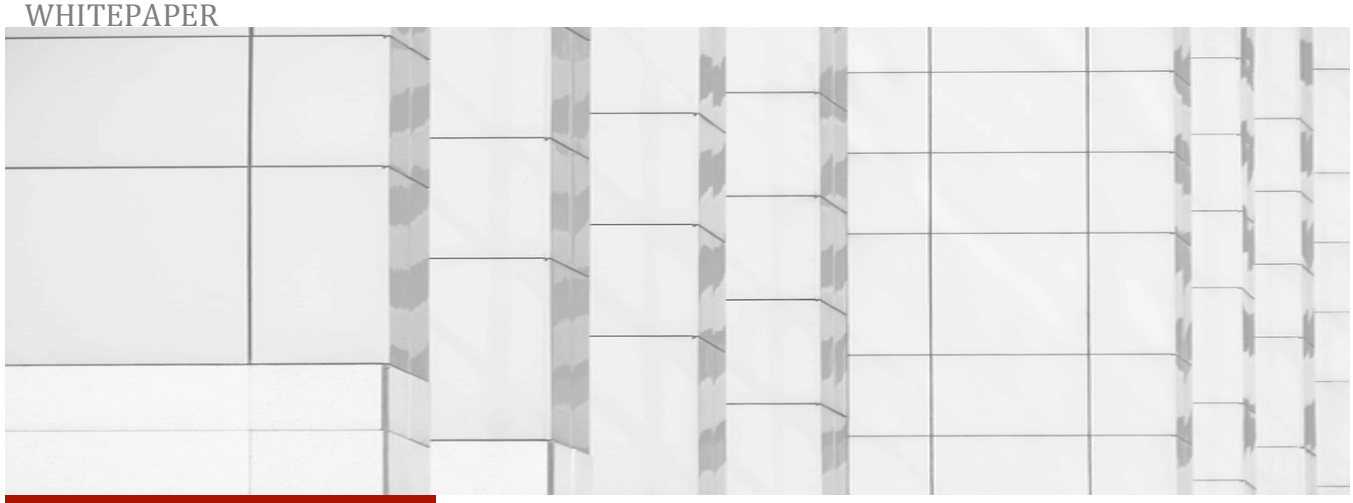
WHITEPAPER



Software Defined Networking (SDN)

Introducing ONOS - a SDN network operating system for Service Providers

This whitepaper describes ONOS, a SDN network operating system designed for high availability, performance, scale-out and rich abstractions.

ON.LAB

SUMMARY

The Open Network Operating System (ONOS) is the first open source SDN network operating system targeted specifically at the Service Provider and mission critical networks. ONOS is purpose built to provide the high availability (HA), scale-out, and performance these networks demand. In addition, ONOS has created useful Northbound abstractions and APIs to enable easier application development and Southbound abstractions and interfaces to allow for control of OpenFlow-ready and legacy devices. Thus, ONOS will

- bring carrier grade features (scale, availability, and performance) to the SDN control plane
- enable Web style agility
- help service providers migrate their existing networks to white boxes
- lower service provider CapEx and OpEx

ONOS has been developed in concert with leading service providers (AT&T, NTT Communications), with demanding network vendors (Ciena, Ericsson, Fujitsu, Huawei, Intel, NEC), R&E network operators (Internet2, CNIT, CREATE-NET), collaborators (SRI, Infoblox), and with ONF to validate its architecture through real world use cases.

THE SERVICE PROVIDER NEED

The proliferation of mobile devices, OTT services and distribution of content across the cloud have pushed Service Provider networks to a point where they are in urgent need of reinvention. Service Providers want to make their networks agile and efficient in order to meet the challenges of these exponential bandwidth demands and want to be able to create revenue streams with innovative services and new business models.

SDN AND ONOS TO THE RESCUE

In just a short time, Software Defined Networking (SDN) has become the technology of choice for enabling mobility, virtualization, and the Cloud. Storage and compute have been virtualized for years, yet one could not capture the full value of these advancements because the network is what enables the value to be unlocked - it is the fabric that connects them together and to applications.

The key SDN concept that enables similar network innovation is the separation of the control plane from the data plane in vertically integrated network devices. A non-proprietary protocol such as OpenFlow allows the control plane to program the data plane in a much more open and efficient way. In addition, this separation allows network hardware and software to evolve independently and facilitates the replacement of expensive, proprietary hardware and firmware with commodity hardware and open source software. Having an operating system that manages network resources and provides the abstractions and APIs for managing, monitoring, and programming network devices greatly simplifies the creation of innovative and beneficial network applications and services that operate across a wide range of hardware. Open Network Operating System (ONOS) was created to be this operating system and has the following goals:

- Liberate network application developers from knowing the intricacies of proprietary hardware.
- Allow network operators to break free from the operational complexities of proprietary interfaces and protocols.
- Re-enable innovation to happen for both network hardware and software, independently, on their own time scales.

WHY A NETWORK OPERATING SYSTEM?

There are many open source SDN controllers; it is therefore reasonable to ask "Why ONOS, and why a network OS?". Collaborators at Stanford, Berkeley, and Nicira Networks developed several open source controllers during the past seven years, including NOX, Beacon, SNAC and POX. These controllers were designed to explore and demonstrate SDN potential. Much was learned from working with these controllers and by building the applications and demonstrations they enabled. However, it is important to understand these controllers were not designed to be a foundation for commercial products. They don't have the key features such as scalability, high availability, and performance. Moreover, they have primitive programming and device-oriented abstractions.

These controllers fundamentally relayed OpenFlow messages directly to applications, and applications directly created OpenFlow messages for the network devices. In this way, these controllers are more like device drivers. They are not designed to have the critical scalability, availability and performance features of a complete SDN control platform. What is needed is a network OS – ONOS has been created to fulfill this need.

Recall that an operating system is responsible for the following functions:

- It manages finite resources on behalf of resource consumers. In doing so, it ensures that all consumers have appropriate access - none is starved, all are handled fairly.
- It isolates and protects NOS users from each other - each user appears to have access to a full set of resources. It multiplexes between multiple applications and multiple devices. It may also virtualize some or all of the resources to allow consumers their own virtual instantiation of the OS.
- It provides useful abstractions to enable users to more easily consume services and resources managed by the operating system, without having to understand all of the complexity. It provides mechanisms for different devices to be easily added to and controlled by the operating system without requiring the application to change.
- It provides security from the external world to the users of the operating system.
- It supplies useful services so that users of the operating system don't have to build and rebuild the same services.

These are *exactly* the things needed by applications running on a network. A controller is typically too limited in scope - it sets out to control a device. It does not necessarily provide useful abstractions; it does not necessarily protect different controller users from each other; it does not provide additional useful services. ONOS sets out to provide all of the functionality of an operating system - not just the functionality of a controller. How does ONOS accomplish this? By architecting the software with service provider features.

ONOS ARCHITECTURE

ONOS has been architected from the beginning with the service provider in mind. High Availability, Scale-out, and Performance are fundamental, as are powerful abstractions at the Northbound and Southbound interfaces.

The following are the defining features of ONOS:

- **Distributed Core** that provides scalability, high availability, and performance – bring carrier grade features to the SDN control plane. The ability of ONOS to run as a cluster is one way that ONOS brings web style agility to the SDN control plane and to service provider networks.
- **Northbound abstraction/APIs** that include network graph and application intents to ease development of control, management, and configuration services. This abstraction is another good example of how ONOS brings web style agility to the SDN control plane and to service provider networks.
- **Southbound abstraction/APIs** that enable pluggable southbound protocols for controlling both OpenFlow and Legacy devices. The southbound abstraction insulates the core of ONOS from the details of different devices and protocols. The southbound is a key enabler for migration from legacy devices to OpenFlow-based white boxes.
- **Software Modularity** makes it easy to develop, debug, maintain, and upgrade ONOS as a software system by a community of developers and by the providers.

Each of these architectural features is described in more detail in the following sections.

DISTRIBUTED CORE

ONOS is deployed as a service on a cluster of servers, and the same ONOS software runs on each server. Deployment symmetry is an important design consideration as it enables rapid failover in the event of an ONOS server failure. The network operator can add servers incrementally, without disruption, as needed for additional control plane capacity. The ONOS instances work together to create what appears to the rest of the network and applications as a single platform. Applications and network devices do not have to know if they are working with a single instance or with multiple instances of ONOS. This feature makes ONOS scalable – one can scale ONOS capacity seamlessly. It is the Distributed Core that does the heavy lifting to realize these capabilities.

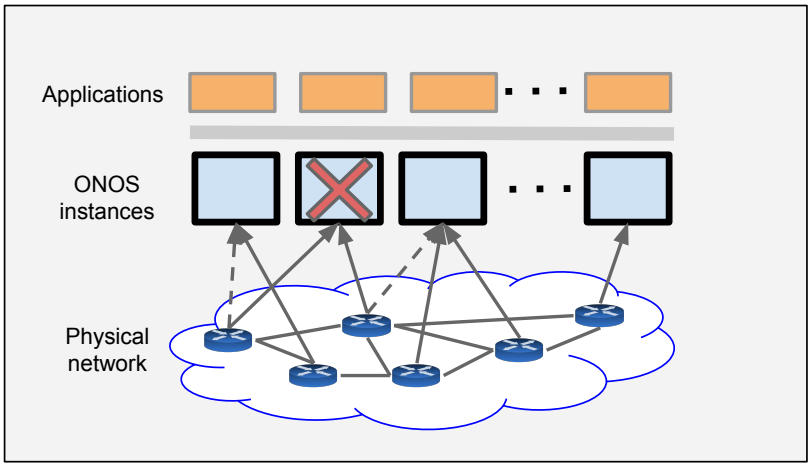


Figure 1. Distributed Core

The distributed core provides messaging, state management and leader election services to and between instances. As a result, multiple instances behave as a single logical entity. Using high speed messaging in a publish/subscribe model, instances can quickly inform other instances of updates. Built into ONOS are recovery protocols for dealing with updates that are lost due to instance failures. A variety of operational state is managed between instances using several mechanisms - each being appropriate for the type of state. Three examples include the application intents, the topology database, and the flow tables - each has unique size, read/write pattern, and durability requirements. A leader election service ensures that switches have one and only one master instance. Together, the messaging, state management, and leader election mechanisms enable high throughput, low latency, and high availability of the cluster.

What does this mean? For devices, they will always have a single master and if the master goes down, they will be able to connect to another instance without having to recreate and resynchronize the flow tables. For applications, they can count on having a consistent view of the network through the network graph abstraction. In addition, a failure of an instance or a failure in the data plane is transparent to the application. These both greatly simplify application development and error handling.

From a business perspective, it brings a very highly available environment so that applications do not experience network-related downtime. It also means that the service provider can easily add control plane capacity as the network grows, without disruption to the network. Through the same mechanism, the network operator also has the capability to update software with zero system downtime by taking an instance offline, upgrading it, and bringing it back online.

In summary, the distributed core is the key architectural feature of ONOS that brings carrier grade features to the SDN control plane.

NORTHBOUND ABSTRACTIONS

There are two powerful Northbound abstractions: the Intent Framework and the Global Network View.

The Intent Framework allows an application to request a service from the network without having to know details of how the service will be performed. This allows network operators as well as application developers to program the network at a high level; they can simply specify their intent: a policy statement or connectivity requirement.

Some example intents:

- Set up a connection between Host A and Host B
- Set up an Optical Path from Switch X to Switch Y with Z amount of bandwidth
- Don’t allow host A to talk to host B

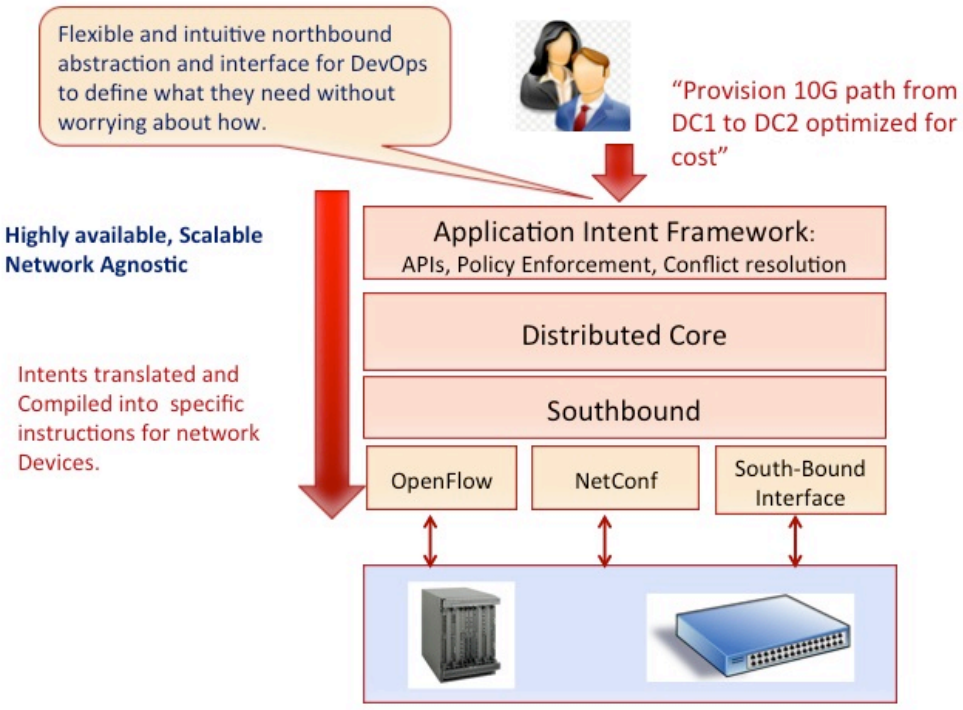


Figure 2. Intent Framework

The Intent Framework takes such requests from all applications, figures out which ones can and cannot be accommodated, resolves conflicts between applications, applies policies set by an administrator, programs the network to provide the requested functionality, and delivers the requested services to the application.

An intent is translated into multiple objectives - for example, an intent to have a connection between two hosts translates into two objectives, each providing one direction of flow. The objectives are compiled into instructions that are sent to the network devices. This process is done under the control of policies specified by the network operator and in a way that resolves conflicts between different intents.

The Global Network View provides the application with a view of the Network - the hosts, switches, links, and any other state associated with the network such as utilization. An application can program this network view through APIs. One API lets an application look at the view as a network graph. Some examples of what can be done with the network graph include:

- create a simple application to calculate shortest paths since the application already has a graphical view of the network
- maximize network utilization by monitoring the network view and programming changes to paths to adjust load (traffic engineering)
- steer traffic away from a part of the network that is being upgraded or that is being quarantined for a virus.

Technically, the northbound abstractions and APIs insulate applications from details of the network that are not needed by the application. The abstractions can also insulate applications from network events (like link down) when desired by the application. Conversely, it insulates

the operating system from the applications allowing the operating system to do its job of managing requests from multiple, competing applications. From a business perspective, this increases application development velocity and allows network changes without application downtime.

SOUTHBOUND ABSTRACTIONS

The southbound abstraction is built using network elements, such as switches, hosts, or links. The southbound abstraction of ONOS represents each network element as an object in a generic form. Through this abstraction, the distributed core can maintain the state of the network element without having to know the specifics of the element represented by the underlying driver. In effect, it allows the core to be southbound protocol and device agnostic. The network element abstraction is also what allows addition of new devices and protocols. ONOS and its southbound abstraction allow plug-ins for various southbound protocols and devices, where a plug-in maps or translates generic network element description or operation on the device to the specific and vice-versa. Thus the southbound enables ONOS to control or manage multiple diverse devices, even if they use different protocols (OpenFlow, NetConf, etc.).

Architecturally, the southbound is composed of the layers shown in figure 3. At the bottom are the network devices or elements. ONOS interacts with devices through protocols. The protocol specifics are abstracted away by the network element plug-in or adapter. As a result, the core of the southbound can maintain its network element objects (devices, hosts, links) without having to know the specifics of the protocols and network elements. Through the adapter API, the distributed core is kept up to date on the status of the network element objects. The adapter API insulates the distributed core from having to know details about protocols and network elements.

The main benefits of the southbound abstractions include:

- ability to manage different devices using different protocols - without effect on the distributed core of the system
- ability to add new devices and protocols to the system
- ease of migration from legacy devices and protocols to white boxes supporting OpenFlow

SOFTWARE MODULARITY

Software construction matters. Done correctly, software is easy to enhance, change, and maintain. The ONOS team has put great care into modularity to make it easy for developers to work with the software.

What is modularity? It is how the software is structured into components and how those components relate to one another. As apparent from diagram below, the major structures of ONOS are its tiers centered around the distributed core. Thus, at the macro level the Northbound and Southbound APIs provide an initial basis for insulating Applications, Core and Adapters from each other. New applications or new protocol adapters can be added as needed without each needing to know about the other.

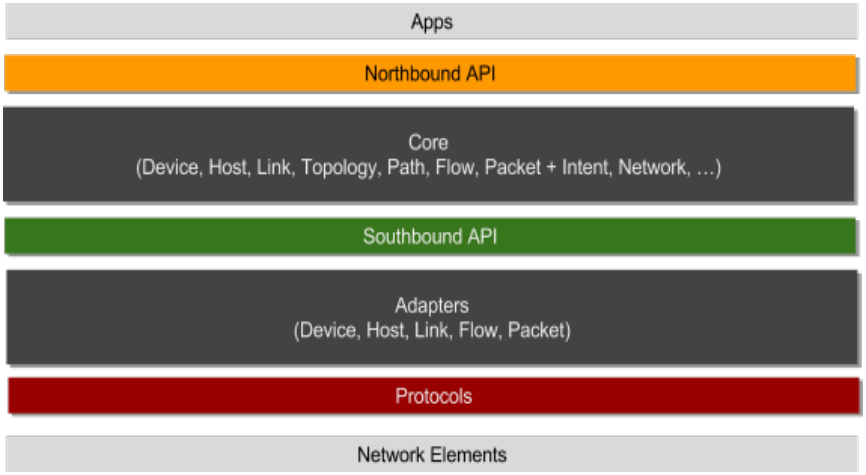


Figure 3. ONOS Layers

Similarly, beneath this macro level depiction are smaller substructures within the Core itself, which exist to limit the size of any particular subsystem and to facilitate modular extensibility. Again, these rely heavily on interfaces to serve as contracts for interactions between different parts of the core, allowing each part of the core to evolve independently from others. This enables new algorithms or more efficient data structures can be provided over time, without affecting large parts of the system or the applications.

Clearly, major focus has been placed on making sure the interfaces encourage separation of concerns and responsibilities in order to keep the interactions between subsystems as natural and simple as possible. This is essential for stable evolution of the software base. For example, on the Southbound API, care was taken to raise the level of abstraction in order to avoid general bias towards any specific protocol and also to enforce the convention where the Core, and not the Adapters create the network model objects.

ONOS source tree structure is setup to not only follow, but to enforce these architectural principles. Modules are kept reasonably small and dependencies among them form an acyclic graph, where direct dependencies between modules are realized through API modules as depicted in diagram below.

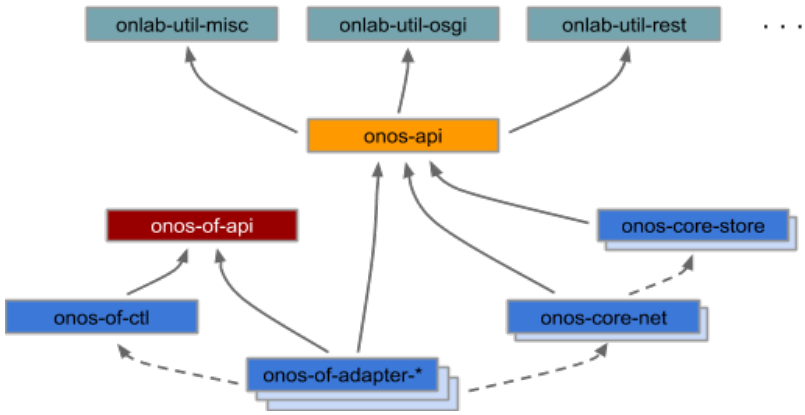


Figure 4. ONOS Modules

In summary, there are many benefits to software modularity:

- Architectural integrity and coherence
- Simplified test structure, allowing more comprehensive testing
- Easier maintenance with fewer side effects of changes
- Extensibility and customization of components
- Avoidance of cyclic dependencies

ONOS VALUE PROPOSITION - ENABLING OPERATOR USE CASES

- Multi-layer SDN control of packet-optical core

Service Providers operate multi-layer networks. For example, a service provider operates IP packet network as well as a transport or optical network. A tunneling layer may also exist on top of the IP substrate to create services such as virtual IP layer networks. Each of these layers is managed independently today resulting in low levels of network utilization, high operational costs, and reconfiguration cycles that are in months rather than minutes. For example, in today's environment, packet network designers provision additional capacity to handle network failures and peak bursty traffic loads as well as failures; and independently, transport networks designers do the same at the optical level. Moreover, packet network designers like to operate at 30% average utilization. This, in aggregate, leads to 4-5x total over-provisioning of capacity.

SDN control of multi-layer networks addresses the problems mentioned above. The solution includes three components: (1) addition of OpenFlow like programmability to optical transport elements such as ROADMs; (2) ONOS to build a network graph view for each layer of the network and maintain the mapping or correspondence among them; and (3) development of a PCE application on ONOS that set ups and tears down path, taking into account multiple correlated network graphs.

ONOS PCE application is able to configure, monitor and orchestrate the layers as though it were a single entity. For example, IP connection changes can trigger automatic provisioning of optical paths. The operator now has the ability to lower operational costs through central control, to raise network utilization on all layers, and to reconfigure the network in minutes. One of the first applications of this capability is a bandwidth calendaring application that allows bandwidth to be reserved in the future. ONOS provisions the packet and optical layers to provide the bandwidth, and then monitors the resources to perform re-routing and adjustments based on real time network events and other changes.

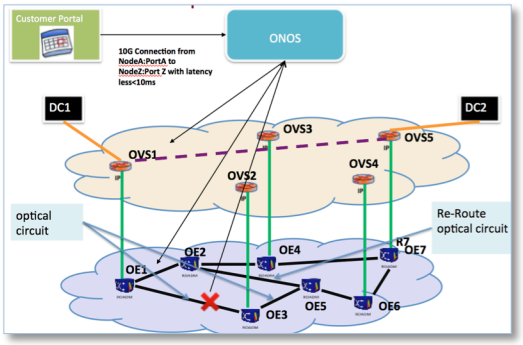


Figure 5 . Multi-layer SDN control

SDN control of multi-layer networks is an excellent example of how SDN control implemented with ONOS can lead to significant savings in CapEx and OpEx as well as enable new class of services.

- SDN-IP Peering and Scaling SDN Control Plane

Today, Autonomous Systems (AS) connect to the Internet and with each other using BGP to share routing information. The SDN-IP peering application on ONOS is designed to help service providers start small and grow their SDN deployments. A service provider can deploy a small SDN network or an SDN island as an AS and use the SDN-IP peering application to seamlessly connect the SDN network to the rest of the Internet using BGP. With the SDN-IP application, an SDN network appears as just another AS to the rest of the Internet. Over time, the service provider can keeping scaling the SDN network and realize all the benefits without impacting its peering with the Internet. Moreover, a service provider can use the SDN-IP peering application to inter-connect multiple SDN networks to create a large SDN Autonomous System (AS) that peers with the Internet the same way as other AS.

The SDN-IP application peers with AS border routers and exchanges route information for IP prefixes as is done between standard AS today. The SDN-IP peering application uses the routing information to set up forwarding paths for various Internet prefixes on the SDN network. Thus SDN network can act as a transit network for some prefixes and can forward IP traffic to and from any IP address that is reachable.

In addition, SDN-IP application can be used to scale ONOS-based SDN control plane. For example SDN networks or domains can be interconnected through BGP similar to non-SDN AS today. Confederation works in a similar way as well - a group of SDN domains can be viewed by the rest of the Internet as a single AS.

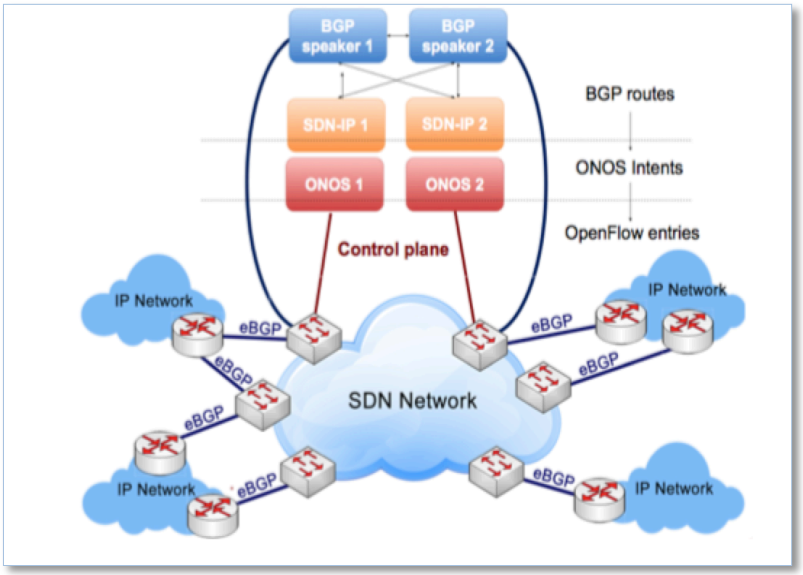


Figure 6. SDN-IP

Thus SDN-IP peering application plays an important role in incremental and seamless SDN deployment in a service provider infrastructure. ON.Lab is deploying SDN-IP peering application on Internet 2, and it is being used to interconnect campus networks that are migrating to SDN via Internet2's SDN backbone.

- Network Functions as a Service (NFaaS) in Central Office

Telecom providers have a valuable asset in their networks today - the central offices. These central offices are geographically close to a large number of end users, and thus they are strategic in that they enable the telecom providers to offer services at scale to a large number of customers with performance and predictability. For example, AT&T operates close to 4,500 central offices in the country and a typical central office in a large metro area can serve as many as ~100K wired and ~100K cellular subscribers, and 100s of enterprises. A typical central office has lot of access and switching capacity and many purpose built middle boxes for various network functions. Though central offices are strategic to telecom operators, they have evolved over many decades and have significant complexity in terms of network switching/routing, middle boxes, and termination of customers. They represent significant CapEx and OpEx for the service providers.

Not surprisingly, service providers want to re-architect the central office. They want to bring the data center economies and agility to the central office. This means building the central office network fabric using SDN, transforming network functions implemented using purpose built middle boxes to software running on x86 servers (so called NFV), and an orchestration system that can orchestrate network functions and network flows for a subscriber or enterprise customer -- dynamically to serve individual customers while realizing various policies of both the customer and the provider.

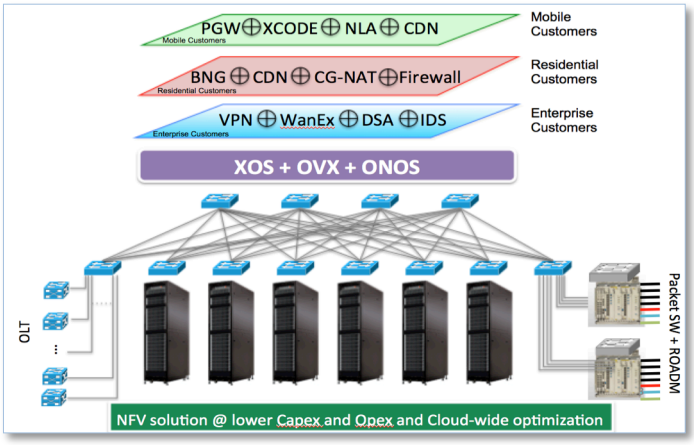


Figure 7. NFaaS in Central Office

This transformation of the central office will allow the service providers to quickly create, deploy and offer new services to customers at scale and at the same time, significantly cut the CapEx and OpEx.

This ONOS use case demonstrates the potential of this re-architected central office at a small scale. ONOS, its applications and OpenFlow enabled switches help transform the network fabric of the central office to SDN providing the agility required for the Network Functions as a Service. We expect to build on this use case in two important directions: demonstrate ONOS can be the network OS for the re-architected central office network in terms of scale and performance; and demonstrate how SDN and Network Functions as a Service fit within the same larger architecture of the central office.

- Segment Routing- Evolving and Improving MPLS

Today's IP/MPLS networks are complex and hard to manage. Label distribution, traffic engineering, and VPNs are very complex operations and services that depend on a collection of distributed protocols in the control plane. Furthermore debugging such networks is incredibly hard given synchronization and state-management issues between multiple protocols in the control plane and a locally-significant label-swapped data plane.

The IETF has introduced the concept of Segment Routing (SR) for MPLS (known by its IETF name SPRING). It introduces globally-significant labels that don't need to be swapped at each hop. It also introduces source-routing based on labels, which eliminates dependence on complex protocols like LDP and RSVP for label distribution and LSP setup. Segment Routing thus simplifies both the control and data plane of MPLS networks. While SR continues to depend on an IGP for routing and label distribution, it opens the possibility for an external controller to program end to end tunnels originating at the source router.

The use case on Segment Routing is being pursued in collaboration with the Open Networking Foundation's SPRING-OPEN project led by Saurav Das. The project demonstrates how SR can be realized using the SDN control plane implemented with ONOS and an SR application, working with bare-metal hardware routers built on merchant silicon that exists today. This solution does not use a distributed IGP embedded in the routers. Instead it uses a routing application on ONOS. And the application programs the edge-routers and core-routers for forwarding with segment routing rules for default routing and policy-based routing. With ONOS providing the control and an application managing the labels in the network, network operators can express their policy requirements to the controller, and the app together with ONOS implements the policy in the IP/MPLS network.

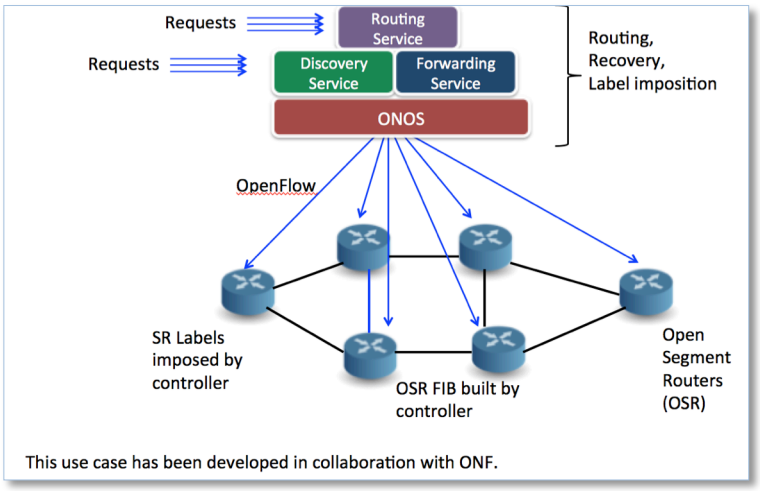


Figure 8. Segment Routing*

Thus segment routing with ONOS demonstrates how service providers can build on the MPLS data plane without the complexity of its control plane. They can have best of both: simplicity of the SR MPLS data plane and agility of the SDN control plane.

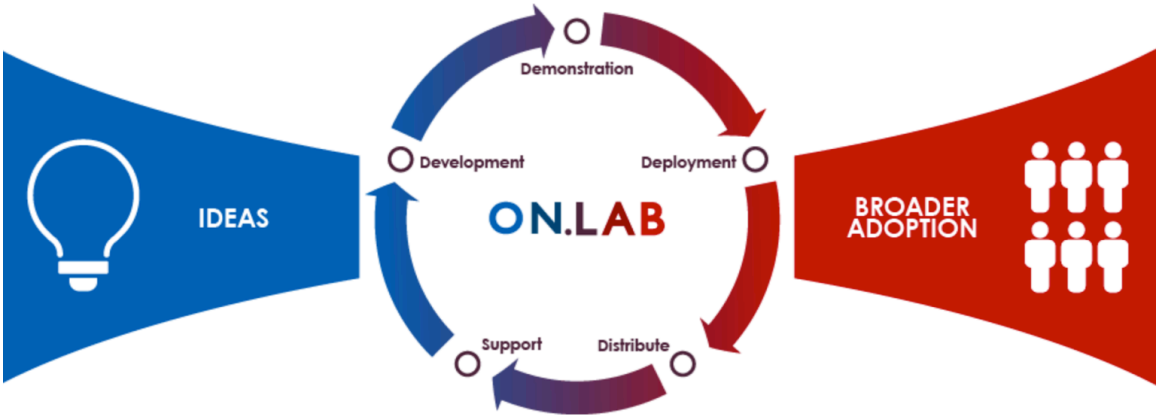
** Source: ONF*

CONCLUSION

Our goal with ONOS has been to create an open source SDN network OS for the Service Provider and other mission critical networks. ONOS is designed to provide (1) carrier grade features such as scalability, high availability, performance in terms of throughput (application intents per second) and latency (time to process network events); (2) northbound abstraction/APIs to make it easy to create new services using ONOS – that is to bring web style agility to networks; and (3) southbound abstraction with device/protocol plug-ins so ONOS can provide SDN control for OpenFlow enabled white boxes as well as legacy devices. This enables easy migration to SDN based on white boxes.

Our work on ONOS has been guided by a set of use cases proposed by our partners that include leading service providers and vendors.

The ONOS November release represents a solid network operating system platform to seed an open source project. However, we still have some ways to go to turn ONOS into a production ready platform. We have to develop many more use cases, continue to improve the performance, enhance key features and do trials and deployment to provide real proof points. Open Sourcing ONOS is a crucial milestone because it brings in the broader community to join us in evolving this platform and truly delivering on our mission of creating a carrier-grade, open source SDN OS for mission critical networks.



ABOUT ON.LAB

The Open Networking Lab (ON.Lab) is a non-profit organization founded by SDN inventors and leaders from Stanford University and UC Berkeley to foster an open source community for developing tools and platforms to realize the full potential of SDN. ON.Lab brings innovative ideas from leading edge research and delivers high quality open source platforms on which members of its ecosystem and the industry can build real products and solutions. ON.Lab has a team of highly motivated and talented individuals, with expertise and a stellar track record in industry and research institutions. ON.Lab's team is focused on creating high quality open source tools and platforms that benefit and bring true SDN value to the community.