SD-FABRIC

Open Source Full-Stack Programmable Leaf-Spine Network Fabric

Executive Summary

SD-Fabric™ is a unique open source, full stack, deeply programmable network fabric. It is optimized for distributed edge networks, supporting cloud native edge cloud applications, while being centrally managed from the public cloud. SD-Fabric is constructed to be deeply programmable, allowing application developers to 'push' functionality down into networking switches and (eventually) to smart NICs and soft switches, using the P4 programming language.

SD-Fabric creates a fully programmable and verifiable data plane created and programmed using P4™. This is constructed, deployed, and maintained as a set of cloud native applications exposing SaaS APIs to support application developers, enabling developers to use the network as a vehicle for implementing new and innovative functionality, at very high speed, directly in the network itself.

Background

The development of SD-Fabric has been influenced by a number of core trends in the industry, including edge cloud, cloud-native, and 5G.

With the expectation that AI and ML will create a transformative opportunity for industrial use cases (i.e., Industry 4.0), and that AR will find broad application for consumer and industrial markets, the requirement for workloads to run at the edge of the network close to the end users and devices has become well understood.

This movement has led to the edge cloud being broadly viewed as the center of innovation for new innovative applications, and 5G as the enabler to connect users and devices to these new workloads running at the edge. But edge sites are constrained by space, power and cost. The model of deploying unlimited cloud-based compute power to solve problems is ill suited for the edge where customized workloads need to be optimized to run on more limited physical resources.

Furthermore, Moore's law has stalled (CPUs no longer are doubling in speed every 18 months), so applications need to be spread across compute resources to meet ever-increasing application demands. While cloud native scale-out has become the tool of choice for such scaling, the inefficiencies of the additional inter-server/inter-process communication consumes vast resources that are driving up cloud hosting expenses and the environmental impact of cloud computing. There is a need to rethink this approach to cloud-native, and to perform more processing in the network itself rather than relying on network functions that run in containers to perform tasks like load balancing.

Lastly, there is a growing trend of enterprises wanting to consume everything 'as-a-service', leveraging APIs to empower developers with simplified access to advanced capabilities. This applies even to the network itself, which should be possible to consume as a cloud-managed service. It is in response to these macro-level drivers that SD-Fabric has been conceived and developed, creating a developer friendly, cloud-managed, programmable network fabric that enables new classes of emerging edge applications for Industry 4.0 powered by 5G.

Overview

SD-Fabric is an open source, full stack, deeply programmable network fabric optimized for edge cloud, 5G, and Industry 4.0 applications. It builds on SDN and cloud native principles to create a disruptive platform that for the first time empowers the network with modern cloud development principles.

- First, SD-Fabric is a natively disaggregated solution using bare metal switches with merchant silicon ASICs. Instead of using OEM networking hardware, SD-Fabric uses hardware directly from ODMs. The trend of using bare metal (white box) switches is unmistakable in the networking industry today, spurred by the massive bandwidth-density and growing sophistication of merchant silicon ASICs.
- Second, SD-Fabric is based on SDN principles using P4-enabled programmable data planes (such as the Intel® Tofino® ASICs). P4 allows for the introduction of new features that cannot be found in traditional fabrics built on fixed-function ASICs. For example, SD-Fabric makes possible a 5G mobile core User Plane Function (UPF) (this UPF is part of the SD-Core™ project, which runs on SD-Fabric). And by externalizing the network's control, management functions, and policy decisions in the ONOS™ SDN controller, SD-Fabric provides network operators with a number of SDN benefits (compared to traditional embedded network control like BGP) that include customizability, centralized configuration, automation, and simplified operations and troubleshooting. As a result, SD-Fabric is truly software defined in both the control and data planes.
- SD-Fabric is optimized for application developers, exposing APIs to enable developers to get better visibility and control while enabling custom P4 forwarding logic to be 'pushed' down into the network. SD-Fabric can further be deployed in a SaaS model, with centralized operations and control running from the public cloud which in turn is running distributed multiple edge clouds - all as one unified solution.
- SD-Fabric delivers superior resilience and security compared to legacy approaches. The programmability of SD-Fabric has enabled the implementation of fine-grained measurement (via In-band Network Telemetry (INT)), network verification, and closed loop control (see https://prontoproject.org/). Furthermore, as open source, SD-Fabric is further secured through the benefit of open inspection by a broad community.

- SD-Fabric can be deployed as-a-Service from the public cloud. When deployed as-a-Service, SD-Fabric provides a full stack implementation designed to run on white box switches at the network edge, with centralized CI/CD and operations running from the public cloud controlling multiple edges.

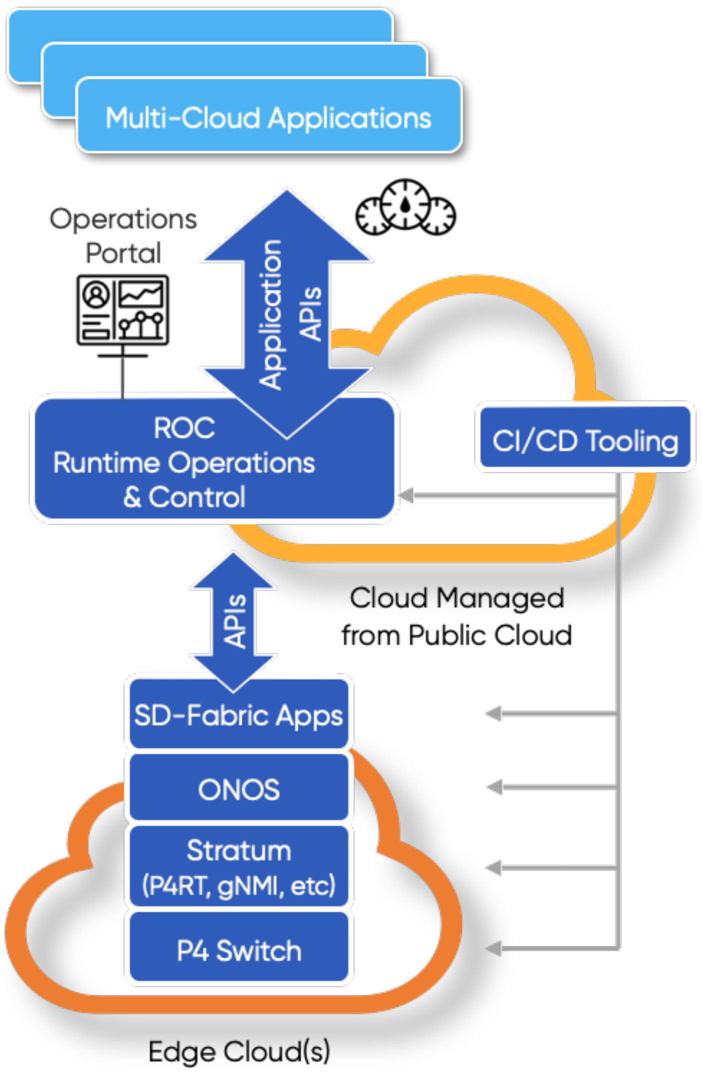

Figure 1: SD-Fabric Software Stack

Beyond Traditional Fabrics

While SD-Fabric offers advancements that go well beyond traditional fabrics, it is first helpful to understand that SD-Fabric provides all the features found in network fabrics from traditional networking vendors in order to make SD-Fabric compatible with all existing infrastructure (servers, applications, etc.).

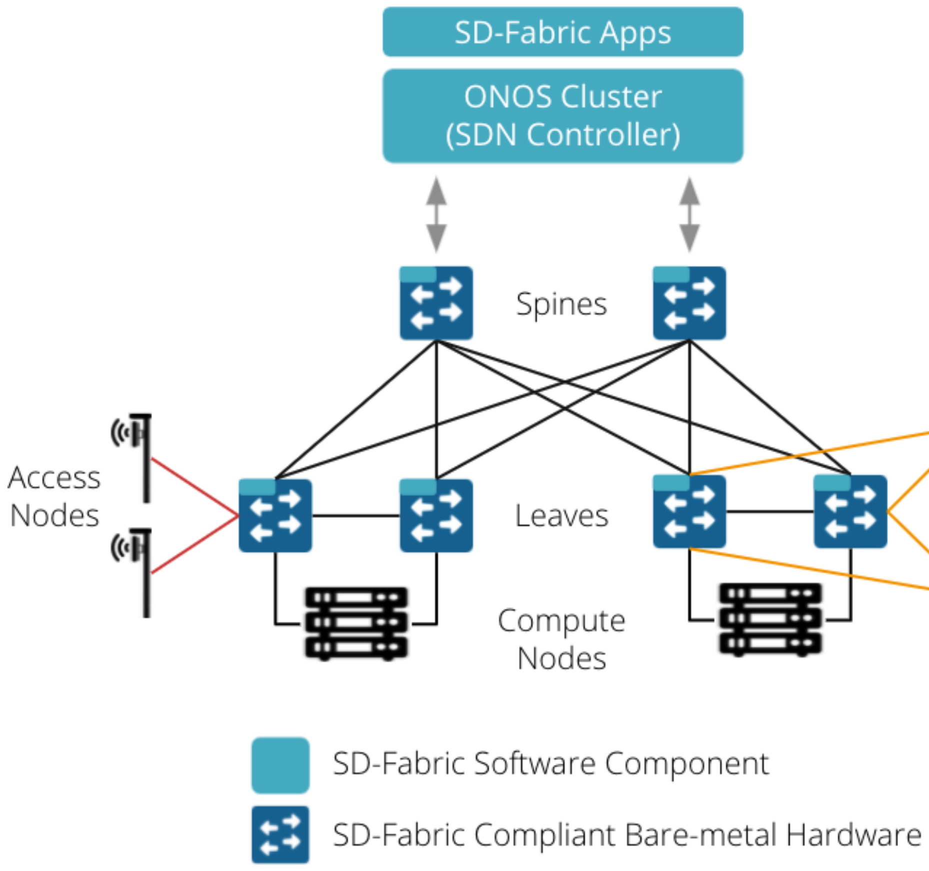
SD-Fabric Apps
ONOS Cluster (SDN Controller)
Spines
Access Nodes
Leaves
Upstream Routers
Compute Nodes
SD-Fabric Software Component
SD-Fabric Compliant Bare-metal Hardware

Figure 2: SD-Fabric in a Typical Leaf-Spine Topology

At its core, SD-Fabric is a L3 fabric where both IPv4 and IPv6 packets are routed across server racks using multiple equal-cost paths via spine switches. L2 bridging and VLANs are also supported within each server rack, and compute nodes can be dual-homed to two Top-of-Rack (ToR) switches in an active-active configuration (M-LAG). SD-Fabric assumes that the fabric connects to the public Internet and the public cloud (or other networks) via traditional router(s). SD-Fabric supports a number of other router features like static routes, multicast, DHCP L3 Relay and the use of ACLs based on layer 2/3/4 options to drop traffic at ingress or redirect traffic via Policy Based Routing. But SDN control greatly simplifies the software running on each switch, and control is moved into SDN applications running in the edge cloud.

While these traditional switching/routing features are not particularly novel, SD-Fabric's fundamental embrace of programmable silicon offers advantages that go far beyond traditional fabrics.

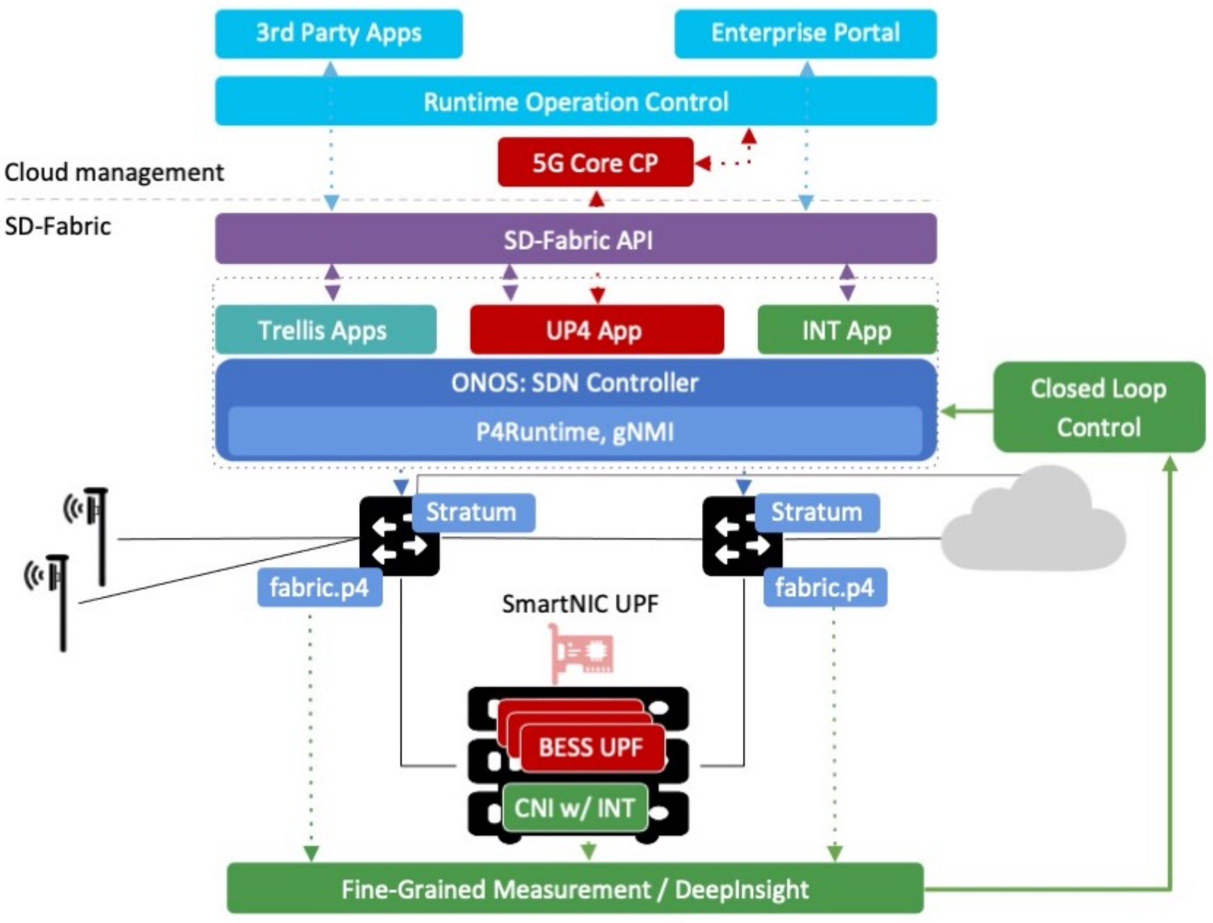

Figure 3: Beyond Traditional Fabrics

- **Right-sized Topology:** Scale from smallest HA setup of a pair-of-ToRs (sub/single rack) to a full leaf-spine fabric (multiple racks) as needed in edge or DC deployments.
- **API Driven:** Programmable API with ability to drop or reroute traffic plus obtain telemetry and program application workloads across switches, CPU and NICs.
- **Cloud Managed:** Fully integrated and configured by Aether™ Management platform.
- **5G as a Workload:** Tofino + BESS UPF scalable on demand, Smart NIC + BESS UPF extensions, 5G slicing as primary construct.
- **Visibility:** Throughout the entire network enabling closed loop control.
- **Integration:** With K8s CNI and overlay enabling true end-to-end programmability and visibility.

Programmable Data Planes & P4

SD-Fabric's data plane is fully programmable. In marked contrast to traditional fabrics, features are not prescribed by switch vendors. This is made possible by P4, a high-level programming language used to define the switch packet processing pipeline, which can be compiled to run at line-rate on programmable ASICs like Intel Tofino (see https://opennetworking.org/p4/). P4 allows operators to continuously evolve their network infrastructure by re-programming the existing switches, rolling out new features and services on a weekly basis. In contrast, traditional fabrics based on fixed-function ASICs are subject to extremely long hardware development cycles (4 years on average) and require expensive infrastructure upgrades to support new features.

SD-Fabric takes advantage of P4 programmability by extending the traditional L2/L3 pipeline for switching and routing with specialized functions such as 4G/5G Mobile Core User Plane Function (UPF) and Inband Network Telemetry (INT).

4G/5G Mobile Core User Plane Function (UPF)

Switches in SD-Fabric can be programmed to perform UPF functions at line rate. The L2/L3 packet processing pipeline running on Intel Tofino switches has been extended to include capabilities such as GTP-U tunnel termination, usage reporting, idle-mode buffering, QoS, slicing, and more. Similar to vRouter, a new ONOS app abstracts the whole leaf-spine fabric as one big UPF, providing integration with the mobile core control plane using a 3GPP-compliant implementation of the Packet Forwarding Control Protocol (PFCP).

With integrated UPF processing, SD-Fabric can implement a 4G/5G local breakout for edge applications that is multi-terabit and low-latency, without taking away CPU processing power for containers or VMs. In contrast to UPF solutions based on full or partial smartNIC offload, SD-Fabric's embedded UPF does not require additional hardware other than the same leaf and spine switches used to interconnect servers and base stations. At the same time, SD-Fabric can be integrated with both CPU-based or smartNIC-based UPFs to improve scale while supporting differentiated services on a hardware-based fast-path at line rate for mission critical 4G/5G applications (see https://opennetworking.org/sd-core/ for more details).

Visibility with Inband Network Telemetry (INT)

SD-Fabric comes with scalable support for INT, providing unprecedented visibility into how individual packets are processed by the fabric. To this end, the P4-defined switch pipeline has been extended with the ability to generate INT reports for a number of packet events and anomalies, for example:

- For each flow (5-tuple), it produces periodic reports to monitor the path in terms of which switches, ports, queues, and end-to-end latency is introduced by each network hop (switch).
- If a packet gets dropped, it generates a report carrying the switch ID and the drop reason (e.g., routing table miss, TTL zero, queue congestion, and more).
- During congestion, it produces reports to reconstruct a snapshot of the queue at a given time, making it possible to identify exactly which flow is causing delay or drops to other flows.
- For GTP-U tunnels, it produces reports about the inner flow, thus monitoring the forwarding behavior and perceived QoS for individual UE flows.

SD-Fabric's INT implementation is compliant with the open source INT specification, and it has been validated to work with Intel's DeepInsight performance monitoring solution, which acts as the collector of INT reports generated by switches. Moreover, to avoid overloading the INT collector and to minimize the overhead of INT reports in the fabric, SD-Fabric's data plane uses P4 to implement smart filters and triggers that drastically reduce the number of reports generated, for example, by filtering out duplicates and by triggering report generation only in case of meaningful anomalies (e.g., spikes in hop latency, path changes, drops, queue congestion, etc.). In contrast to other sampling-based approaches which often allow some anomalies to go undetected, SD-Fabric provides precise INT-based visibility that can scale to millions of flows.

Flexible ASIC Resource Allocation

The P4 program at the base of SD-Fabric's software stack defines match-action tables for common L2/L3 features such as bridging, IPv4/Ipv6 routing, MPLS termination, and ACL, as well as specialized features like UPF, with tables that store GTP-U tunnel information and more. In contrast to fixed-function ASICs used in traditional fabrics, table sizes are not fixed. The use of programmable ASICs like Intel Tofino in SD-Fabric enables the P4 program to be adapted to specific deployment requirements. For example, for routing-heavy deployments, one could decide to increase the IPv4 routing table to take up to 90% of the total ASIC memory, with an arbitrary ratio of longest-prefix match (LPM) entries and exact match /32 entries, while reducing the size of other tables. Similarly, when using SD-Fabric for UPF, one could decide to recompile the P4 program with larger GTP-U tunnel tables, while reducing the IPv4 routing table size to 10-100 entries (since most traffic is tunneled) or by entirely removing the IPv6 tables.

Closed Loop Control

With complete transparency, visibility, and verifiability, SD-Fabric becomes capable of being optimized and secured through programmatic real-time closed loop control. By defining specific acceptable tolerances for specific settings, measuring for compliance, and automatically adapting to deviations, a closed loop network can be created that dynamically and automatically responds to environmental changes. We can apply closed loop control for a variety of use cases including resource optimization (traffic engineering), verification (forwarding behavior), security (DDoS mitigation), and others. In particular, in collaboration with the Pronto™ project, a microburst mitigation mechanism has been implemented in order to stop attackers from filling up switch queues in an attack attempting to disrupt mission critical traffic.

SDN, White Boxes, and Open Source

SD-Fabric is based on a purist implementation of SDN in both control and data planes. When coupled with open source, this approach enables faster development of features and greater flexibility for operators to deploy only what they need and customize/optimize the features the way they want. Furthermore, SDN facilitates the centralized configuration of all network functionality, and allows network monitoring and troubleshooting to be centralized as well. Both are significant benefits over traditional box-by-box networking and enable faster deployments, simplified operations, and streamlined troubleshooting.

The use of white box (bare metal) switching hardware from ODMs significantly reduces CapEx costs when compared to products from OEM vendors. By some accounts, the cost savings can be as high as 60%. This is typically due to the OEM vendors amortizing the cost of developing embedded switch/router software into the price of their hardware.

Finally, open source software allows network operators to develop their own applications and choose how they integrate with their backend systems. And open source is considered more secure, with ‘many eyes’ making it much harder for backdoors to be intentionally or unintentionally introduced into the network.

Such unfettered ability to control timelines, features and costs compared to traditional network fabrics makes SD-Fabric very attractive for operators, enterprises, and government applications.

Extensible APIs

People usually think of a network fabric as an opaque pipe where applications send packets into the network and hope they come out the other side. Little visibility is provided to determine where things have gone wrong when a packet doesn’t make it to its destination. Network applications have no knowledge of how the packets are handled by the fabric.

With the SD-Fabric API, network applications have full visibility and control over how their packets are processed. For example, a delay-sensitive application has the option to be informed of the network latency and instruct the fabric to redirect its packet when there is congestion on the current forwarding path. Similarly, the API offers a way to associate network traffic with a network slice, providing QoS guarantees and traffic isolation from other slices. The API also plays a critical role in closed loop control by offering a programmatic way to dynamically change the packet forwarding behavior.

At a high level, SD-Fabric’s APIs fall into four major categories: configuration, information, control, and OAM.

- **Configuration:** APIs let users set up SD-Fabric features such as VLAN information for bridging and subnet information for routing.
- **Information:** APIs allow users to obtain operation status, metrics, and network events of SD-Fabric, such as link congestion, counters, and port status.
- **Control:** APIs enable users to dynamically change the forwarding behavior of the fabric, such as drop or redirect the traffic, setting QoS classification, and applying network slicing policies.
- **OAM:** APIs expose operational and management features, such as software upgrade and troubleshooting, allowing SD-Fabric to be integrated with existing orchestration systems and workflows.

Edge-Cloud Ready

SD-Fabric adopts cloud native technologies and methodologies that are well developed and widely used in the computing world. Cloud native technologies make the deployment and operation of SD-Fabric similar to other software deployed in a cloud environment.

Kubernetes Integration

Both control plane software (ONOS™ and apps) and, importantly, data plane software (Stratum™), are containerized and deployed as Kubernetes services in SD-Fabric. In other words, not only the servers but also the switching hardware identify as Kubernetes 'nodes' and the same processes can be used to manage the lifecycle of both control and data plane containers. For example, Helm charts can be used for installing and configuring images for both, while Kubernetes monitors the health of all containers and restarts failed instances on servers and switches alike.

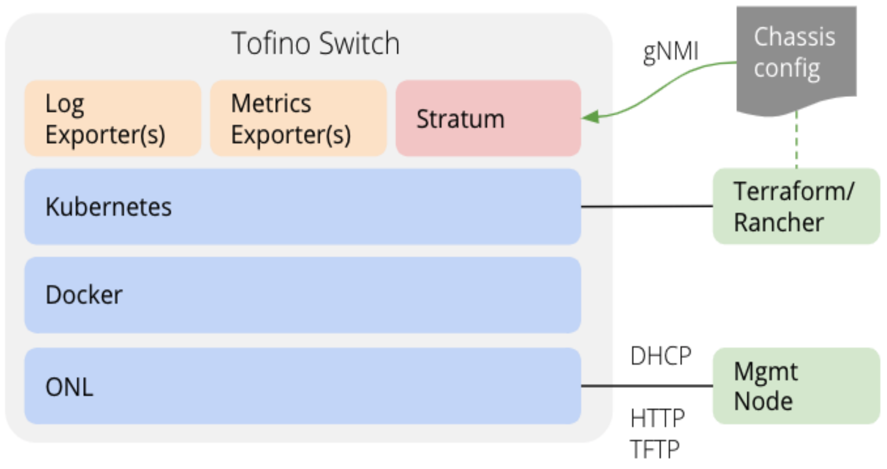

Figure 4: Kubernetes Integration

Configuration, Logging, and Troubleshooting

SD-Fabric reads all configurations from a single repository and automatically applies appropriate config to the relevant components. In contrast to traditional embedded networking, there is no need for network operators to go through the error-prone process of configuring individual leaf and spine switches. Similarly, logs of each component in SD-Fabric are streamed to an EFK stack (ElasticSearch, Fluentbit, Kibana) for log preservation, filtering and analysis. SD-Fabric offers a single-pane-of-glass for logging and troubleshooting network state, which can further be integrated with operator's backend systems.

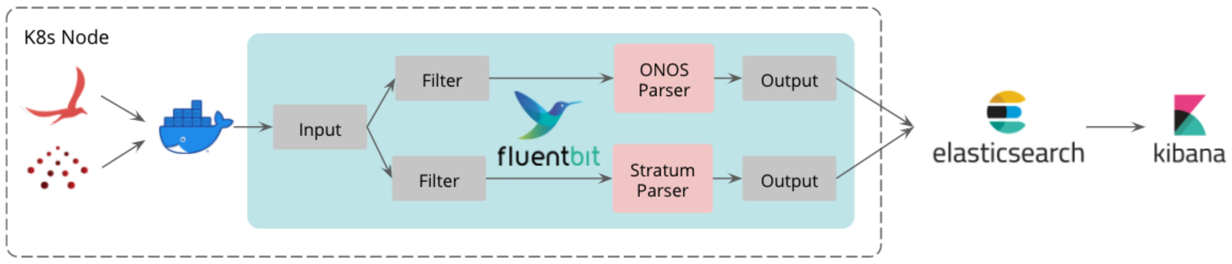

Figure 5: Logging Integration with EFK Stack

Monitoring and Alerts

SD-Fabric continuously monitors system metrics such as bandwidth utilization and connectivity health. These metrics are streamed to Prometheus and Grafana for data aggregation and visualization. Additionally, alerts are triggered when metrics meet predefined conditions. This allows the operators to react to certain network events such as bandwidth saturation even before the issue starts to disrupt user traffic.

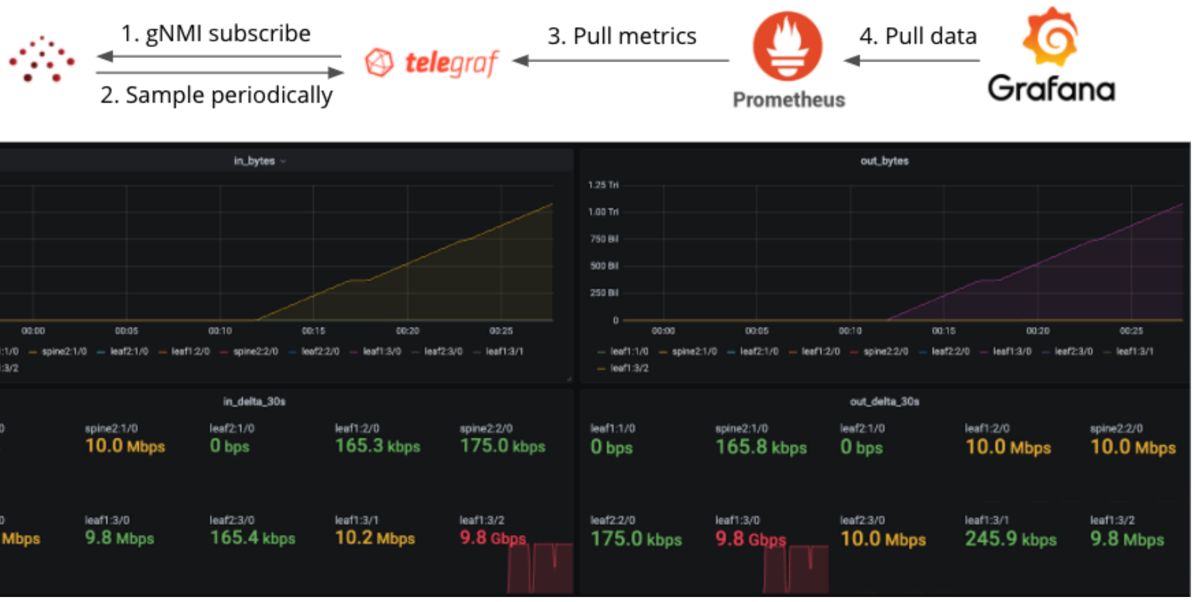

Figure 6: Monitoring Integration with Prometheus and Grafana

Deployment Automation

SD-Fabric utilizes a CI/CD model to manage the lifecycle of the software, allowing developers to make rapid iterations when introducing a new feature. New container images are generated automatically when new versions are released. Once the hardware is in place, a complete deployment of the entire SD-Fabric stack can be pushed from the public cloud with a single click fabric-wide in less than two minutes.

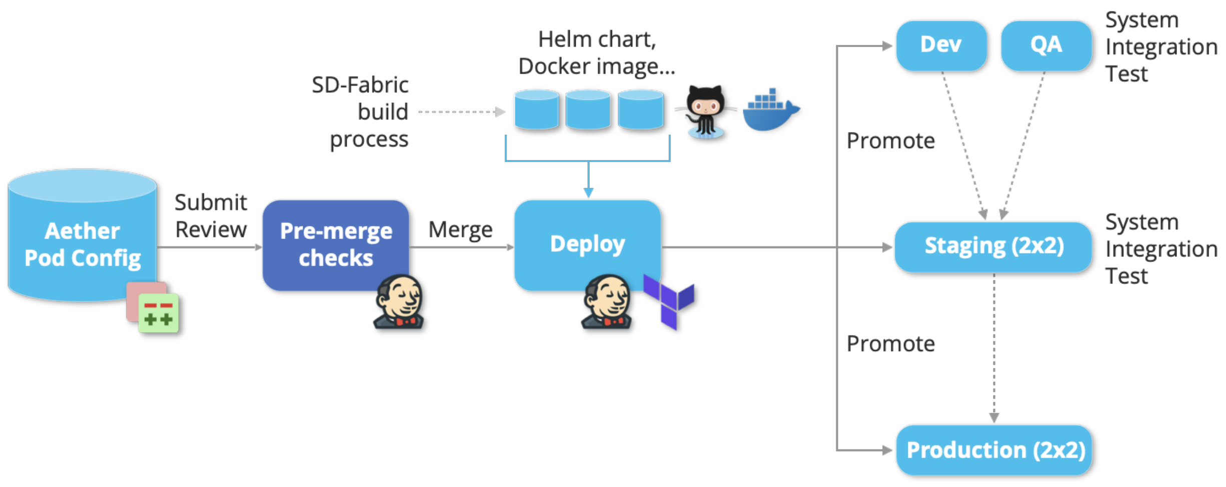

Figure 7: Deployment Automation and CI/CD Pipeline

Aether™-Ready

SD-Fabric fits into a variety of edge use cases. Aether is ONF's private 5G/LTE enterprise edge cloud platform, currently running in a dozen sites across multiple geographies as of early 2021.

Aether consists of several edge clouds deployed at enterprise sites controlled and managed by a central cloud. Each Aether Edge hosts third-party or in-house edge apps that benefit from low latency and high bandwidth connectivity to the local devices and systems at the enterprise edge. Each edge also hosts O-RAN compliant private-RAN control, IoT, and AI/ML platforms, and terminates mobile user plane traffic by providing local breakout (UPF) at the edge sites. In contrast, the Aether management platform centrally runs the shared mobile-core control plane that supports all edges from the public cloud. Additionally, from a public cloud a management portal for the operator and for each enterprise is provided, and Runtime Operation Control (ROC) controls and configures the entire Aether solution in a centralized manner.

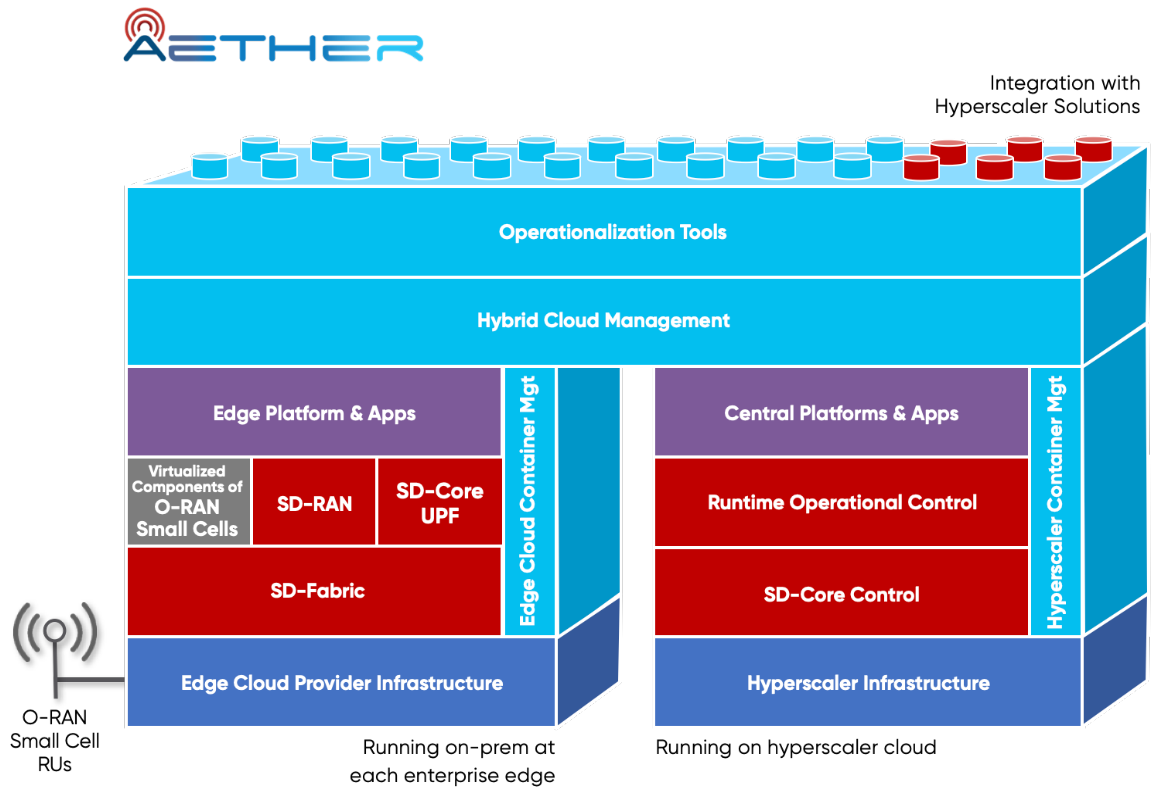
AETHER
Integration with Hyperscaler Solutions
Operationalization Tools
Hybrid Cloud Management
Edge Platform & Apps
Virtualized Components of O-RAN Small Cells
SD-RAN
SD-Core UPF
SD-Fabric
Edge Cloud Container Mgt
Edge Cloud Provider Infrastructure
Central Platforms & Apps
Runtime Operational Control
SD-Core Control
Hyperscaler Container Mgt
Hyperscaler Infrastructure
O-RAN Small Cell RUs
Running on-prem at each enterprise edge
Running on hyperscaler cloud

Figure 8: SD-Fabric in Aether

SD-Fabric has been fully integrated into the Aether Edge as its underlying network infrastructure, interconnecting all hardware equipment in each edge site such as servers and disaggregated RAN components with bridging, routing, and advanced processing like local breakout. It is worth noting that SD-Fabric can be configured and orchestrated via its configuration APIs by cloud solutions, and therefore can be easily integrated with Aether or third party cloud offerings from hyperscalers. In Aether, SD-Fabric configurations are centralized, modeled, and generated by ROC to ensure the fabric configurations are consistent with other Aether components.

In addition to connectivity, SD-Fabric supports a number of advanced services such as hierarchical QoS, network slicing, and UPF idle-mode buffering. And given its native support for programmability, we expect many more innovative services to take advantage of SD-Fabric over time.

Architecture

Classic SDN

SD-Fabric operates as a hybrid L2/L3 fabric. As a pure (or classic) SDN solution, SD-Fabric does not use any of the traditional control protocols typically found in networking, a non-exhaustive list of which includes: STP, MSTP, RSTP, LACP, MLAG, PIM, IGMP, OSPF, IS-IS, Trill, RSVP, LDP and BGP. Instead, SD-Fabric uses an SDN Controller (ONOS) decoupled from the data plane hardware to directly program ASIC forwarding tables in a pipeline defined by a P4 program. In this design, a set of applications running on ONOS program all the fabric functionality and features, such as Ethernet switching, IP routing, mobile core user plane, multicast, DHCP Relay, and more.

Topologies

SD-Fabric supports a number of different topological variants. In its simplest instantiation, one could use a single leaf or a leaf-pair to connect servers, external routers, and other equipment like access nodes or physical appliances (PNFs). Such a deployment can also be scaled horizontally into a leaf-and-spine fabric (2-level folded Clos), by adding 2 or 4 spines and up to 10 leaves in single or paired configurations. Further scale can be achieved by distributing the fabric itself across geographical regions, with spine switches in a primary central location, connected to other spines in multiple secondary (remote) locations using WDM links. Such 4-level topologies (leaf-spine-spine-leaf) can be used for backhaul in operator networks, where the secondary locations are deeper in the network and closer to the end-user. In these configurations, the spines in the secondary locations serve as aggregation devices that backhaul traffic from the access nodes to the primary location which typically has the facilities for compute and storage for NFV applications.

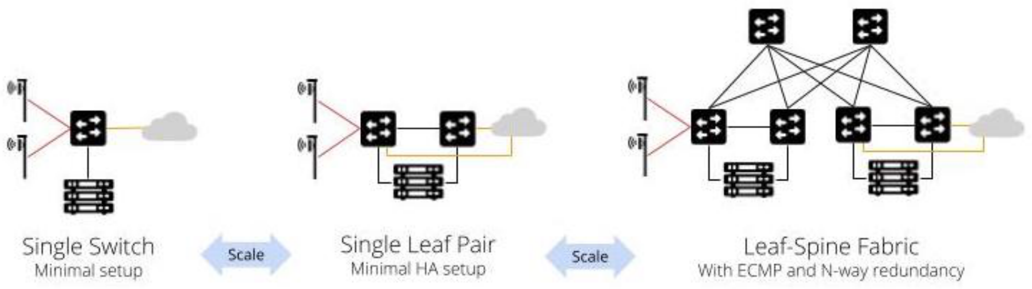

Figure 9: SD-Fabric Scales as Data Center Grows

Redundancy

SD-Fabric supports redundancy at every level. A leaf-spine fabric is redundant by design in the spine layer, with the use of ECMP hashing and multiple spines. In addition, SD-Fabric supports leaf pairs, where servers and external routers can be dual-homed to two ToRs in an active-active configuration. In the control plane, some SDN solutions use single instance controllers, which are single points of failure. Others use two controllers in active backup mode, which is redundant, but may lack scale as all the work is still being done by one instance at any time and scale can never exceed the capacity of one server. In contrast, SD-Fabric is based on ONOS, an SDN controller that offers N-way redundancy and scale. An ONOS cluster with 3 or 5 instances are all active nodes doing work simultaneously, and failure handling is fully automated and completely handled by the ONOS platform.

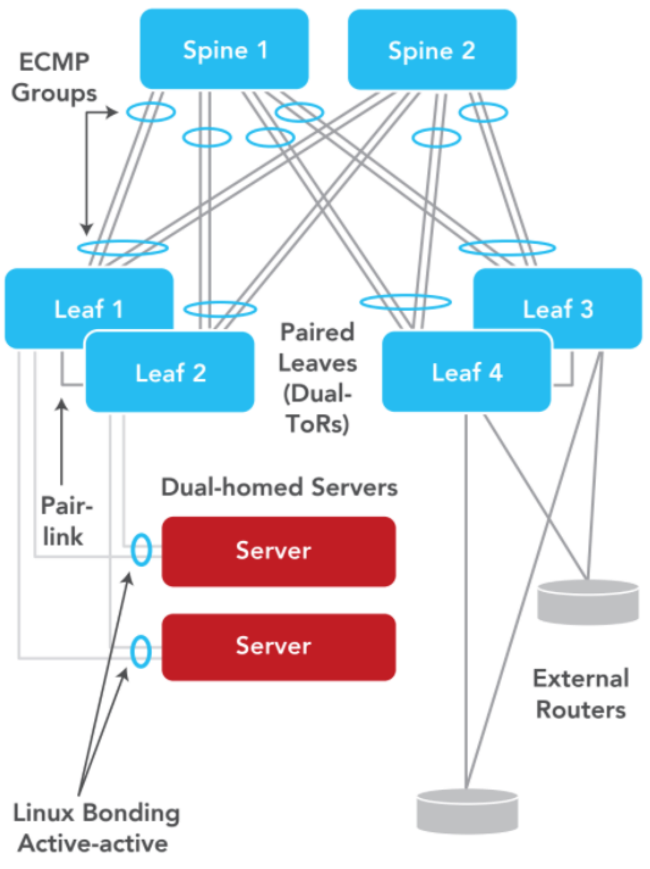

Figure 10: Redundancy Throughout the Network

MPLS Segment Routing (SR)

While SR is not an externally supported feature, SD-Fabric architecture internally uses concepts like globally significant MPLS labels that are assigned to each leaf and spine switch. The leaf switches push an MPLS label designating the destination ToR (leaf) onto the IPv4 or IPv6 traffic, before hashing the flows to the spines. In turn, the spines forward the traffic solely on the basis of the MPLS labels. This design concept, popular in IP/MPLS WAN networks, has significant advantages. Since the spines only maintain label state, it leads to significantly less programming burden and better scale. For example, in one use case the leaf switches may each hold 100K+ IPv4/v6 routes, while the spine switches need to be programmed with only 10s of labels! As a result, completely different ASICs can be used for the leaf and spine switches; the leaves can have bigger routing tables and deeper buffers while sacrificing switching capacity, while the spines can have smaller tables with high switching capacity.

System Components

Open Network Operating System (ONOS)

SD-Fabric uses ONF's Open Network Operating System (ONOS) as the SDN controller. ONOS is designed as a distributed system, composed of multiple instances operating in a cluster, with all instances actively operating on the network while being functionally identical. This unique capability of ONOS simultaneously affords high availability and horizontal scaling of the control plane. ONOS interacts with the network devices by means of pluggable southbound interfaces. In particular, SD-Fabric leverages P4Runtime™ for programming and gNMI for configuring certain features (such as port speed) in the fabric switches. Like other SDN controllers, ONOS provides several core services like topology discovery and end point discovery (hosts, routers, etc. attached to the fabric). Unlike any other open source SDN controller, ONOS delivers these core services in a distributed way over the entire cluster, such that applications running in any instance of the controller have the same view and information.

ONOS Applications

SD-Fabric uses a collection of applications that run on ONOS to provide the fabric features and services. The main application responsible for fabric operation handles connectivity features according to SD-Fabric architecture, while other apps like DHCP relay, AAA, UPF control, and multicast handle more specialized features. Importantly, SD-Fabric uses the ONOS Flow-

Objectives API, which allows applications to program switching devices in a pipeline-agnostic way. By using Flow-Objectives, applications can be written without worrying about low-level pipeline details of various switching chips. The API is implemented by specific device drivers that are aware of the pipelines they serve and can thus convert the application's API calls to device-specific rules. In this way, the application can be written once, and adapted to pipelines from different ASIC vendors.

Stratum

SD-Fabric integrates switch software from the ONF Stratum project. Stratum is an open source silicon-independent switch operating system. Stratum implements the latest SDN-centric northbound interfaces, including P4, P4Runtime, gNMI/OpenConfig, and gNOI, thereby enabling interchangeability of forwarding devices and programmability of forwarding behaviors. On the southbound interface, Stratum implements silicon-dependent adapters supporting network ASICs such as Intel Tofino, Broadcom™ XGS® line, and others.

Leaf and Spine Switch Hardware

In a typical configuration, the leaf and spine hardware used in SD-Fabric are typically Open Compute Project (OCP)™ certified switches from a selection of different ODM vendors. The port configurations and ASICs used in these switches are dependent on operator needs. For example, if the need is only for traditional fabric features, a number of options are possible - e.g., Broadcom StrataXGS ASICs in 48x1G/10G, 32x40G/100G configurations. For advanced needs that take advantage of P4 and programmable ASICs, Intel Tofino or Broadcom Trident 4 are more appropriate choices.

ONL and ONIE

The SD-Fabric switch software stack includes Open Network Linux (ONL) and Open Network Install Environment (ONIE) from OCP. The switches are shipped with ONIE, a boot loader that enables the installation of the target OS as part of the provisioning process. ONL, a Linux distribution for bare metal switches, is used as the base operating system. It ships with a number of additional drivers for bare metal switch hardware elements (e.g., LEDs, SFPs) that are typically unavailable in normal Linux distributions for bare metal servers (e.g., Ubuntu).

Docker/Kubernetes, Elasticsearch/Fluentbit/Kibana, Prometheus/Grafana
While ONOS/Stratum instances can be deployed natively on bare metal servers/switches, there are advantages in deploying ONOS/Stratum instances as containers and using a container management system like Kubernetes (K8s). In particular, K8s can monitor and automatically reboot lost controller instances (container pods), which then rejoin the operating cluster seamlessly. SD-Fabric also utilizes widely adopted cloud native technologies such as Elastic/Fluentbit/Kibana for log preservation, filtering and analysis, and Prometheus/Grafana for metric monitoring and alert.

Conclusion

SD-Fabric is an open source, full stack P4-programmable network fabric optimized for distributed edge networks, 5G, and emerging Industry 4.0 applications. It utilizes SDN and cloud native principles as a foundation.

SD-Fabric supports the creation of customized edge clouds, exposing programmable network resources via SaaS APIs that enable developers to build advanced, complex edge applications and at the same time reduce CPU and compute power. Application functionality can be moved from CPUs into network switches, thereby improving performance while reducing both cost and footprint.

SD-Fabric is deployed in a production Aether network that is part of project Pronto where the programmability of SD-Fabric enables fine-grained measurement, network verification, and closed loop control via simple APIs, creating a resilient, reliable, and secure 5G network.

Learn More

SD-Fabric Web Page: https://opennetworking.org/sd-fabric/

Aether Website: https://aetherproject.org/

SD-Fabric Wiki: https://wiki.opennetworking.org/x/HABGKQ

SD-Fabric Email List: https://groups.google.com/a/opennetworking.org/g/sdfabric-announce

Abbreviations

BESS	Berkeley Extensible Software Switch
INT	In-band Network Telemetry
GTP-U	GPRS Tunnelling Protocol User Plane
LPM	Longest-Prefix Match
ONIE	Open Network Install Environment
ONL	Open Network Linux
ONOS	Open Network Operating System™
P4RT	P4Runtime™
PFCP	Packet Forwarding Control Protocol
QoS	Quality of Service
ToR	Top-of-Rack
UPF	User Plane Function

About ONF

The Open Networking Foundation (ONF) is an operator-led consortium spearheading disruptive network transformation. Now the recognized leader for open-source solutions for operators, the ONF first launched in 2011 as the standard bearer for Software Defined Networking (SDN). Led by its operator partners AT&T, China Unicom, Deutsche Telekom, Google, NTT Group and Turk Telekom, the ONF is driving vast transformation across the operator space. For further information visit http://www.opennetworking.org

Appendix: SD-Fabric Specifications

Features	Description
SDN Features	• ONOS cluster of all-active N instances affording N-way redundancy and scale, where N = 3 or N = 5. • Unified operations interface (GUI/REST/CLI). • Centralized configuration: all configuration is done on the controller instead of each individual switch. • Centralized role-based access control (RBAC). • Automatic host (end-point) discovery: attached hosts, access-devices, appliances (PNFs), routers, etc. based on ARP, DHCP, NDP, etc. • Automatic switch, link and topology discovery and maintenance (keep-alives, failure recovery).
L2 Features	Various L2 connectivity and tunneling support: • VLAN-based bridging: o Access, Trunk and Native VLAN support. • VLAN cross connect: o Forward traffic based on outer VLAN id. o Forward traffic based on outer and inner VLAN id (QinQ). • Pseudowire: o L2 tunneling across the L3 fabric. o Support tunneling based on double tagged and single tagged traffic. o Support VLAN translation of outer tag.
L3 Features	IP connectivity: • IPv4 and IPv6 unicast routing (internal use of MPLS Segment Routing). • Subnetting configuration on all non-spine facing leaf ports; no configuration required on any spine port. • IPv6 router advertisement. • ARP, NDP, IGMP handling. • Number of flows in spines greatly simplified by MPLS Segment Routing.

	• Further reduction of per-leaf flows with route optimization logic.
DHCP Relay	DHCP L3 relay: • DHCPv4 and DHCPv6. • DHCP server either directly attached to fabric leaves, or indirectly connected via upstream router. • DHCP client directly either attached to fabric leaves, or indirectly connected via LDRA. • Multiple DHCP servers for HA.
vRouter	vRouter presents the entire SD-Fabric as a single router (or dual-routers for HA), with a disaggregated control/data plane. • Uses open source protocol implementations like Quagga (or FRR). • BGPv4 and BGPv6. • Static routes. • Route blackholing. • ACLs based on port, L2, L3 and L4 headers.
Multicast	Centralized multicast tree computation, programming and management: • Support both IPv4 and IPv6 multicast. • Dual-homed multicast sinks for HA. • Multiple multicast sources for HA.
APIs	• Provide easy access for 3rd party edge application developers and for the Aether centralized management platform. • Support for traffic redirecting, dropping, network slicing and QoS.
Programmability	• Support for Stratum, P4Runtime and gNMI and P4 programs. • Innovative services enabled by programmable pipeline: ○ 4G/5G UPF - GTP encap/decap, idle-mode buffering, QoS and more. ○ BNG - PPPoE, anti-spoofing, accounting and more.

Visibility	<ul><li>In-band Network Telemetry (INT) support:<ul><li>Integrated with Intel DeepInsight.</li><li>Smart filters and change detectors to reduce volume of INT reports.</li></ul></li></ul>
Troubleshooting & Diagnostics	<ul><li>T3: Troubleshooting tool to diagnose broken forwarding paths fabric wide.</li><li>ONOS-diags: One-click Diagnostics collection tool.</li></ul>
Topology	<ul><li>Single leaf (ToR) or dual-ToR (dual-homing).</li><li>Supports typical leaf-spine topology, 2 to 4 spines, up to 10 leaves.</li><li>Multi-stage leaf-spine fabric (leaf-spine-spine-leaf).</li><li>Can start at the smallest scale (single leaf) and grow horizontally.</li></ul>
Resiliency	Provides HA in following scenarios:<ul><li>Controller instance failure (requires 3 or 5 node ONOS cluster).</li><li>Link failures.</li><li>Spine failure.</li></ul>Further HA support in following failure scenarios with dual-homing enabled:<ul><li>Leaf failure.</li><li>Upstream router failure.</li><li>Host NIC failure.</li></ul>
Scalability	<ul><li>Up to 50k routes, 110k flows, 8 leaf, 2 spines, with route optimization enabled, ASIC-dependent (in production).</li><li>Up to 120k routes, 250k flows, 8 leaf, 2 spines, with route optimization enabled, ASIC-dependent (in pre-production).</li></ul>
Security	<ul><li>TLS-secured connection between controllers and switches.</li><li>AAA 802.1x authentication.</li></ul>

Aether-Ready	Fully integrated with Aether (5G/LTE private enterprise edge cloud solution) including deployment automation, CI/CD, logging, monitoring, and alerting.
Overlay Support	Can be used/integrated with 3rd party overlay networks (e.g., OpenStack Neutron, Kubernetes CNI).
Orchestrator Support	Can be integrated with an external orchestrator, optionally running from the public cloud. Supports logging, telemetry, monitoring and alarm services via REST APIs and Elastic/Fluentbit/Kibana, Prometheus/Grafana.
Controller Server Specs	Recommended (per Kubernetes node): • CPU : 32 Cores. • RAM: 128GB RAM. 64GB dedicated to ONOS JVM heap (based on 50K routes).
White Box Switch Hardware	• Multi-vendor: APS Networks™, Dell™, Delta Networks™, Edgecore Networks™, Inventec™, Netburg™, QCT™. • Multi-chipset: ○ Intel Tofino (supports all features, including programmability, UPF & INT). ○ Broadcom Tomahawk®, Tomahawk+®, Trident2 (traditional fabric features only). • 1/10G, 25G, 40G, 100G. • Refer to https://github.com/stratum/stratum#supported-devices for the most up-to-date hardware list.
White Box Switch Software	• Open source ONL, ONIE, Docker, Kubernetes. • Stratum available from ONF.