HOW CAN MY NETWORK BENEFIT FROM SDN?

With the flood of solutions and products that are promoted under the name Software Defined Networks one will inevitably ask the question "What's in it for me? How can my network benefit from SDN?" And the answer, as is mostly the case, is: "It depends." SDN is different things to different people and different organizations.

OVERVIEW

Commonly SDN is defined as the separation of the network data and control planes.

Today, practically every network box has its own control plane.

- The control plane of a router runs routing protocols and makes sure packets are routed through the network.
- The control plane of a firewall compares packet headers to a set of rules and either permits or denies packet transit.
- The control plane of an IPS/IDS compares packet traffic to a set of signatures that describe an attack against the network and raises alarms and so on.

Basically, we have specialized network devices with specialized software and chipsets for specialized purposes and most of them are purposefully built which means you can't take your router and simply transform it into a different device, say firewall. If we could rip the control plane out of these devices and move it somewhere else in the network, like a central network controller, we could replace them with general purpose devices, which would likely drive down the cost of our network and also make it more evolvable, because we would only need to reprogram the controller if we wanted to make a change to the network behavior. This is basically the idea behind data and control plane separation. Of course, if you're going to do something this drastic, it better be standardized, right? We want these devices to be truly interchangeable, despite the vendor we bought them from.

That's where the **OpenFlow protocol** comes in. **It supports data and control plane separation in a standardized way** and it promises that maybe someday we will be even able to buy merchant silicon based switches controlled by a centralized OpenFlow controller.

Why don't we implement OpenFlow immediately? The OpenFlow standard is not mature enough yet and we have yet to see a network operator or service provider (SP) that will go and replace all of their current equipment with OpenFlow switches. This would require a major network overhaul and also a lot of software development to implement the control plane to their liking and bring the network management systems up to speed. OpenFlow is currently getting a lot of traction in lab environments where different network functions need to be tried and tested.

So does that mean implementing SDN requires total network overhaul or that SDN is limited to lab environments? Fortunately it is not so because SDN does not equal data and control plane separation or does not it equal OpenFlow. SDN mainly proposes network programmability and abstraction and these principles are the ones that can be implemented without immediately migrating to OpenFlow switches.

The main reason behind SDN is the lack of network programmability which has prevented the networks from evolving faster, especially when compared to software development. One of the main culprits for this state is the lack of abstract models in networking. In software development abstract models are everywhere and we can no longer imagine software development without them. Abstract models simplify otherwise complex systems and drastically speed up the development process.

THE SOLUTION

Instead ripping out current network devices, implementing a centralized control plane and replacing network management solutions, **companies can evolve their networks towards SDN by introducing another layer to their networks that will provide abstractions**. This layer sits on top of network devices and below network management systems or operations support systems.

It's called the **orchestration layer**. It provides network abstraction and hides the devices from NMS/OSS. Most importantly, it also removes the need for proprietary network adapters, which are usually required for communication between the NMS/OSS and network devices. These adapters are usually costly and they need to be upgraded along with the firmware upgrades on network devices. The network orchestration layer gets rid of these adapters by using a standardized protocol called **NETCONF**, which has been designed with the purpose of network device configuration management in mind. We've seen configuration management platforms before but they are usually one directional since they are limited to grabbing the configuration from the device and storing them in a central database. Changing the configuration in the database and effectively pushing that configuration towards the network is a much bigger challenge and solutions like SNMP protocol and CLI scripting have utterly failed in delivering this functionality in an error prone and consistent way. Too many times have we seen these solutions push configurations to devices only to end up with faulty configuration on a part of the network and inconsistencies between the central database and the network devices.

NETCONF was built specifically to address these issues by implementing a transactional approach to network configuration. Network configuration change is an atomic transaction, which means the entire device either, confirms the new configuration or all the changes are rolled back and no changes are committed. The NETCONF protocol is combined with **YANG** modeling language, which is used to describe the network device configuration. YANG models are either provided by devices themselves or are a part of the network orchestration layer and they essentially eliminate the need for proprietary network adapter development.

Standardization with NETCONF and YANG also has another far-reaching implication. If network services are also modeled using YANG and these models are mapped to device YANG models by the network orchestration layer **the service configuration inside NMS/OSS systems can become completely independent of the network device model or brand**. As long as the network orchestration layer maps YANG service models to device YANG models the underlying physical (or virtual) device **is no longer important to the NMS/OSS**.

The consequences of NETCONF and YANG are far reaching. Suddenly it is possible to simply rip out and replace devices from one vendor and replace them with devices from a different vendor and the network orchestration layer takes care of device configuration. Such a migration would normally take months or even years. It is also possible to instantly configure network services without endangering configuration consistency or risking configuration errors and downtimes, which would usually take huge efforts to fix. New network services can be rolled out faster and if a service doesn't commercially or otherwise succeed it can be taken out of the network by simply rolling back the transaction that instantiated it. It is also possible to build custom NMS/OSS systems without worrying about the compatibility with underlying devices. The network orchestration layer will take care of that.

When it comes to implementing SDN today it is the network orchestration layer that makes this possible without requiring replacement of current network equipment with OpenFlow switches and turning the whole network architecture upside down. While it doesn't yet truly separate the data plane from the control plane it achieves a very important goal by implementing an abstract model of the network and supplying a single programming interface to the network as a whole.

Cisco is well aware of the **consequences NETCONF and YANG can have on the industry**. Cisco probably acquired a company called **Tail-F**, which built their own network orchestration layer framework called NCS (now **Cisco NSO**) with the progress of NETCONF and YANG in mind. Their network orchestration layer goes one step further by enabling NETCONF like behavior on devices which currently do not have NETCONF implemented and only support configuration methods like CLI or SNMP. Cisco NSO is able to mimic NETCONF behavior towards NMS/OSS without limiting itself to NETCONF network devices. With currently so many devices with no NETCONF support or poor NETCONF implementation out there, this is a more than a welcome feature if we truly want the benefits of SDN without having to resign to complete network overhaul. The solution is also ready for OpenFlow and once OpenFlow devices become a reality in the network, the transition from the current network to OpenFlow network will simply be a matter of providing the correct YANG models.

NIL VALUE PROPOSITION

NIL can help service providers evaluate and implement the network orchestration layer and enable them to become SDN companies today. We specialize in Cisco NSO implementation, integration and service model design. We can help you automate new network services and provide a network abstraction to your NMS/OSS systems or help you develop a new management system based on standards based orchestration layer. NIL also delivers training and consulting services around network orchestration and automation.

For more information, please contact us via sales@nil.com or find our local office.