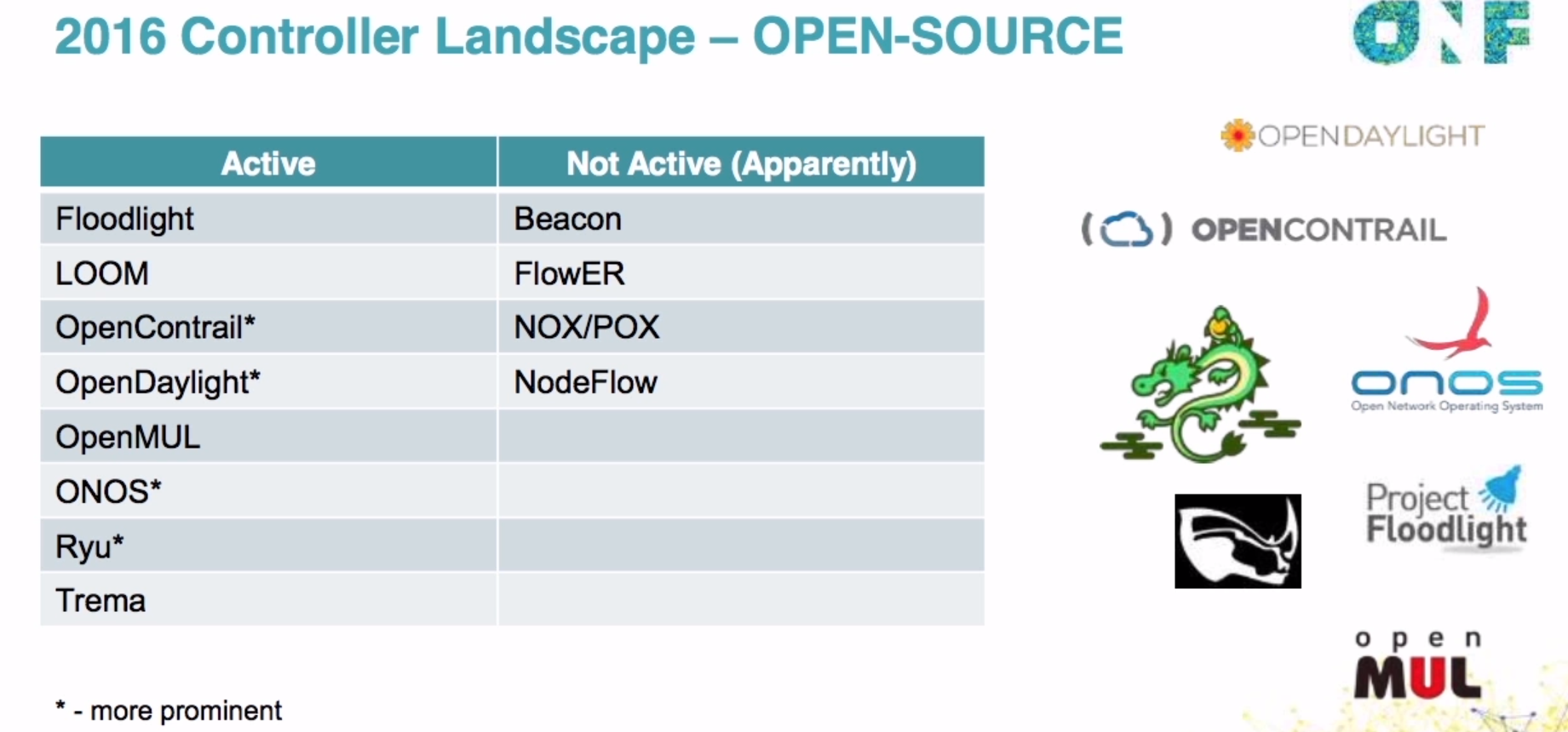
# SDN solutions

## controllers

As we’ve mentioned in previous sections, SDN is a networking scenario which changes the traditional network architecture by bringing all control functionalities to a single location and making centralized decisions. SDN controllers are the brain of SDN architecture, which perform the control decision tasks while routing the packets. Centralized decision capability for routing enhances the network performance. As a result, SDN controller is the core components of any SDN solutions.

While working with SDN architecture, one of the major point of concerns is which controller and solution should be selected for deployment. There are quite a few SDN controller and solutions implementations from various vendors, and every solution has its own pros and cons along with its working domain. In this section we’ll review some of the popular SDN controllers in the market, and the corresponding SDN solutions.

## SDN controller reports



TODO, some research about today’s market players, may skip

* [2015](https://www.sdxcentral.com/wp-content/uploads/2015/08/SDxCentral-SDN-Controllers-Report-2015-B2.pdf)
* [2016](https://www.opennetworking.org/images/stories/downloads/sdn-resources/special-reports/Special-Report-OpenFlow-and-SDN-State-of-the-Union-B.pdf)
* [Controllers in SDN: A Review Report. 2018](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8379403)

## opendaylight (ODL)

OpenDaylight, aften abbreviated as ODL, is a Java based open source project started from 2013, it was originally led by IBM and Cisco but later hosted under the Linux Foundation. it was the first open source Controller that can support non-OpenFlow southbound protocols, which can make it much easier to be integrated with multiple vendors.

ODL is a modular platform for SDN. It is not a single piece of software. It is a modular platform for integrating multiple plugins and modules under one umbrella There are many plugins and modules built for OpenDaylight. Some are in production, while some are still under development.

![opendaylight "Boron"](data:image/png;base64;base64,)

opendaylight "Boron"

Some of the initial SDN controllers had their southbound APIs tightly bound to OpenFlow, But as we can see from the diagram, besides openflow, many other southbound protocols that are available in today’s market are also supported. Examples are NETCONF, OVSDB, SNMP, BGP, etc. Support of these protocols are done in a modular method in the form of different plugins, which are linked dynamically to a central component named "Service Abstraction Layer (SAL)". SAL does translations between the SDN application and the underlaying network equipments. for instance, when it receives a service request from a SDN application, typically via high level API calls (northbound), it understands the API call and translates the request to a language that the underlying network equipments can also understand. That language is one of the southbound protocols.

While this "translation" is transparent to the SDN application, ODL itself needs to know all the details about how to talk to each one of the network devices it supports, their features, capabilities etc. a topology manager module in OLD manages this type of information. What topology manager does is to collect topology related information from various modules and protocols, such as ARP, host tracker, device manager, switch manager, OpenFlow, etc, and based on these info, it visualize the network topology by drawing a diagram dynamically, all the managed devices and how they are connected together will be showed in it.

![old topology](data:image/png;base64;base64,)

newly added devices will be added in the database and reflected immediately in the diagram. Remember earlier we mentioned that an SDN controller has "global view" of the whole SDN network. In that sense ODL has all necessary visibility and knowledge of the network that can be used to draw the network diagram in realtime.

* <https://www.opendaylight.org/technical-community/getting-started-for-developers/roadmap>
* <https://www.opendaylight.org/what-we-do/current-release/boron>
* <https://www.sdnlab.com/community/article/odl/1>

## mininet

Mininet creates a realistic virtual network that runs a real kernel, a switch, and application code on a single machine (VM, cloud, or native), in seconds, with a single command. It normally is in the form of a virtual machine that can be loaded on your hypervisor, and it has its own tiny operating system with a single CLI command. It is a tool to create OpenFlow virtual switches and hosts. With Mininet, you can also create virtual hosts that could be connected to the virtual ports of virtual switches; once it connects them, it assigns them with IP addresses as well. It’s a complete all-in-one test tool to use with OpenDaylight.

## overlay SDN solution (VN)

## ONOS

## calico

## nuage (Nokia)

## OVN

### OVS

### OVN

## contrail (brief)

## vmare NSX

## others

### cisco: apic

### openflood

### opendaylight

# Overview of Tungsten Fabric

## Openstack integration (brief)

## Neutron

## Nova

# summary

# resources