Understanding SDN

A Survey of SDN Technologies and Associated Use Cases

Glenn Dasmalchi
gdasmalchi@strategize.consulting

v1.0 - December, 2014

Introduction:

Software Defined Networking (SDN) continues to be **the** dominant topic of interest in networking today. Despite a continued lack of clarity on what SDN is, and where it might be useful, it's a topic that has captured the attention of customers and the IT community.

The SDN industry conversation has become broad, and in a sense "hyped" with many parties attempting to guide the SDN definition to suit their purposes. But while positioning will no doubt continue, at its core SDN has come to represent a new way of designing, integrating, and operating networks to address a variety of customer imperatives and use cases.

The goal of this report is to help frame the industry conversation around SDN, starting with a working definition and technology-based categorization model. Each category is then described, along with associated use cases and industry activity. The report concludes with additional key use cases (that span SDN categories), and how software-defined networks "as a whole" are being integrated into larger IT-level systems.

The intent is for the reader to become familiar with the myriad of approaches being called *SDN*, but also to understand *why* they are important (via use cases). While the organization of this report is based on technology categorization, the *motivation* for understanding SDN lies in the new IT-level architectures it enables. To that end, the ultimate goal of this report is to show how SDN technologies enable networks with the flexibility, efficiency, and dynamism needed by new styles of IT.

SDN: *What* and *Why*?

SDN has no precise definition that is widely accepted. Still, from a technology perspective there are architectural principles and implementation approaches often associated with the topic.

The **architectural principles** are:

1. Separation of network *control* and *forwarding* functions
2. *Logically centralized* network control elements (e.g. "controllers")
3. *Programmable* interfaces for the network (at multiple levels)

SDN **implementations** typically have the following characteristics:

- Network controllers are independent software elements.
- System elements use *open, standardized* interfaces.

While there is room for interpretation in what constitutes "SDN", the principles and characteristics above constitute a good working definition by which to discuss SDN technology approaches.

Definition aside, *interest* in SDN has been driven by the promise of:

- Better economics (both CapEx and OpEx)
- Increased "feature velocity"
- Increased operational agility (automated or operator-driven)
- Easier integration with higher-layer software (e.g. applications, orchestrators)

This is a broad and general list of potential benefits, and the value that individual customers place on them will vary. In addition it's still early days in SDN, and many benefits remain unproven. Nevertheless, there is a strong willingness by customers to listen and encourage a movement that promises *agility, efficiency, and choice* in networking. Companies such as Google, Facebook, and Amazon have also provided *validation* by developing SDN approaches that demonstrate promised benefits.

As a final point on motivation, it's worth highlighting that in addition to benefiting *traditional* network use cases, SDN is seen as a key enabler for new, dynamic *IT-level* architectures (e.g. multi-tenant cloud, analytics, and DevOps frameworks).

This report presents a categorization and survey of SDN technologies. The intent is to organize and describe the technologies along with associated benefits, use cases, and the ecosystem driving them in the market.

The hope is that this information will serve as useful background for anyone wishing to understand the overall *SDN movement* from a technology perspective.

Surveying SDN Technologies

The industry has not yet proposed (let alone *settled on*) a categorization framework for the ever-expanding list of SDN technologies. In this report, we propose that the various approaches can be considered in three main categories:

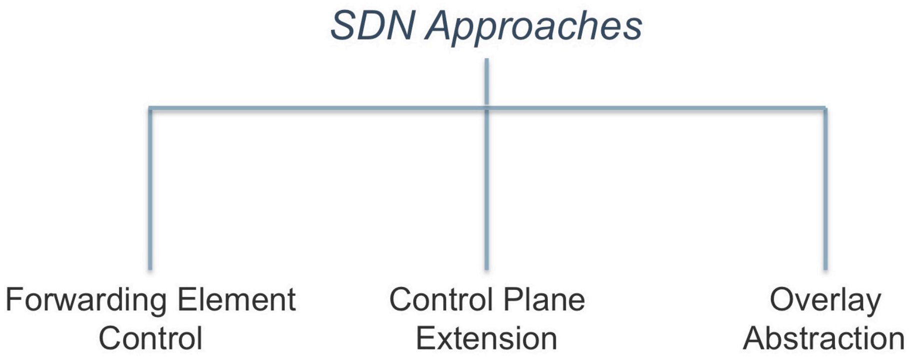


Figure 1 - SDN Categorization Framework

1. **Forwarding Element Control** - in which forwarding capabilities of individual network elements are centrally controlled and managed.

2. **Control Plane Extension** - in which existing network control functions (e.g. routing) are extended via external software elements.

3. **Overlay Abstraction** – in which new network abstractions are created (via centralized control) on top of existing network capability.

This categorization is just a model[1], but as described in the following sections it allows us to organize the various technologies that the industry has associated with SDN.

Per the principles and approaches introduced in the previous section, each category includes an external *software* control element that:

- is central to the network control mechanism, and
- defines and uses open interfaces (e.g. protocols, APIs) with elements in the network[2]

This report now examines each category, describing: technical approach, illustrative use cases, and related industry and community activity.

[1] *"Essentially, all models are wrong, but some are useful."* – George E.P. Box

[2] And sometimes interfaces with the higher-order IT system on behalf of the network as a whole. See the *Integrating SDN* section later in this report.

Category 1: SDN via Forwarding Element Control (OpenFlow)

This category is defined by *centralized control* of traffic forwarding on network elements (e.g. switches, routers).

In the "early days" of SDN[3] (just a few years ago), the focus was on separating control plane functionality from individual network elements (e.g. switches, routers). Initial interest centered on OpenFlow, an interface/protocol between a logically centralized *controller* and forwarding elements. OpenFlow includes a set of very low-level, granular instructions that a controller uses to "program" tables in the forwarding elements (the forwarding elements are known collectively as the *forwarding* or *data plane* – DP). These table entries determine how the DP elements handle traffic per flow[4].

In addition to separation from the DP, an OpenFlow controller also has expanded *scope*, coordinating control over all DP elements that constitute a cohesive "network". Figure 2 shows this overall architecture, along with examples of vendors and organizations that made up the initial OpenFlow ecosystem (circa 2012).

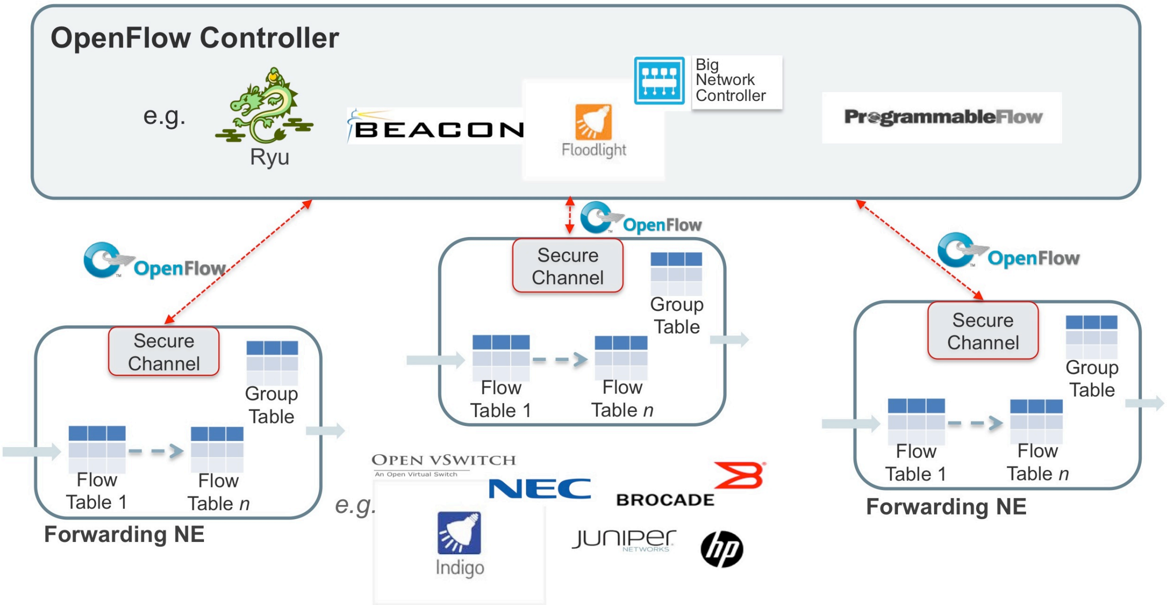


Figure 2 - Initial OpenFlow Architecture and Ecosystem Examples

[3] This report doesn't attempt to "reach back" to technologies that many would claim to be SDN-like (e.g. ATM). For whatever reason, those attempts didn't spark the interest or scope of discussion that "SDN" is now driving. They were no doubt the source of many ideas that SDN has leveraged, but that's for another discussion.

[4] A flow is generally defined as a sequence of packets from a source to a destination. Network elements recognize a flow by matching on a combination of header field (addresses, ports, etc.).

At this point it's useful to understand the original motivation for this initial SDN technology, and why it continues to be a topic of interest. There are several aspects:

- OpenFlow originated in academia, driven by network researchers who desired granular control of the network to support **network research** (e.g. new protocols, programmable frameworks). A key goal was establishing a common framework with the *critical mass* to support a research (and supporting industry) ecosystem.

- Some (primarily in academia) viewed OpenFlow (or some evolved version of it) as a new "**intellectual foundation**" for networking – the basis of a true discipline. Their claim is that traditional networking's evolution of layered, complex protocols limits the ability to extend, operate and automate networks. The claim is that the right foundational primitives and control capability could serve as the basis for a networking development "tool chain" similar to what exists in the compute domain (e.g. *network compilers, debuggers*).

- Certain customers see OpenFlow as a counter to network HW vendor dependency. The resulting **vendor independence** would yield price leverage (including commoditization at the extreme).

- There is a belief that network **feature velocity** could also be improved, either by sophisticated customers able to consume OpenFlow directly – or via an active ecosystem that would compete to provide features based on an OpenFlow foundation.

The OpenFlow standard is governed by a consortium called the Open Networking Foundation[5] (ONF). As a user-driven organization, its board members consist of major network operators (*telcos*: DT, NTT, Verizon; *web*: Google, Facebook, Yahoo, Microsoft; *FSI*: Goldman Sachs) as well as Stanford University and UC Berkeley. There are no vendors on the board, however many vendors are members in a supporting capacity. The ONF recognizes a broader SDN architectural scope, but believes it should be anchored in support for OpenFlow. This view of OpenFlow being a *fundamental* part of any SDN system is not generally accepted in the industry (especially as other SDN technologies have emerged).

But OpenFlow is the technology that started the SDN movement, and it continues to be a popular topic. It meets the criteria for SDN previously described and is an open, evolving standard for controlling network elements (e.g. switches, routers). Several vendors provide OpenFlow support on network elements[6], and there are several controllers available that support OpenFlow.

[5] www.opennetworking.org

[6] OpenFlow client functionality can also co-exist with traditional control plane capabilities (aka hybrid mode). E.g. certain ports/VLANs controlled via OF, with others controlled "traditionally"

OpenFlow Use Cases

Regardless of initial motivations and standardization efforts for OpenFlow, its significance will ultimately depend upon the use cases it enables or optimizes. The following are sample OpenFlow use cases based on customer interest and experience:

- Network Visibility: In the data center, some (e.g. Financial Services) customers are interested in leveraging OpenFlow as an open, cost-effective way to TAP/SPAN traffic of interest in the network (e.g. for diagnostics and compliance). This could potentially eliminate special purpose devices used today.

- Traffic Shedding: Redirect traffic based on some match criteria to lessen the load on subsystems that would otherwise become a bottleneck. For example, redirect known Netflix video traffic to bypass application firewalls.

- Bulk Data Transfer: Steer traffic in a TCP session across multiple, (bandwidth-limited) paths. The compelling use case is data transfer over distance, with OpenFlow used to program disjoint paths. This works in conjunction with Multipath TCP (MPTCP) on the end-points.

- Centralized Traffic Engineering: Google used OpenFlow to control WAN traffic between their internal data centers. This internal network (a.k.a. G-Scale) uses OF-capable dense 10GE devices under centralized control to achieve >90% link utilization in Google's environment.

- Research "Sandbox": This represents multiple use cases to support both *network research* (e.g. protocol development, network-based security frameworks, network "tool chains") and also using the network as an enablement platform for *science research* (e.g. enabling ad hoc distributed analytics platforms across multiple organizations).

OpenFlow is very relevant when a use case needs individual control of traffic *flows*. If flow-level control is not needed, then using OpenFlow directly is akin to programming a higher-level application in assembly language – possible, but cumbersome.

In fairness, later versions of OpenFlow can accommodate more general access to the processing pipeline of data plane elements (e.g. merchant network ASICs). This opens up use cases beyond flow-based networking (as we'll see later in the discussion of network virtualization). This use, though, is more about leveraging OpenFlow as access "mechanism" rather than a network interface "model".

There is an effort, however, to evolve the OpenFlow *model* to be more flexible and capable. The approach is based on a declarative language for specifying network element (e.g. switch) behavior[7]. Switches underpinning the approach could provide programmable parsing and forwarding control, supporting the desired flexibility. The ONF is currently developing this model under a project called *Protocol Independent Forwarding* (which even includes an open-source implementation project to encourage a variety of languages and target devices).

[7] See "P4: Programming Protocol-Independent Packet Processors" arxiv.org/pdf/1312.1719v2.pdf

Category 2: SDN via Control Plane Extension

This category of SDN involves the "opening up" of network functionality and information to external software. External software elements (e.g. controllers, management software) interface to the network control plane to enable new features. This differs from the previous SDN category in that network control planes are *extended*, or augmented, by external software (rather than replaced by a new control mechanism).

Control Plane Extension Technologies

The following are very brief descriptions of sample technologies in this category. Each is a topic in its own right, but these examples describe approach along with motivating use cases.

PCE (Path Computation Element): An architecture that enables multiple elements to participate in dynamically determining and controlling a service's traffic path through an MPLS or GMPLS (optical) network. The elements participating in *path computation* (PCEs) can be a combination of compute elements, network nodes, or management stations. These PCEs use a defined protocol (PCEP) to cooperate and to specify traffic paths in the network. This approach represents an extension to traditional MPLS where paths are normally determined by the head-end router.

The benefit of PCE is that more sophisticated algorithms can be applied to traffic in the network based on additional compute power and information sources. Key use cases include: path determination in optical networks, multi-layered network coordination, *business policy*-based traffic engineering, and traffic engineering across network boundaries currently not supported via other methods.

ALTO (Application-Layer Traffic Optimization): ALTO is a service that allows distributed applications to access resources optimally, based on state in the network. The killer application example is peer-to-peer (P2P) file sharing, where knowledge of network topology and link state/capability can be used to select the best peer for data exchange. From a business perspective, this approach is also often a win-win for both users and providers (e.g. ISPs) since it avoids sub-optimal routes and unnecessary link usage. Other applications that could benefit from ALTO include cache/mirror selection (for content delivery), and P2P live media streaming and communications.

ALTO itself specifies the architecture and protocol that creates an information *resource selection* service between an ALTO server and an application. ALTO servers provide abstracted "maps" of network topology, along with the "cost" to communicate within the topology. However, the raw input and methods by which an ALTO server creates the abstracted network information is outside the scope of ALTO (but see BGP-LS below).

BGP-LS/TE (BGP for Link State and Traffic Engineering information): A mechanism by which link state and traffic engineering information can be collected within networks and shared with external components by leveraging the BGP routing protocol. Applications of this technique include PCE and ALTO, which rely on gathering network topology and capability information as inputs to fulfill their functions.

The information shared by BGP-LS is subject to control by the provider, so different *levels* of information sharing are possible.

I2RS (Interface to the Routing System): An architecture to enable external (e.g. controller) interaction with a routing system (for both visibility and targeted control). Example use cases include: anomaly or intrusion detection, transport (e.g. optical) network status, or business-level cost considerations. I2RS proposes to specify standard, programmatic interfaces to routing-related elements (e.g. RIB). As an enabler for providing useful input, applications would also be able to query state (e.g. topology) from the network. A further claim is that providing programmatic interfaces to the routing system could also make established routing protocols (e.g. BGP) easier to configure and operate.

The previously described technologies either extend existing network capabilities, or provide new levels of network visibility to external software. Not surprisingly, they are all being developed as IETF standards.

However there is also another SDN community/organization incorporating many of these technologies in the form of a reference implementation. *OpenDaylight*[8] is a Linux Foundation open source project that is developing an SDN reference framework and implementation. Their scope is a complete SDN controller-based framework, where multiple SDN technologies are combined. For example, from technologies mentioned so far, the OpenDaylight Controller includes support for OpenFlow, BGP-LS/TE, and PCE.

OpenDaylight efforts are recognition of multiple viable SDN technologies emerging[9].

The Control Plane Extension category of SDN is an important recognition that existing network control mechanisms can be leveraged or augmented to provide new capability for the benefit of applications.

Aside: But wait… isn’t NAC also SDN then?

In a sense, *yes* – in that it implements a specific capability using a “SDN approach”. It’s an access control mechanism/feature that satisfies SDN principles – separate, logically centralized control based on standard interfaces (e.g. 802.1X, RADIUS/Diameter, IF-MAP). But NAC was created before the SDN term existed. Whether customers consider NAC a form of SDN, then, is largely semantics.

Beyond basic network access, though, expanded forms of policy-based network control are being pursued as promising use cases for SDN. See the discussion of group-based policy in a later section.

[8] www.opendaylight.org

[9] In fact, OpenDaylight includes technology approaches from all three SDN Categories in this report.

Category 3: SDN via Overlay Abstraction

This final category builds on the concept of leveraging existing network capability. In this category, though, new network abstractions are built *on top of* an existing network. Think of a separate control and forwarding mechanism "overlaying" an existing network (as opposed to being a *cooperative extension* or *replacement* of the existing network's control plane per previously described categories).

In the Overlay Abstraction category, the *edge overlay fabric* is the approach with the most traction and current customer interest (for data center networks).

Data Center Edge Overlay Fabrics

On-demand, multitenant use case requirements have given rise to a data center network architecture based on a mesh of *overlay tunnels* connecting data center end-devices.

VXLAN is the tunnel (or encapsulation) type most often discussed in this approach[10]. In the design, a mesh of tunnels run on top of an IP network "underlay". Figure 3 is a conceptual view of the approach.

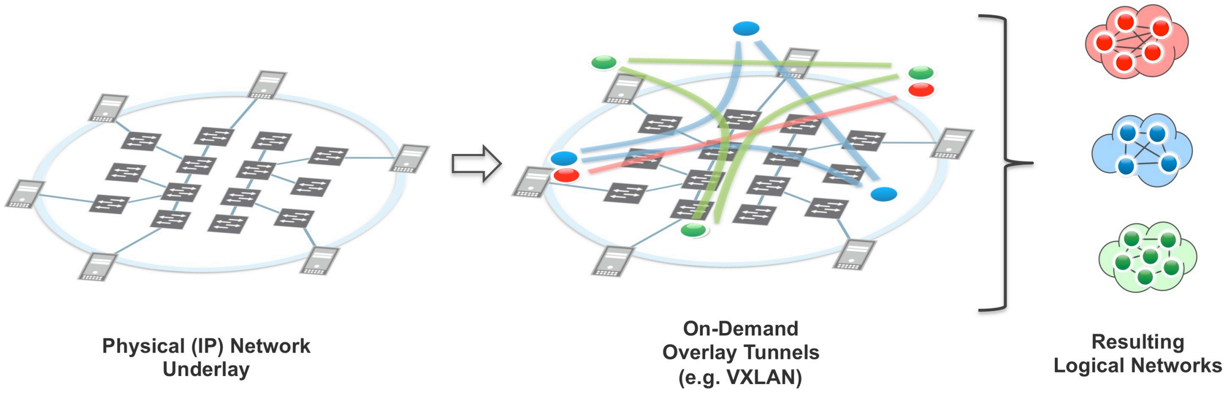


Figure 3 - Edge Overlay Fabric Concept

In this simplified view, physical network infrastructure is overlaid with a set of tunnels that allow VMs (represented by colored dots) to be assigned to specific, segmented logical networks. These logical (a.k.a. virtual) networks support L2 or L3 network services from the standpoint of the attached VM end-points. This simple view shows only VM end-points, but as we'll cover later the approach can also support the gamut of data center devices (e.g. bare metal servers, L4-7 network appliances, storage devices).

[10] VXLAN is just a data plane encapsulation, but as the first one popularly implemented some use the *VXLAN* term to reference the overlay fabric approach in general. In any case, VXLAN is just one of several DP encapsulations used in edge fabric overlays (see later forwarding plane discussion).

Is This the Same as "**Network Virtualization**"?

It's more appropriate to consider *edge overlay fabrics* just one implementation approach to network virtualization (albeit the most popular one in the industry today).

Network Virtualization has historically encompassed a range of technologies. (Think of many network technologies that start with "V" – VPN, VLAN, VRF. And even some that don't – MPLS as a form of link virtualization.) There are also books and papers on "Network Virtualization" that predate the SDN term.

So there is a broader context and history to the term *network virtualization (NV)*.

Network Virtualization -> Virtual Networks

In the context of SDN, though, be aware that many use "network virtualization" (or NV) exclusively to describe the approach of creating multiple logical networks on top of shared physical infrastructure. These logical networks are typically called *virtual networks (VNs)*. This interpretation has the appeal of strong analogy to compute virtualization (i.e. many logical/virtual from one physical).

Even under this more restricted definition (i.e. NV=VN's), *edge fabric overlays* are **still** just one technology approach to creating "virtual networks". As an alternative example, in the current release of OpenDaylight there is a VN implementation (called *VTN* for virtual tenant network) that is based on the OpenFlow model. Another alternative example in the vendor space is Big Switch Networks' *Big Cloud Fabric*, which uses OpenFlow as access mechanism to program forwarding ASICs on bare metal switches in a leaf-spine topology. In fact Big Switch doesn't use the term "network virtualization" (perhaps to avoid terminology confusion) but the solution nevertheless creates logical network segments from shared physical infrastructure.

A Matter of Terminology

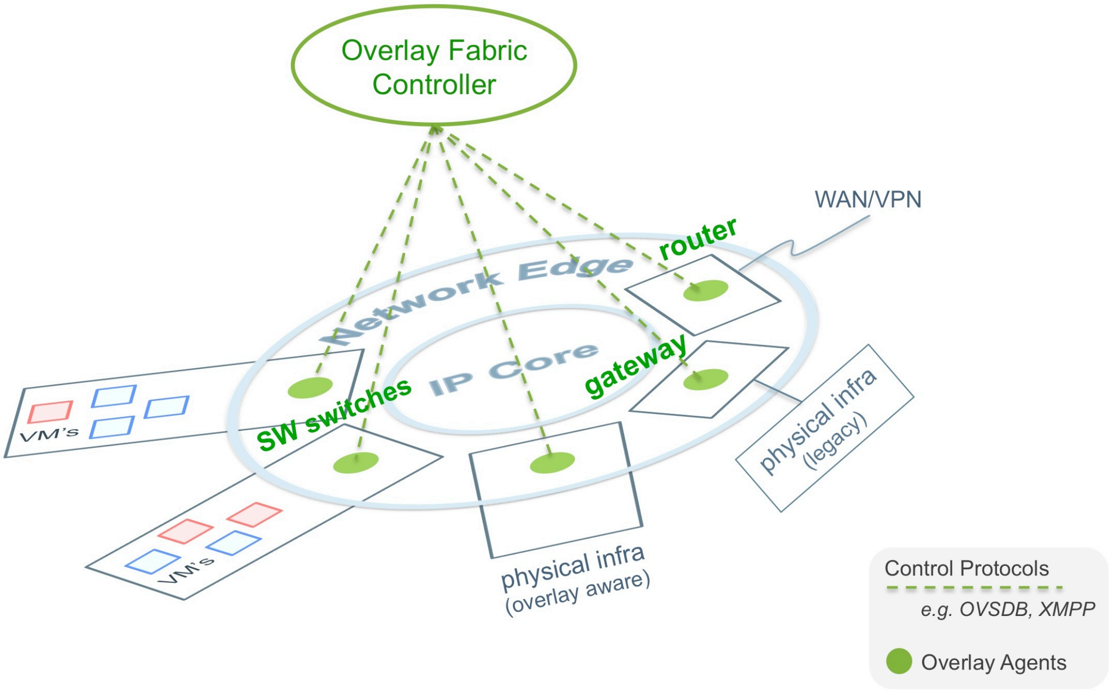
Just be aware that these terms are often used interchangeably in the market (sometimes on purpose by vendors looking to establish mind share and preference). In these cases, you will need to consider further context to understand what is meant.

In short, for purposes of this report we consider *edge overlay fabrics* as just one implementation approach for creating *virtual networks* – and *virtual networks* to be one "construct" within the broader category of *network virtualization*.

That said, *edge overlay fabrics* are the technical basis for many solutions in the market, and a focus in several SDN community efforts. The following sections describe the approach in more detail.

Edge Overlay Fabric Control

Figure 4 is a view of the **control system** elements in a data center edge overlay fabric. Per SDN principles, the approach consists of a centralized controller. The controller interacts with a set of agents (in the network edge) that essentially “bind” various types of end-points to the overlay fabric.



G. Dasmalchi, 2014

Figure 4 - Overlay Fabric Control Elements

Starting on the left in the above figure, hypervisor-based *software switches/routers* manage the connection of VM’s to the overlay fabric. This includes the assignment of VM’s to logical/virtual networks (e.g. represented by the VM box colors above). This also corresponds to the conceptual view in Figure 3.

Physical infrastructure elements (e.g. bare metal servers, firewalls, storage devices) can also integrate into the overlay fabric, supported by agent functionality in device embedded software (or NIC subsystems).

Legacy physical infrastructure (devices or networks with no overlay awareness) can be integrated via proxy *gateways* – typically implemented in: appliances (physical or virtual), ToR switches, or routers.

Finally, edge *router* agents play a key role in accessing end-points external to the fabric (e.g. via VPN), and in extending (or federating) the fabric beyond a single data center.

Edge overlay fabric implementations use a variety of control protocols between the controller and various agents (a single fabric implementation may even use several). By themselves, these control protocols are just "mechanism" leveraged by the larger control plane (e.g. Overlay Fabric Controller).

Overlay Fabrics and **OpenFlow**!?

The logical networks created by the overlay fabric architecture can be thought of as a kind of "logical overlay switch" (where the tunnels carry traffic *flows* between end-points). This means that *OpenFlow* could be used as a management interface to this (higher-level) logical switch (in contrast to OpenFlow's original use case of controlling individual network elements).

In the logical switch case, the use of OpenFlow is less architecturally significant. But it's interesting to see OpenFlow get leveraged to program this higher-level of abstraction (without the requirement that physical network elements natively support it). As an example, Nicira (now VMware) took the approach of using OpenFlow (with extensions) to control its NVP (Network Virtualization Platform). Under VMware, NVP has evolved into the NSX platform and now primarily uses a control protocol called OVSDB.

In another example, NTT's *Ryu* controller can also use OpenFlow to control overlay tunnels (as well as to control individual network elements that support OpenFlow).

The internals of the larger control plane (i.e. how the controller+agents define and run the overlay system) determine *how well* the fabric works. In particular, the internal design of the control plane has a significant effect on non-functional requirements including:

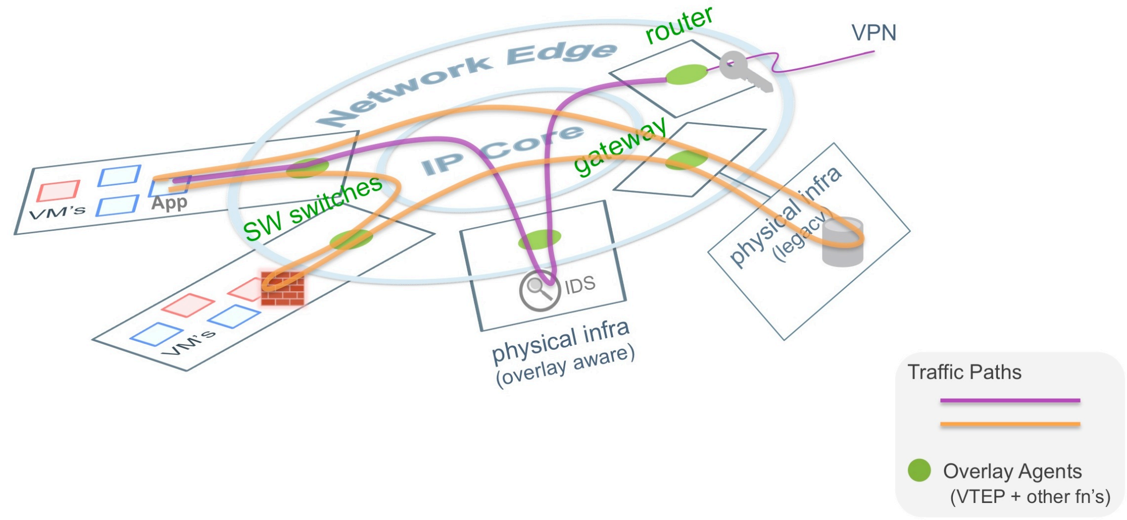
- scalability (e.g. maximum number of end-points, or segments)
- security (e.g. ability to apply security policies to traffic)
- extensibility (e.g. federation across multiple data centers)

At a high level, though, all edge overlay fabric control planes support the ability to define **network segments**, and to assign end-points to those segments (via the agents described earlier) – e.g. VM's to a virtual network. As described in the next section, this forms the basis for how traffic is controlled *within* segments, *between* segments, and *external* to the overlay fabric[11].

[11] This model also lends itself to end-point group-based policy management (a.k.a. GBP) - more on this later in the *Integrating SDN* section.

Edge Overlay Fabric Forwarding

The edge overlay fabric creates a **forwarding model** that provides L2 or L3 connectivity between end-points in a segment (and *isolation by default* from end-points outside the segment). In addition, traffic redirection (or *steering*) between the larger set of fabric end-points is supported. Figure 5 shows examples of this redirection capability.



G. Dasmalchi, 2014

Figure 5 - Overlay Fabric Forwarding Examples

The purple line shows external VPN traffic (destined for a VM-based application) getting redirected through a (physical appliance) Intrusion Detection System. The orange line shows redirection of a VM-based query through a (virtual appliance) firewall.

While not shown explicitly in the figure, an overlay fabric can also support controlled connectivity between end-points in different segments. While default behavior is typically to isolate, flexibility exists to provide connectivity with optional redirection. For example, traffic between VM's running different application tiers (e.g. a red and a blue VM in the figure above) could be allowed, with traffic first directed through a security appliance.

As with the control plane, there are several *data plane protocols (or encapsulations)* used in various overlay fabric implementations. To date VXLAN is the most popular, but others include NVGRE, GRE, and STT. They all provide the same basic capability (outer "tunnel" addresses and the ability to tag traffic to a segment or virtual network). Differences do exist in terms of: ASIC (e.g. header) support, ability to load balance, and ability to offload/accelerate in hardware. But at this point the industry has recognized that the control plane is the main point of differentiation, and there are efforts to move toward a unified, standard data plane encapsulation called "Geneve[12]".

[12] tools.ietf.org/html/draft-gross-geneve-01 Geneve also includes *improvements* – e.g. additional (optional) header fields to carry metadata (for service chaining – see the later section on NFV).

Edge Overlay Fabrics in the Market

The edge overlay fabric approach was pioneered by Nicira (acquired by VMware in 2012). Today many edge overlay fabric solutions exist in the market. These include VMware *NSX* (based in part on Nicira *NVP*), Nuage Networks *VSP*, PLUMgrid *ONS*, Midokura *MidoNet*, Microsoft *HNV*, and Juniper *Contrail*. The main point to note is that it's a popular approach, backed by an interesting mix of vendors.

Edge overlay fabrics have the advantage of working on any IP (underlay) infrastructure. They require minimal physical network integration. This broad applicability, along with easy-to-consume interfaces for creating *virtual networks*, has accelerated adoption of edge overlay fabrics among system and self-integrators. In particular, use cases have centered on enabling multi-tenant cloud platforms (more on use cases later).

A secondary benefit of edge overlay fabrics (or more specifically the data plane encapsulations used) is a larger number of network segments than would be possible using traditional techniques (e.g. VLANs).

Despite the benefits, there is also concern re: the ability of edge overlay fabrics to scale in some dimensions - in particular, the ability to support a large number of end-points (e.g. at "cloud scale"). Part of this depends on how the overlay fabric control plane is implemented; with some overlay fabrics leveraging control plane technologies that have been proven to scale (e.g. BGP). Scalability is still an open question, though, with some advocating that high degrees of scale can only be achieved via non-overlay (a.k.a. *inlay*) techniques.

Another area of interest is how the physical network (underlay) might further contribute to the capabilities of the edge overlay fabric. Efforts to date have focused on "gateways" to integrate legacy infrastructure end-points, and access to resources outside the fabric. Federating fabrics across multiple data centers is also an area of interest (e.g. for multi-DC *distributed or hybrid clouds*). Here again, the overlay control planes that leverage BGP can more easily integrate with existing VPN services (as a scalable, tested foundation for extending the fabric).

Finally, capabilities of the physical network *underlay* do affect capabilities of the overall system. And even without coordination, simply running an overlay fabric on a highly capable physical underlay yields a more capable system. For example, a "flat" physical underlay with low, deterministic latency will improve the latency characteristics of the virtual networks "carved out" of it. This may sound obvious, but to-date hasn't received much visibility in SDN discussions[13].

Additional *coordination* (ideally standards-based) between edge overlay fabrics and physical underlays could go beyond gateway functionality to enable better: *visibility* (e.g.

[13] It's important to note, though, that non-differentiated substrates (ones that just provide basic connectivity) will be "good enough" for some use cases.

for troubleshooting, billing, more intelligent placement of VMs based on network metrics), *performance* (e.g. via network TE, QoS, multicast, etc.), and *security*.

Edge overlay fabrics are the central topic in the *IETF's NVO3* (Network Virtualization Overlays) working group.

A Final Note on SDN Technology Categorization

The categorization in this report is simply a way to organize and present a myriad of technologies that can all be considered "SDN". In some cases, it was easy to map a technology to a specific use case (e.g. OpenFlow and network "taps"). In general, though, it's often a *choice* of multiple technologies (as discussed in the next section).

Accordingly, nothing in this report is meant to suggest that a category (or particular set of technologies) will "win out". They will all evolve based on a variety of factors, hopefully including their utility in use cases that matter to customers.

Category Spanning SDN Use Cases

Use cases are the critical linkage between technology and business value. While some use cases are a natural "match" for specific SDN technologies, others can be addressed by multiple, alternative technology approaches (or combinations thereof).

Virtual Networks

The most popular intermediate use case in SDN today is *Virtual Networks* (a subset of the Network Virtualization topic). However virtual networks themselves are just constructs that serve a higher-level use case. Among those higher-level use cases, **cloud** is the most popular. Figure 6 is a simplified view of the relationship between SDN technologies, virtual networks, and cloud.

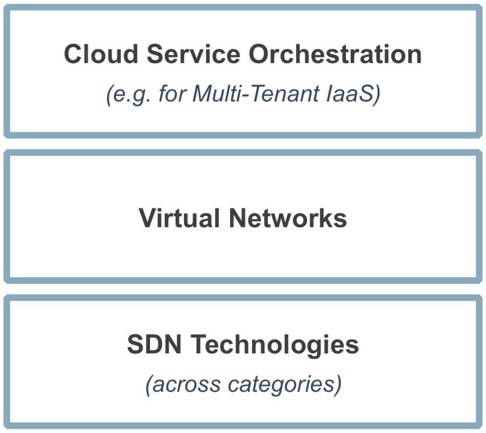


Figure 6 - SDN for Cloud

Virtual Networks are a natural match for multi-tenant cloud platforms. The approach can be used to create *per tenant (or application tier) logical networks* for cloud IaaS or hosting environments. *OpenStack* (see below), and VMware's *vRealize Automation* are examples of popular cloud orchestration stacks.

What is OpenStack?

OpenStack is an open source cloud computing platform project for public and private Infrastructure-as-a-Service. The architectural approach is based on modularity, and over several years (and releases) there is a large set of interrelated projects delivering various components of a cloud infrastructure solution*.

OpenStack includes a subsystem/project called Neutron (formerly *Quantum*) that allows for model-based integration of the network via APIs (in support of the core IaaS capabilities). As it turns out, SDN architectures that enable "Virtual Networks" are a good fit for OpenStack.

From an ecosystem perspective, the open nature of the OpenStack project (with its Apache open source licensing) has made it a popular integration target for many SDN solutions, both open and proprietary.

* www.openstack.org

As mentioned earlier, there are several SDN technology approaches to creating virtual networks (based on various techniques of segmenting a shared physical network). The most popular is the *edge overlay fabric* approach, and in fact the approach was originally created to address the "virtual networks for cloud" opportunity. In addition to the vendor solutions mentioned earlier, OpenDaylight is also developing an edge overlay fabric reference implementation.

Other technology approaches for virtual networks include "inlay" approaches including those based on OpenFlow (e.g. NEC's VTN) and even lower-level direct control of network ASICs.

It will be interesting to see which additional technologies get leveraged or developed to evolve the *virtual networks* use case. For example, will edge overlays further integrate with network underlays? If so, how will network "intelligence" be balanced between overlay and underlay.

Or will new *inlay* approaches replace (or potentially augment) the edge virtual overlay as a "better way" to create logically segmented virtual networks?

Network Functions Virtualization (NFV)

What is NFV

Network Functions Virtualization (NFV) is a call to action by many Tier 1 telcos* to accelerate the transition from proprietary hardware appliances to platforms based on standard IT virtualization technologies (e.g. virtual appliances on standard high-volume servers).

The European Telecommunications Standards Institute (ETSI) launched the call to action and requirements for NFV. Per the authors of the original ETSI white paper, NFV is distinct from, but highly complementary to, SDN. This can be reconciled in that virtual appliances hold value in and of themselves. However, to the *complementary* point, their value would increase by standardized (SDN) mechanisms to automate their deployment and management in the context of an overall solution.

More recently, carriers and vendors have launched an open source project to develop an NFV reference implementation. See Open Platform for NFV (OPNFV) at www.opnfv.org

* DT, NTT, and Verizon are among this list of NFV authors – and they are also on the ONF board. The other NFV authors are AT&T, BT, CenturyLink, China Mobile, Colt, FT, KDDI, Telecom Italia, Telefonica, and Telstra.

NFV is best considered a higher-level use case. While network-specific, it is nevertheless a statement of *requirements* by service providers.

Like the virtual networks case, several SDN technology approaches can potentially address NFV requirements.

The ability of edge overlay fabrics to flexibly redirect traffic (and accommodate VM fabric end-points) makes it a good match for NFV. And in fact almost all of the edge fabric overlay solutions in the market also list NFV as a target use case.

But here again, other technology approaches are also being pursued in the market – particularly those based on OpenFlow. An example is the recently announced Open Network Operating System (ONOS) effort from ON.Lab (onosproject.org). ONOS is focused on several SP-oriented use cases, including NFV for Central Offices (dubbed Network Functions as a Service, or NFaaS).

From a community standpoint, NFV is also a targeted use case for the Open Networking Foundation and OpenDaylight project.

Micro-Segmentation

What is Micro-Segmentation

The ability to support segmentation *at-scale* has led to a popular intermediate use case called micro-segmentation. The most popular *application* of micro-segmentation (at least currently) is the enablement of *east-west security* in the data center. Sometimes referred to as a “Zero-Trust Model”, even end-points within a traditional data center “perimeter” are subject to security (i.e. not trusted) under this model.

In micro-segmentation, end-points are assigned to segments on a very granular basis. For example, VM’s in an individual application *tier* could constitute one segment. Or at the extreme, even a single end-point (physical or virtual) could constitute its own segment.

Connectivity between these segments can then be “controlled” by the network (e.g. disallowed, allowed, allowed but via a chain of L4-7 network services) based on policy.

In conjunction with network-based segmentation and traffic steering capability, the network services themselves (for controlled connectivity and other L4-7 services) are typically implemented in a distributed fashion for scale. For example, in the east-west security application this could be distributed firewall capability in hypervisors (augmented by traffic steering for “service chaining” to dedicated security appliances when desired).

Once again edge overlay fabrics were the initial SDN technology approach to implement micro-segmentation (e.g. VMware *NSX*).

However any SDN technology approach that allows sufficient segmentation granularity, scale, and service insertion/chaining could also support micro-segmentation. Cisco’s ACI solution, for example, claims the capability.

Like virtual networks, micro-segmentation is considered an *intermediate* use case because it in turn supports a higher-level use case. For example, the east-west security capability enabled by micro-segmentation could be for the benefit of an IaaS platform.

Integrating Software Defined Networks

At the beginning of this report, *programmable interfaces* were identified as an architectural principle for SDN. And in fact for the technologies covered, the internal interfaces used (e.g. control protocols, data plane encapsulations) are well-understood elements of the various SDN approaches.

A more significant question, though, is how an SDN-based network "as a whole" can be integrated into higher-level IT systems (ultimately in support of applications).

Network management and automation is an ongoing topic, but it has come to the forefront again as a *necessity* for IT workloads to dynamically control the capabilities of software defined networks. A familiar example by now is the cloud orchestration system that requests the creation and management of virtual networks on-demand. Or the example of a container (e.g. Docker or Kubernetes) management framework that dynamically instantiates network security services as part of an application's deployment.

How SDN capabilities are exposed to the larger system (e.g. IT frameworks, applications) is an active, evolving industry topic. The following describes the two general integration approaches that are currently driving active community and industry discussion.

Model-based Network Management

This approach consists of defining *abstractions*, *models* and associated *northbound interfaces (NBIs)* to allow control/visibility of the network to "client" IT workloads. In other words, there is an abstracted *programming model* for the network that is used by higher-level systems (e.g. operators, cloud orchestrators, application frameworks).

A vendor example of this management approach would be Tail-f's (now Cisco's) Network Control System – a network-wide, multi-vendor configuration solution based on common data model abstractions (and used by operators, or programmatically via APIs). A nascent community (or at least operator-driven) example comes from the OpenConfig work group. OpenConfig[14] is currently developing network models (based on the YANG data modeling language) for configuring and managing multivendor networks.

Or as mentioned earlier, a specific (cloud) integration example can be found in OpenStack. OpenStack contains a networking service integration framework called *Neutron.* Neutron defines a network model based on abstractions of network services and elements (e.g. L2 segments, L3 subnets, L4-7 services, ports, routers). OpenStack IaaS (e.g. VMs on-demand) use API's "wrapped around" these abstractions[15] to configure network support. (As an implementation note, Neutron *plug-in's* translate interactions

[14] OpenConfig members include: Google (lead), AT&T, BT, and Microsoft.

[15] In fact Neutron even uses the term *virtual network* to describe the logical networks created. Going back to earlier points on terminology, Neutron doesn't care how a VN is implemented (edge overlay fabric, OpenFlow switches, carrier pigeons…).

with abstractions to underlying network implementations. And SDN solutions targeting OpenStack typically provide these plug-in's[16].)

Another (security application) integration example exists in OpenDaylight, where an Anti-DoS application (called *Defense4All*) leverages *network statistics* and *traffic redirection* API's from the OpenDaylight controller.

The network models (and corresponding API's) used in these approaches are based on abstractions that closely mirror physical networks (similar to the way VMs mirror physical compute). As described in the examples above, this is a useful approach. But despite its utility, a network-centric model-based approach is not viewed as sufficient to optimally address how IT workloads interact with dynamic infrastructure (which of course includes the network). Enter Group-Based Policy…

Group Policy-based Network Management

Group-Based Policy (GBP) is an approach that has gained recent industry focus. In this model, IT workload end-state requirements (a.k.a. *contracts*) are negotiated with the network. These contracts are expressed in terms of policies between "groups of end-points" (e.g. bandwidth needed between groups of VMs, access policies between groups of VMs representing application tiers, etc.). These policies are expressed in a way that is relevant to the IT workload (e.g. application, app framework, cloud orchestrator).

The policies are distributed to agents throughout the network, and the agents handle the specific actions required to implement (or *render*) the policies to elements under their control.

GBP is a *declarative* approach, because the architecture is based on distributing *end-goals* (contract policies) rather than detailed *instructions* on how to achieve those end-goals (which would be an *imperative* model).

The claim is that GBP is more scalable and modular than a purely imperative approach, and provides an interface that is a more natural match for IT workloads (describing and distributing *what* end-points require, rather than centralized command and control of network elements).

The current prominent commercial implementation of GBP for networking is Cisco's *Application Centric Infrastructure* (ACI). ACI implements a group end-point API and centralized policy store (as part of a controller called APIC). Policies are then distributed to network agents (e.g. hardware switches, software switches, gateways) via a protocol called OpFlex. The network agents then configure/operate those elements under their control based on a variety of interfaces and protocols. Note that a fully realized ACI

[16] There is a legacy mechanism for network integration independent of Neutron, but that is a corner case not covered in this report.

solution requires the use of Cisco physical network gear (specifically the Nexus 9000 series switches)[17].

Other GBP implementations are currently being developed in both OpenDaylight and OpenStack.[18] As part of these efforts, the OpFlex protocol (submitted to the IETF by Cisco for standardization) is being leveraged.

As mentioned earlier, network management and automation is a **necessity** for SDN capabilities to be consumed as part of dynamic IT infrastructure. Both administrators and software (e.g. orchestrators) need to be supported, as both will continue to play a role in operating IT systems and applications.

As *programming interfaces* for the network (either direct or indirect), these integration approaches (and likely a combination of both) are critical to delivering the value of SDN.

Final Thoughts

This report has focused on presenting a survey of SDN technologies, but ultimately what's important is how these technologies are applied to build real networks. Different networks serve different needs, but SDN approaches promise to enable networks at new levels of efficiency, flexibility, and programmability.

SDN comes at an exciting time in IT as static infrastructure platforms continue to be replaced by dynamic ones. Cloud platforms, DevOps approaches, and new application frameworks are all examples of trends that need efficient and operationally agile networks.

The timing of SDN is not an accident. Other IT virtualization technologies have driven both the *need* (for a flexible network to match flexible compute) and the *solution* (e.g. VM's as agile network elements).

Among customers, there has been early SDN adoption in: Research and Education (where "SDN" started), Financial Services (a traditional early adopter of tech), and Service Providers (for Public Cloud, and NFV use cases).

Beyond this direct interest, though, the *relevance* of SDN is truly widespread considering its role as the efficient and agile network foundation for dynamic IT.

It's an exciting time in networking.

[17] To be fair, most commercial SDN solutions require specific vendor elements that cannot be "swapped out". However a requirement for bundling SDN software **and** hardware does stand out.

[18] Where vendor politics is in "overdrive".

Appendix A – Additional SDN Use Cases

This report was organized to present a large number of SDN *Technologies* in a structured fashion. Use cases were included throughout to illuminate the technology. In many cases, though, interesting SDN use cases leverage multiple technologies and it would have been difficult to include them in the report (or would have otherwise interrupted the flow).

What follows is a collection of other interesting SDN use cases. It is by no means complete.

Multi-Layer Network Control and Consolidation

While the multi-tenant, data center use cases tend to get most of the attention, using SDN principles for multi-layer network convergence is also a popular topic. It's also different in that it's a use case that *cuts across* network domains, as well as the various SDN approaches described in this paper.

For example, at the network element level, some players (e.g. ADVA) have advocated for OpenFlow extensions targeting circuit switched optical networks.

Per other element-level approaches, one use case for *PCE* is routing and wavelength assignment for optical networks. There is also a proposal to use I2RS for a multi-layer routing service that also integrates information from an underlying (from the perspective of IP) optical transport network.

At the network-level, there are a variety of industry solutions and proposals as per Ciena, Cyan (Blue Planet) and Cariden (now Cisco) for converged IP/optical networks,

Approaches that create and integrate new physical layer capabilities are also emerging. E.g. Plexxi - a controller for a flexible, high-bandwidth optical mesh network in the data center. In this solution physical network capabilities (e.g. bandwidth, latency, isolation) can be assigned dynamically based on application requirements (e.g. a *Big Data* workload deployment would configure for bandwidth and latency between associated compute and storage elements).

Approaches that detect elephant flows, and then "offload" them to the optical network (to save performance and resources in the packet-based network) are also popular use cases.

Other use cases:

- A single network element abstraction of a geographically distributed network (R&E use case). As a real example, a many-to-one "virtual switch" was demonstrated at the SuperComputing 2012 Conference. While the abstraction of a single switch was presented, the underlying implementation consisted of multiple NE's across geographies.

- Extensions to policy-driven IT in the Enterprise beyond network access control (e.g. abstracting the network over multiple campus/branch locations, role-based dynamic security mechanisms and QoS, etc.).

- For providers, current 3GPP standards specify a Policy and Charging Rules Function (PCRF) that offers, for example, per subscriber QoS control in real time. While 3GPP applies to mobile, there is discussion around generalizing the approach to other forms of networking.

- On-Demand WAN capabilities for SPs and/or Enterprises (e.g. for cloud backbone connectivity, bulk data transfer, backups) – including potential coordination with transport networks.

- Intelligent network on-ramping and handoff services for SPs (e.g. WiFi offload).

[end of document]