WHITE PAPER

SmartSwitches

Silvano Gai & Mario Baldi

Abstract

SmartSwitches are fundamental building blocks of a Distributed Services Platform in a top-of-rack form factor. This new category of product can be easily integrated into any new or existing enterprise and cloud environment, regardless of whether it is public, private or hybrid.

Summary

Enterprise networks and cloud computing can benefit from distributing services closest to where data is created for scalability and performance reasons. Two approaches have been proposed:

- **SmartNIC**, i.e. a PCIe card that is installed in any server to implement distributed services
- **SmartSwitch**, a top-of-rack switch that, in addition to implementing all classical network protocols, also supports distributed services for all the hosts in the rack.

SmartSwitches can be easily integrated into any new or existing enterprise and cloud environment regardless of whether it is public, private or hybrid. SmartSwitches do not require changes to the servers' hardware or software, they do not make any assumptions on the server operating systems or require any driver or agent to be installed on the servers, while providing distributed services at scale and wire-rate. They are ideal for any brownfield deployment.

A History of Network Switches

1990s: Layer 2 Switches

Network switches (*switches* for short) are the evolution of network bridges whose behavior was defined by the Institute of Electrical and Electronics Engineers (IEEE) in the standard IEEE 802.1 to connect two or more Ethernet segments. Switch and switching are terms that do not exist in standards; they were introduced to indicate a multiport bridge.

Initially, switches were pure layer 2 devices that forwarded Ethernet frames without knowing their content and without modifying the frames, thus providing connectivity for the many layer 3 protocols deployed in those days. The forwarding model of layer 2 switches is straightforward and based on the *exact* lookup of the destination MAC address in a forwarding (or filtering) table; this is usually accomplished with a hashing table, which is easy to implement in hardware. Layer 2 switches do not require any configuration; the forwarding table is initially empty and is populated by associating the source MAC address of a frame being received with the port through which it is received - a technique called *backward learning*. When the lookup of the destination MAC address fails (i.e., the association of the MAC address with a port has not yet been done), the frame is forwarded on all ports other than the one it was received from - a technique named selective broadcast.

This forwarding technique works only on a tree topology because backward learning does not operate properly when frames travel along a loop and selective broadcast creates copies of frames traveling along a loop until the capacity of links and/or switches is saturated. On a mesh topology, the Spanning Tree protocol is deployed to block a set of ports that are not used to ensure that forwarding takes place on a tree. Blocking ports is inefficient from a bandwidth perspective because the corresponding links, although potentially capable of carrying traffic, are not being used. In the nineties, there was a lot of innovation around the spanning tree protocol trying to increase the overall network bandwidth, but all proposed solutions introduce significant complexity in the network design and configuration to ensure high bandwidth utilization.

2000s: The Introduction of Layer 3 Switches

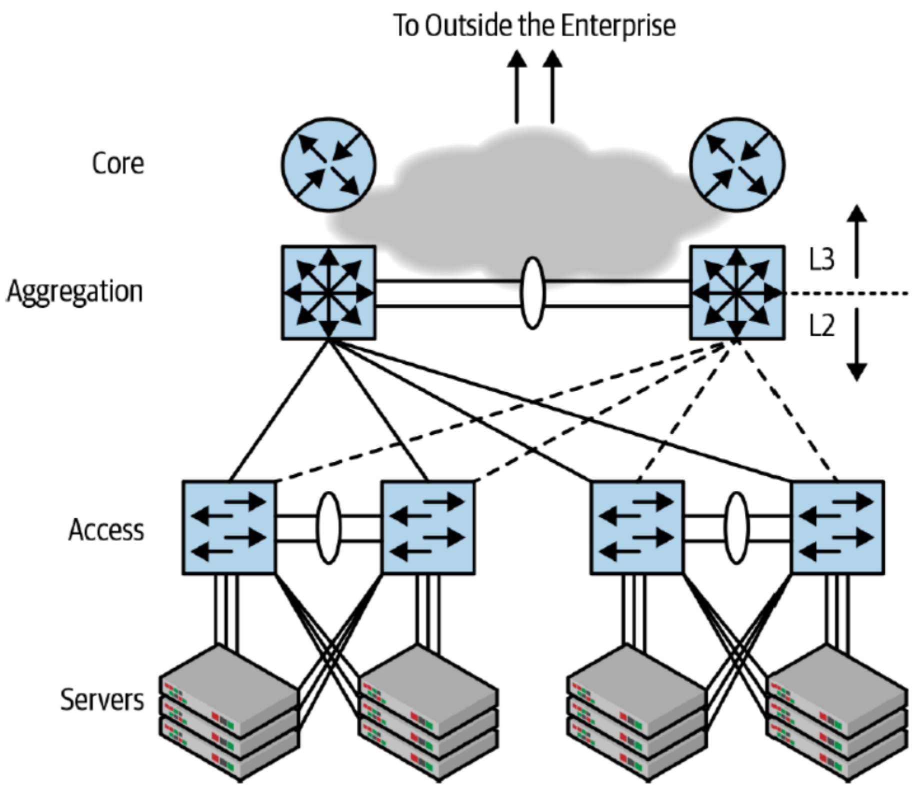
In the 2000s, it became apparent that the only surviving layer 3 protocol was IPv4 (Internet Protocol version 4). Therefore, it made sense to bypass the limits of the Spanning Tree Protocol by implementing IPv4 routing into the switches. Layer 3 switches were born. IP forwarding does not require blocking any port in the network, hence it can take advantage of all the links in a network.

Traditionally a Layer 3 switch forwards all traffic to a destination on the same path. However, a feature called ECMP (Equal Cost Multi Path) even enables to load balance traffic to each destination among multiple paths that are considered to have the same cost according to the routing protocol in use, like for example BGP (Border Gateway Protocol).

Appreciating the difference between layer 2 (L2) forwarding and layer 3 (L3) forwarding is essential to understanding why Layer 3 switches are substantially more complex than Layer 2 switches. At the IP level, forwarding is done packet-by-packet by looking up the destination IP address into a forwarding table with a technique called LPM (Longest Prefix Match), which is much more complex to implement in hardware than the exact, possibly hash-based, matching required in L2 switching. Deploying LPM enables the routing table not to include all the possible IP addresses, but just prefixes, thus allowing for larger networks without proportionally larger forwarding tables. A prefix is expressed as a combination of an IP address and a netmask to identify which bits in the address represent the prefix. Being 32 bit long, a netmask is cumbersome to write and not intuitive to understand; hence often the length of the prefix - or number of most significant bits in the address that represent the prefix - is explicitly indicated (e.g., /16, /24).

An entry of the forwarding table can be used to determine the forwarding information (e.g., the next hop and/or output interface) for any packet with a destination address that has that prefix in its most significant bits. If multiple entries match, the switch selects the one with the longest prefix: since it refers to a smaller set of addresses, it is somewhat more specific to the destination at hand.

This kind of forwarding is stateless since each packet is forwarded independently of previous packets. With the growing size of ICs (Integrated Circuits), more features were added to layer 3 switches, such as ACLs (Access Control Lists) to classify and filter packets, still in a stateless fashion, with limited scalability.



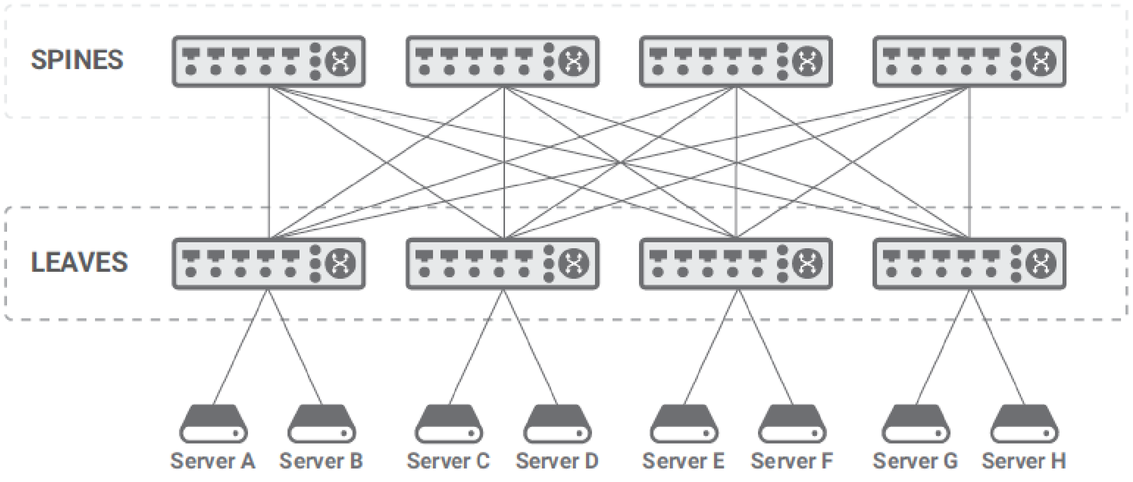
In terms of network design, the preferred topology was a Core-Aggregation-Access Design (picture courtesy of Dinesh Dutt): a mixture of layer 2 at the periphery and layer 3 in the network's core, to allow layer 3 subnets to span across multiple edge switches. This allows, for example, Virtual Machine mobility among the servers connected to the same access switches. This design provided more aggregate bandwidth than a pure layer 2 approach, but still had a bottleneck in the two core and aggregation switches: traffic to all destinations not attached to the same pair of access switches must travel through the access switch identified as the root of the spanning tree.

2010: Modern L3 Switches

In the next decade, with the advent of even denser integrated circuits, layer 3 switches gained other functionality that can be grouped in three categories.

Clos Network and ECMP Support

Clos networks remove the bottleneck present in the core of the Core-Aggregation-Access Design. A Clos network is a multistage network that Charles Clos first formalized in 1952. It is a two-stage network in its simplest embodiment, like the one shown in the figure, but it can scale to an arbitrary number of stages.



In their current usage and terminology, two-stage Clos networks have leaves (equivalent to access switches) and spines (equivalent to aggregation switches). The spines interconnect the leaves, and the leaves connect the network users. The network users are mostly, but not only the servers and can include network appliances, wide-area routers, gateways, and so on. No network users are attached to the spines.

The Clos topology is widely accepted in data center and cloud networks because it scales horizontally: adding more leaves allows to connect as many network users as needed, while adding more spines enables supporting larger amounts of East-West traffic. In fact, traffic to destinations not attached to the same leaf can transit through any of the spines (and corresponding link connecting the selected spine to the source and destination leaf). In order to enable this, both leaves and spines must be layer 3 switches (as layer 2 switches would block all leaf-spine links but one) and ensure that all spines are being used when forwarding packets. This is achieved through a technique called equal-cost multipath (ECMP) that stores multiple next hops (e.g., spines) for the same destination and load-balances forwarded packets to all of them.

Network Virtualization Support

Network virtualization enables the definition of multiple independent and separate virtual networks on a single physical network. In other words, although there is a single set of switches and links (physical network), multiple networks can be defined: the configuration (e.g., addressing scheme) of each network is independent from the others and the traffic is completely separated from the others.

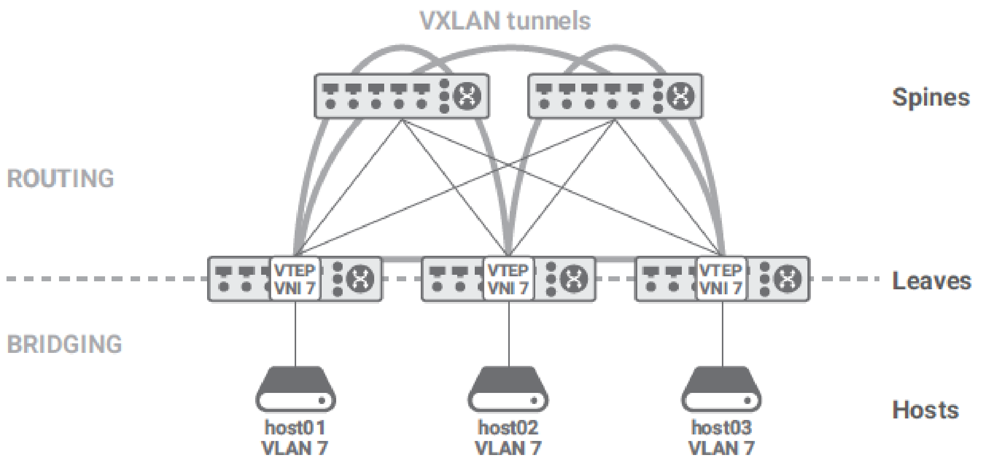
Overlay networks are a common way to satisfy this requirement. An overlay network is a virtual network built on an underlay network, i.e., a physical infrastructure.

The underlay network's primary responsibility is forwarding the overlay encapsulated packets (for example, using VXLAN encapsulation) across the underlay network efficiently using ECMP when available. The underlay provides a service to the overlay. In modern network designs, the underlay network is always an IP network (either IPv4 or IPv6) because we are interested in running over a Clos fabric that requires IP routing.

In overlays, it is possible to decouple the IP addresses used by the applications (overlay) from the IP addresses used by the infrastructure (underlay). The VMs that run a user application may use a few addresses from a customer IP subnet (overlay), whereas the servers that host the VMs use IP addresses that belong to the cloud provider infrastructure (underlay). VXLAN is probably the best-known encapsulation protocol for overlay networks.The points where packets are encapsulated/decapsulated are called VTEPs (VXLAN tunnel endpoints).

As we mentioned in the introduction, Core-Aggregation-Edge networks allowed layer 2 domains (that is, layer 2 broadcast domains) to span multiple switches. The VID (VLAN identifier) was the standard way to segregate the traffic from different users and applications. Each VLAN carried one or more IP subnets that were spanning multiple switches.

Clos network changed this by mandating routing between the leaves and the spines and limiting the layer 2 domains to the southern part of the leaves toward the hosts. The same is true for the IP subnets where the hosts are connected. VXLAN solves the problem of using Clos networks while at the same time propagating a layer 2 domain over them. In a nutshell, VXLAN carries a VLAN across a routed network. In essence, VXLAN enables seamless connectivity of servers by allowing them to continue to share the same layer 2 broadcast domain.



SDN Support

At the end of the previous century, all the crucial problems in networking were solved. On my bookshelf, I still have a copy of Interconnections: Bridges and Routers by Radia Perlman dated 1993. In her book, Perlman presents the spanning tree protocol for bridged networks, distance vector routing, and link-state routing: all the essential tools that we use today to build a network. During the following two decades, there have been improvements, but they were minor, incremental. The most significant change has been the dramatic increase in link speed.

In 2008, Professor Nick McKeown and others published their milestone paper on OpenFlow. Everybody got excited and thought that with software-defined networking (SDN), networks would change forever. They did, but it was a transient change; many people tried OpenFlow, but

few adopted it. Five years later, most of the bridges and routers were still working as described by Perlman in her book.

But two revolutions had started:

- A distributed control plane with open API
- A programmable data plane

The second one was not successful because it was too simple and didn't map well to hardware. However, it created the condition for the development of P4 architecture to address those issues. OpenFlow impacted even more host networking, where the SDN concept is used in a plethora of solutions.

SDN enables virtualized networks with a less complex overlay not based on running control protocols like BGP, but programmed through an SDN controller.

Scalability Considerations

These switches have impressive performance in terms of bandwidth and PPS (Packet Per Second) but they are substantially stateless devices that are not idoneous to implement stateful services like firewall, NAT, load balancing, etc. They also have scalability issues in the number of routes and ACLs that they can support.

Scalability is critical because each virtual network has its addressing space that cannot be aggregated with one of the others and its own separate tables for services like ACL, NAT, etc.. Hence, even though the virtual topology of virtualized networks may be simpler than the physical topology, and hence routing tables smaller than the ones of the physical network, still the amount of memory required for routing tables and data structures for other services increases the size of routing tables. Moreover, VTEP mapping requires large tables that are not present in traditional switches that do not support network virtualization.

2020: SmartSwitches (Distributed Service Switches)

SmartSwitches, a.k.a Distributed Service Switches, add to the state of the art layer 3 switches a rich collection of wire-rate stateful services, e.g., firewall, LB (Load Balancer), NAT (Network Address Translation, TAP (Test Access Point) Network (traffic observability), streaming telemetry, etc. A SmartSwitch can apply these services both to the underlay network and to the overlay network. In the second case, they are typically colocated with the VTEPs.

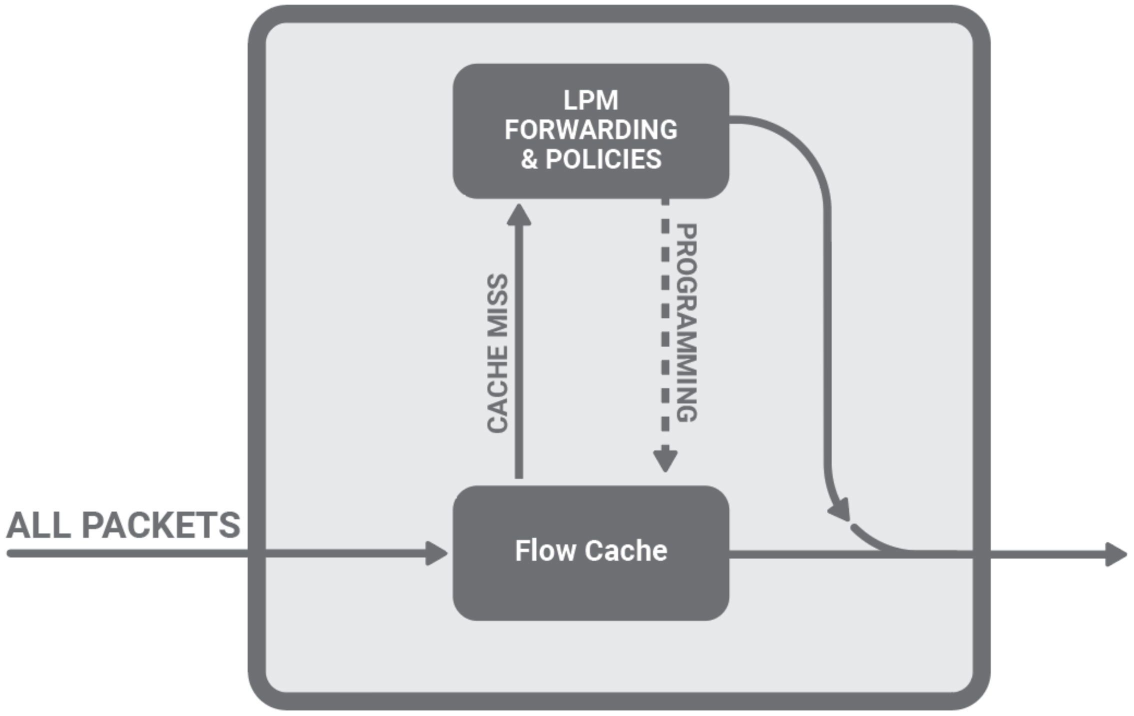
To effectively support these services, SmartSwitches have a different forwarding mechanism that is stateful: decision of what to do with the packet may depend on previous packets of the same communication. Hence, each packet between host A and host B is processed as a part of a session between A and B that comprises two unidirectional flows: one from A to B and one from B to A. The term session applies as well to connection oriented protocols like TCP or connectionless protocols like UDP.

The concept of session is important because in many cases reaching the forwarding decision (whether to forward the packet and if so how) requires complex processing of one or more packets and then remains the same for subsequent packets of the same communication,

The stateful forwarding mechanism is also called *cache-based forwarding*. It relies on a flow cache, a binary data structure capable of "exact" matching packets belonging to a particular flow. The word "exact" implies a binary match easier to implement both in hardware or software, unlike a ternary match, such as LPM. The flow cache contains an entry for each flow, i.e., two entries per session. The flow can be defined with an arbitrary number of fields, thus supporting IPv4 and IPv6 addresses, different encapsulations, policy routing, and firewalling.

Cache entry contains information needed to forward the packet (e.g., layer 2 and/or layer 3 address of the next hop, type of encapsulation, etc.).

The separate initial processing that leads to the forwarding decision may create new cache entries. When a packet is received by the switch, if it does not match any entry in the flow cache (a "miss") it is processed according to the initial processing. Otherwise (“hit”), it is forwarded according to the information in the matching flow cache entry.



The above figure shows the organization of this solution. Any packet that causes a miss in the flow cache is redirected to a software process that applies a complete decision process and forwards or drops the packet accordingly. This software process also adds two entries in the flow cache (dashed line) for the session, so that subsequent packets can be forwarded or dropped as the previous packets of the same flow.

Because the processing of initial packets (misses) is so clearly separated from the processing of the following packets (hits), it can be implemented independently and possibly on a separated execution engine. For example, hits could be processed by a specialized integrated circuit or an application specific processor (hardware data path), while misses could be processed by software running on a general purpose CPU (software data path).

Let's suppose that an average flow is composed of 500 packets: one will hit the software process, and the remaining 499 will be processed directly in hardware with a speed-up factor of 500 compared to a pure software solution. In other words, cache-based forwarding allows one to take advantage of the flexibility of software with the performance of hardware. Of course, this solution should guarantee that packets processed in software and hardware do not get reordered.

Since in cache-based forwarding the handling of the first packet of a flow or session can be implemented completely independently from the handling of subsequent packets, when evaluating the performance of a SmartSwitch we look at two different indexes:

- Connections Per Second (CPS) offers an indication of the performance in processing the initial packet of each flow.
- Packet Per Second (PPS) provides a measure of the performance in processing other packets.

In specific use cases, complex and time consuming processing is allowed on the first packet to decide whether and how a flow should be forwarded without affecting the throughput in terms of PPS.

Scalability Considerations

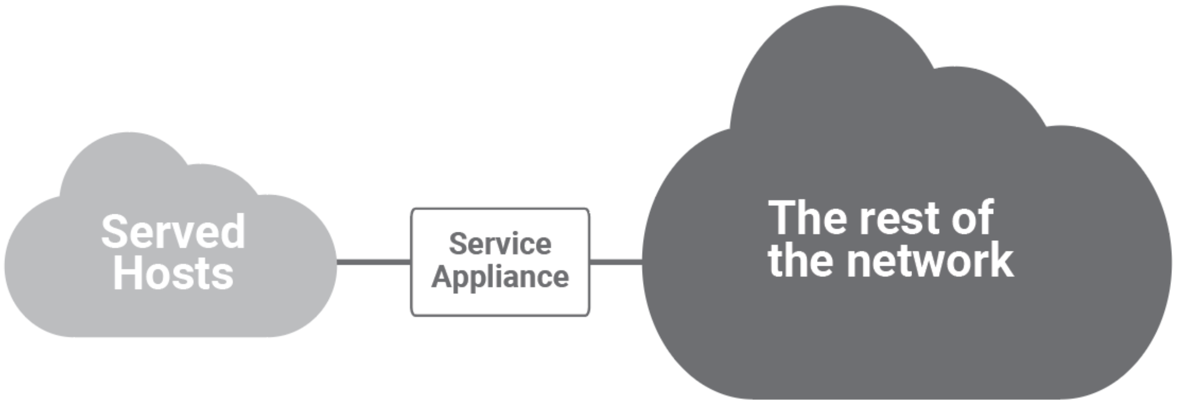
This architecture requires that the SmartSwitches have a large volume of high-speed memory to store all the flow descriptors, the ACLs, and the routes. Cloud providers have a large number of tenants each with their virtual network. Even if routing and access control lists (ACLs) requirements for each tenant are moderate, when multiplied by the number of tenants, there is an explosion in the number of routes and ACLs, which can be difficult to accommodate on the host with traditional table lookups. It is not unreasonable to support millions of routes and ACLs and tens of millions of flows. If the flow descriptor is, let's say, 256 bytes, and we have ten million, the flow cache consumes 2.5 GB of memory. When we add the size of the route table, the ACL table, intermediate data structure and code, it is not unreasonable to use 8 to 16 GB of memory. This size of memory cannot be integrated on-chip, and therefore external memory needs to be used. Multiple DDR busses are required to obtain high performance, with DDR5 at the highest possible frequency being desirable.

It helps that, with the exception of the flow table, the rest of information/tables is needed only for processing misses, which is a small fraction of the overall traffic.

Services Implementation

Since services are a vital feature of SmartSwitches, before going more into detail on the SmartSwitch architecture, let's consider how historically services have been implemented in networks.

Network and security services have traditionally been deployed in front of a group of end hosts that share some common function and are topologically close: e.g., private corporate clients, public corporate servers, the servers of a specific department, the hosts of a corporate branch. In this context it is natural to implement services in an appliance that is located in the appropriate topological position, as shown in the following figure that depicts the most common deployment model for services such as NAT, load balancers and firewalls.



Two characteristics of modern data centers, combined, conflict with this model:

- Traffic patterns across Clos topology (E-W vs N-S)
- Network virtualization

With virtualization, while the same logical model still applies, the physical placement of the hosts, their physical proximity, and the natural point to deploy the service application (on the shortest path of traffic to and from the served hosts) have disappeared. Of course! It is in the very nature of virtualization (as well as its value and goal) to make all functional components (resources, topology, location) independent of the physical ones. Hosts communicating with each other can be anywhere in a data center and consequently the network design must:

- Provide equal and abundant bandwidth between any pair of hosts.
- Separate "physical" adjacency (in the IP terminology sense, namely layer two connectivity) from the actual physical topology.

The Clos network topology with its fractal capability of scaling (possibly incrementally) by adding identical components (leaf and spine switches) interconnected according to the same pattern, effectively solves the first requirement.

Network virtualization techniques, such as SDN (Software Defined Networking) and VXLAN (Virtual Extensible LAN), address the second requirement.

In this context, the remaining challenge is how to deploy the networking and security services in front of the (virtual) hosts they serve. In the rest of this section we will discuss the three architectures used before the advent of SmartSwitches, namely:

- Appliances
- Software on the host
- SmartNICs

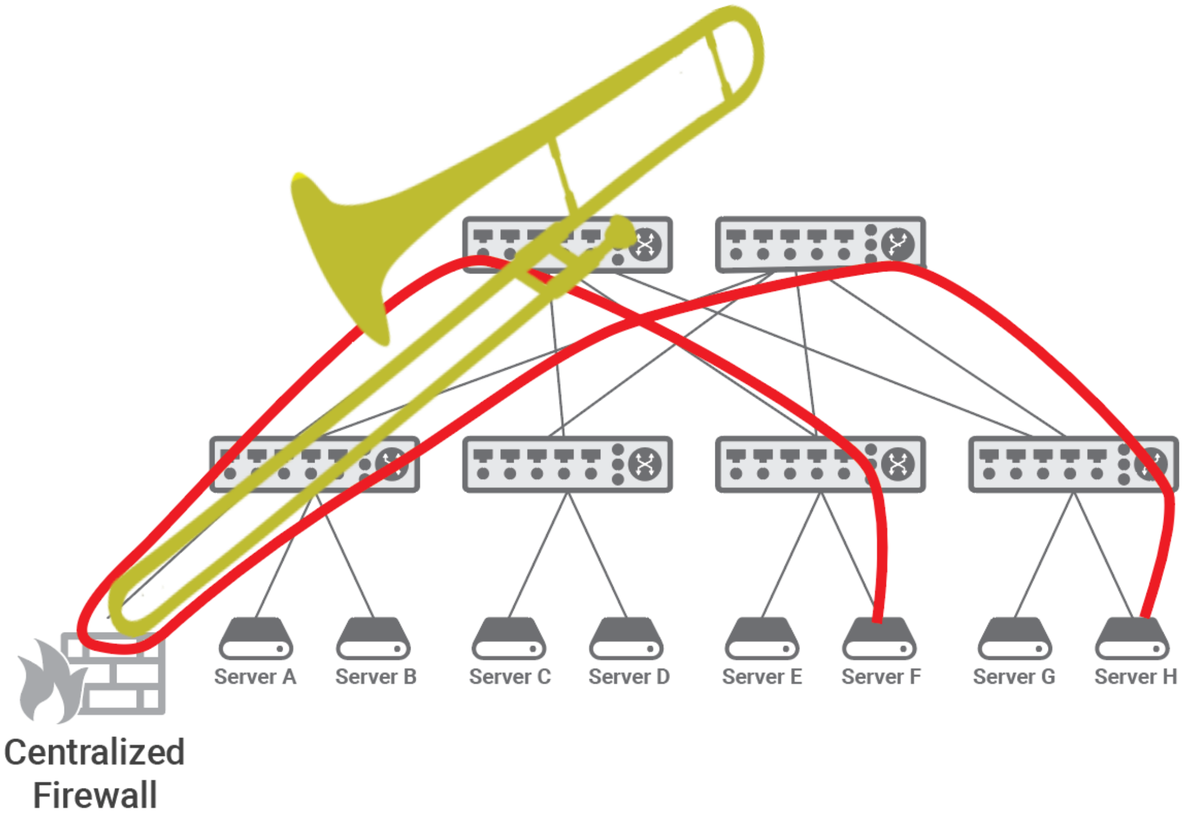
Appliances

Appliances are physical boxes installed in datacenters to provide a specific function, e.g., Firewall, load-balancing, VP termination. They are usually built using an x86 complex and some network processor or dedicated ASICs to improve performance. They have network interfaces that can be configured in multiple VLANs, VXLANs, IP subnets, etc. To perform their tasks, they need to sit on the path of the network traffic.

It is possible to ensure this by using separate physical or virtual networks and layer 2 or layer 3 techniques. For example, they can be configured as the IP default gateway for a group of hosts in a subnet forcing all the traffic destined to other subnets to go through them (layer 2 technique). They can also inject routes into the network using protocols like BGP to redirect the traffic for such subnets through them (layer 3 technique): a typical case is for a firewall to inject the default route for the Internet, forcing all external traffic to go through it.

Traffic Tromboning

Whichever of the above techniques is deployed, traffic between topologically close hosts might take a long detour and get back, as shown in the figure below.

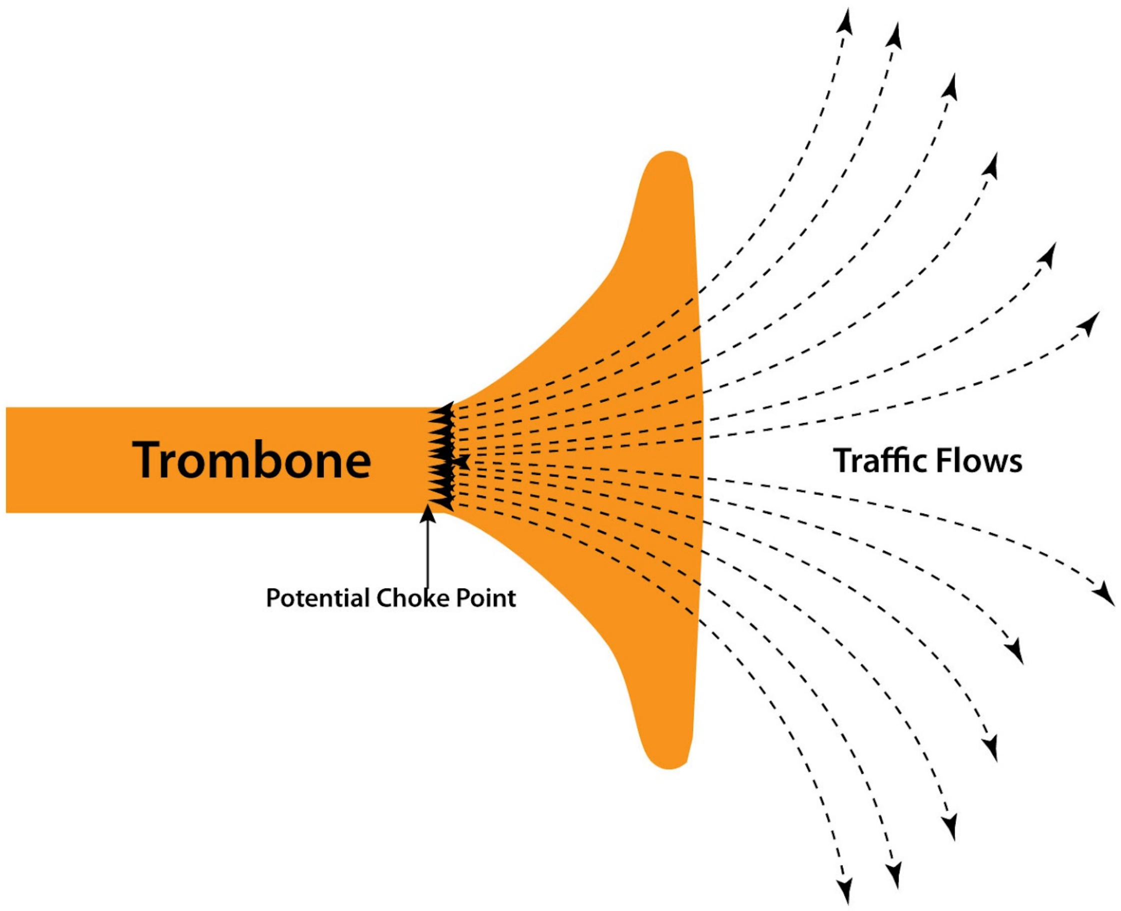


This phenomenon is commonly known as *traffic trombone* or *traffic tromboning*, as the shape of the path taken by the traffic in the network resembles the shape of the brass musical instrument that dates back to the 15th century and whose name, *trombone*, derives from *tromba*, the Italian word for trumpet: a large trumpet.

In a data center fabric based on a Clos topology, traffic tromboning results in traffic traversing the leaf-spine fabric twice, which leads to duplication of both network load and the latency experienced by the two hosts.

Scalability and Fault Tolerance

When services are applied to traffic between most pairs of hosts, the appliance becomes a funneling point—also recalling the shape of a trombone—that must be traversed by traffic from a disparate set of source-destination pairs (see the following figure for a graphic representation). We will therefore subsequently refer to service appliances such as firewalls, IDSs (Intrusion Detection Systems), NAT (Network Address Translation), load balancers, as trombones.



Appliances, by their very nature, create a network traffic scalability problem, as well as a fault tolerance problem: if the single appliance fails, its service(s) and the traffic it handles stop.

Addressing this through load balancing among multiple appliances is challenging since their services are often stateful and *synchronizing state across appliances is at best impractical and at worst impossible*.

Let's consider a ubiquitous combination of a firewall and a NAT that we all have in the router that connects our house to the Internet. When a family member browses the Internet, a TCP session is created. The router needs to create a NAT rule for the outgoing traffic, but also a second one for the return traffic. These two rules need to be kept for the duration of the TCP session.

If, for example, in a corporate network context, two or more NAT appliances are required for scale or fault tolerance, one of the following should apply:

- Direct and reverse directions of a flow should go through the same appliance.
- State should be synchronized across the various appliances.

For example, if a NAT function is present, the packet on the Internet will have the NAT source IP address. If multiple appliances are deployed in parallel, the reverse traffic will be explicitly addressed and routed to the NAT instance with the IP address used for the translation. Hence, when a flow is assigned to a traffic trombone, it uses that particular trombone for all its duration or until the trombone fails.

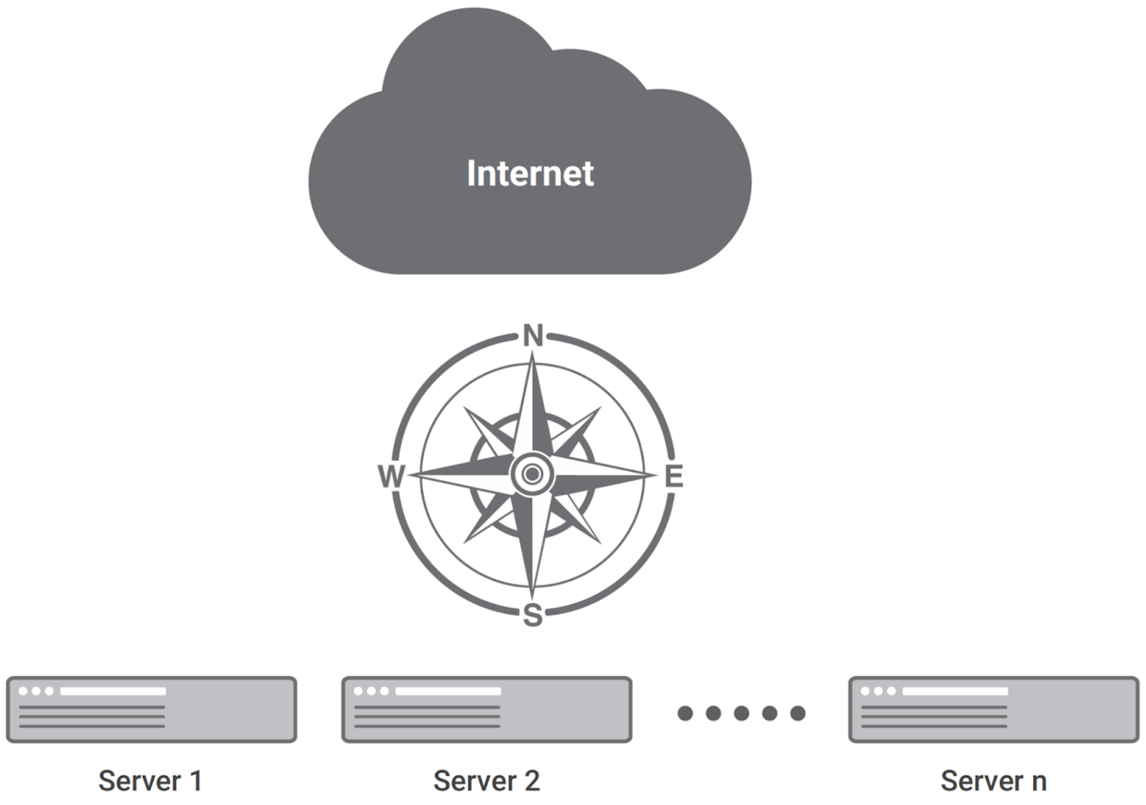
In the more general case where multiple instances of an appliance are used and the destination IP address is the same for traffic among them (e.g., a stateful firewall), the load balancing solution in front of the appliance must ensure that forward and reverse flows go through the same instance of the appliance.

Mechanisms exist for high availability across multiple trombones in case of catastrophic failures.

North-South vs East-West Traffic

Not every trombone is a bottleneck. Historically tromboning was happening towards the wide area network, a typical example being the firewall connecting a corporate network to the Internet. However, usually in that scenario the capacity of the link to the service provider is the bottleneck.

Today some high end firewalls, albeit costly, are capable of supporting up to 100 Gbps. Firewalls operating at 1 Gbps are very common and cheap, with 10 Gbps ones being considered the reasonably priced midrange. Therefore firewalls are typically not choke-points when used to connect to the Internet, a type of traffic that is also called North-South in a data center.



Different considerations apply to East-West traffic, i.e., the traffic inside the data center. Several studies have shown that the amount of East-West traffic is at least ten times larger than the North-South one.

When applied to East-West traffic, the firewall trombone may become a bottleneck and additionally destroy the optimal routine provided by the Clos network, thus increasing latency and causing a large percentage of the network bandwidth to go wasted.

There are indeed enterprise deployments where this might not be a problem because the traffic is limited compared to the large capacity of the data center fabric and the specific applications utilized can tolerate latency. But, there are a significant number of cases where traffic tromboning drawbacks are critical, with obvious examples being cloud provider data centers (where the fabric bandwidth is shared by the various tenants) and market segments that deploy time sensitive applications, such as financial trading.

This analysis of traffic tromboning leads to a simple conclusion: "Whenever possible, centralized appliances should be avoided for East-West traffic since they are bottlenecks, disrupt optimal routing, and introduce significant latency and jitter."

Instead, the same scale out model utilized for compute resources should be leveraged by naturally placing services at the edge so that they are executed in a *distributed* fashion by Distributed Services Nodes (DSNs). This can be achieved in three different ways:

- Through software running on the host
- With the use of SmartNICs on the host
- With SmartSwitches

A terminology consideration: there is not yet an industry-wide agreement on naming the Distributed Services Nodes (DSNs). Other proposals are DPUs (Data Processing Units), IPUs (Infrastructure Processing Units), FOCPs (Function Offload Coprocessors), FACs (Function Accelerator Cards), and DSCs (Distributed Service Cards). We will consider these terms' synonyms in the rest of the document.

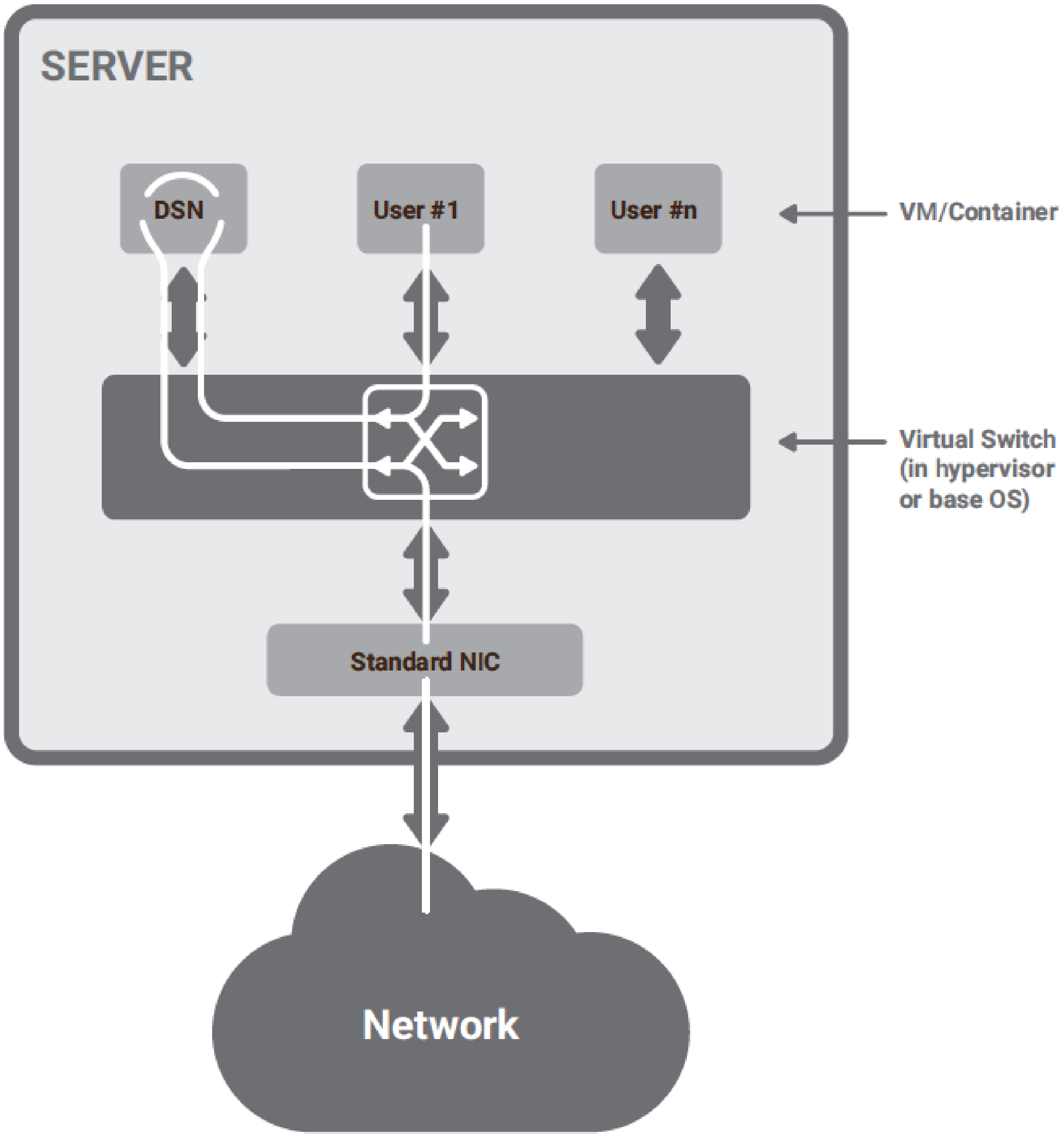
Software Running on Servers

The idea of running network, security, and other services in software on the server is not new. It has the intrinsic advantage of providing service scalability since anytime a new server is added to the data center, the associated services are also added. It also has two significant disadvantages:

1. Services consume precious CPU cycles on the main server CPU, thus reducing the amount of computing power available for user applications. This creates a significant performance issue.

2. If the server is compromised in an attack, the associated services are compromised, thus not protecting the data center from "lateral" movements of the attacker.

In virtualized and containerized environments, running services on the server can be achieved by executing a DSN in a Virtual Machine (VM) or container running on the very host where workloads receiving the services are executed, as depicted by the following picture.



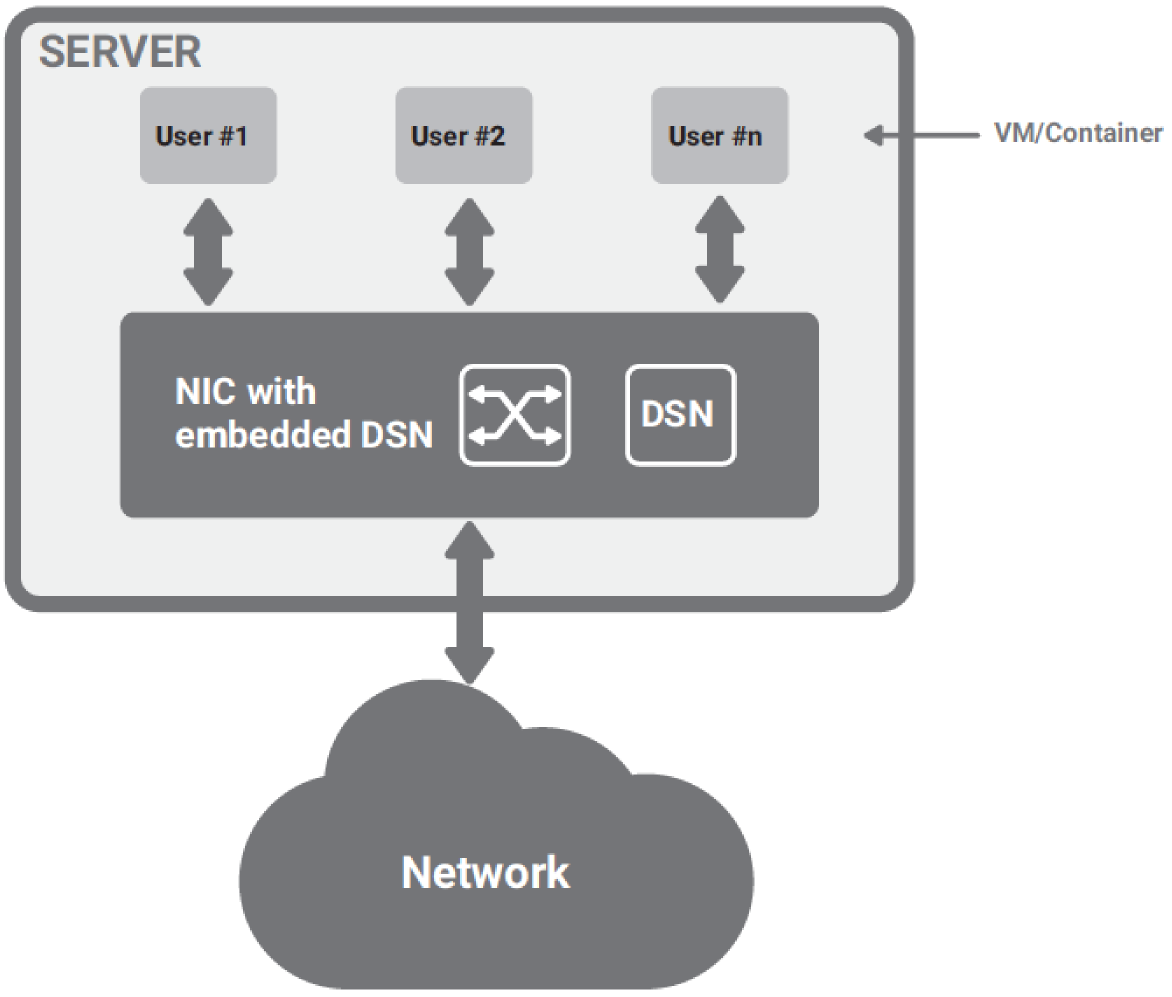
This approach still results in a small tromboning effect within hosts (as traffic flows between workloads and service VM) or between neighboring hosts if service VMs are not placed in every single host.

Such local tromboning can be avoided by including services in the hypervisor software stack, for example within the virtual switch.

Both above solutions cannot be applied to bare metal (not virtualized) environments since there is no footprint to run the services. Bare metal servers are all different, and the assumption is that their application software, operating system, drivers, and the kernel cannot be modified since they are tied to specialized/legacy applications.

SmartNIC

To solve the performance and security issues associated with software running on the host, SmartNICs were introduced. These are NICs (Network Interface Cards) with embedded specialized hardware (DSN) to implement and run services. They sit at the ideal point to implement these services, i.e., the attachment point where the server communicates with the outside world. They are naturally on the traffic path of network and storage, see next figure.



Adopting domain-specific hardware for services at the edge and installing it in conjunction with servers allows services to:

- run at wire speed without taking away any resources from the workloads executing on the hosts
- scale at the same rate as compute.

The genesis of SmartNICs is in the public cloud, where distributed hardware-accelerated services are a proven model. In the public cloud, there is also an intrinsic need for multi-tenancy. Each tenant must be protected from other tenants. The infrastructure itself should be protected against tenants.

One problem that SmartNICs only partially solve is the "bare metal" server. SmartNICs require drivers, and sometimes the bare metal server runs a recent OS for which the appropriate driver is available, but most of the time, it runs a proprietary OS, or an old version of an OS for which drivers are not available. The server operator also has intrinsic resistance to physically opening each server to replace its NIC with a SmartNIC.

Greenfield vs. Brownfield Deployment

A standard definition of the term *greenfield project* is "a project that lacks any constraints imposed by prior work."

In a greenfield installation of cloud infrastructures, each component is chosen carefully. For example, installing the more desirable server configuration is possible, including a SmartNIC capable of supporting a DSN. In general, a greenfield installation offers the highest level of flexibility on where to locate the DSNs.

When servers are deployed in production, they become *brownfield*, and their hardware configuration is usually kept unchanged until the server is decommissioned. Therefore, if a server was installed with a NIC incapable of supporting a DSN, the NIC is not upgraded.

In brownfield projects, retrofitting a distributed services platform is done by adding or replacing network boxes with new ones capable of supporting DSNs.

Sizing distributed services

Another important consideration is how to size distributed services in terms of performance and table sizes. Some servers, being physical servers, virtual machines, or containers, have minimal requirements in terms of external communications, while others have extreme ones. Sizing all the SmartNICs for maximum server performance may be too expensive, while choosing a more effective solution may be a bottleneck for servers requiring significant I/O or storage bandwidth.

The solution is statistical multiplexing, i.e., having a pool of high-performance SmartNICs shared by a group of servers. The larger the collection of SmartNICs and the group of servers, the better statistical multiplexing works. The question is where to host this pool of SmartNICs.

The answer is the seed idea of SmartSwitch, or Distributed Service Switch, i.e., a layer 3 access switch that hosts one or more DSNs shared by all of the hosts attached to it. A SmartSwitch can be implemented as a pool of SmartNICs in a tightly integrated fashion so that the SmartNICs disappear and the SmartSwitch presents to the users a stateful service capability.

Enterprise Data Centers vs. Public Cloud

Another important consideration is that Enterprise data centers, while potentially extremely large, don't have the scale of public clouds.

A distributed services platform located in a switch is appealing to them for the following reasons:

- It increases East-West security by deploying more firewalling and opportunistic encryption everywhere (if encryption is a free service, let's use it).
- It simplifies the network infrastructure by eliminating traffic trombones and the associated overlay encapsulation.
- It brings them closer to a cloud-like infrastructure that plays well with public clouds and can lend itself to becoming a hybrid cloud.
- The amount of traffic an enterprise solution should support, although high, is not comparable to that of public cloud providers, and therefore, statistical multiplexing works well.

Enterprises are interested in a turnkey solution that includes a policy manager and telemetry tools required for troubleshooting and performance analysis.

SmartSwitches eliminate the need for costly discrete appliances such as the firewall and load balancer to achieve a lower total cost of ownership. This requirement is fundamental in large data centers. Still, it is extreme in small satellite data centers, where the cost of discrete appliances is shared across a limited number of servers and, therefore, becomes the dominating factor.

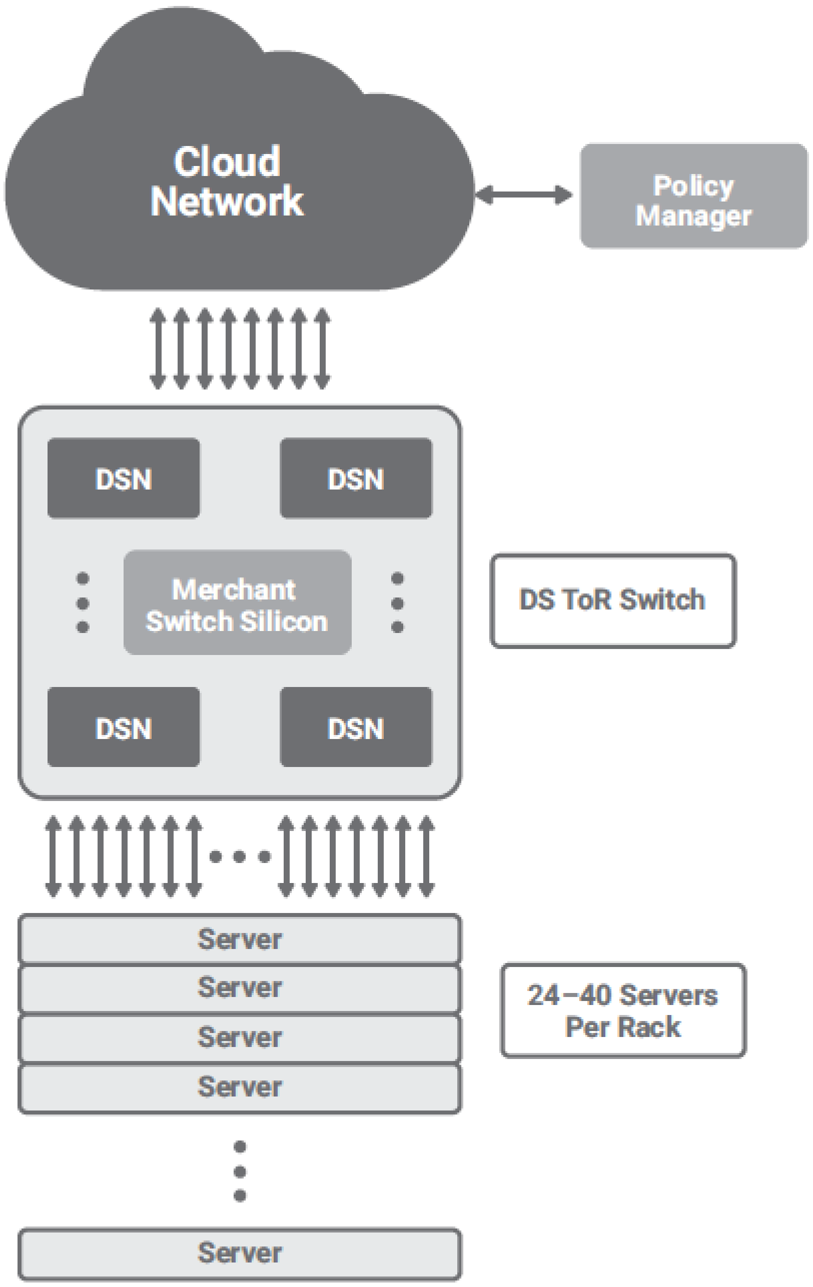
SmartSwitches: The New Paradigm

The SmartSwitch is a Layer 3 switch, augmented and modified to offer the stateful services usually provided by a SmartNIC or by software running on the server. It is augmented because it needs to have all the hardware circuitry to provide stateful services like firewall, load balancer, telemetry, etc. It is modified since its forwarding algorithm needs to be stateful to provide such services.

To ease the insertion, SmartSwitches need to provide the ability to deploy service insertion with all the existing data center fabric designs, whether it is a layer 3 Clos network with leaves and spines or traditional access, aggregation, and core design.

SmartSwitches also need to accommodate the standard best practices in the data center, such as the widely adopted server dual-homing design (which implies support for active/active service implementation), virtual machines live migration, etc.

One possible hardware organization for a SmartSwitch is shown in the following figure.



This solution shows a ToR (Top of Rack) switch (also known as a *leaf switch*) containing multiple DSN ASICs that implement stateful services shared among servers in a rack. This solution reduces the total cost of ownership, especially for enterprise applications, where it may be difficult to justify a dedicated DSN for each server because the traffic volume is not as high as in the public cloud.

The figure shows a possible arrangement with a SmartSwitch built with a merchant silicon switch ASIC and four DSN ASICs. The number of DSN ASICs may change according to the stateful service performance required by the servers. It is also conceivable that as ASIC technology progresses, the number of ASIC will reduce, and in some cases, a single chip will integrate the SmartSwitch functionality.

Please note that from a management and telemetry perspective, managing policies on a SmartSwitch is similar to managing policies on a SmartNIC; there is still a policy manager that can work in any of these solutions and should support a mix of different implementations in the same network.

To implement security across VMs and containers in the same server, the NIC in the server should send all the traffic to the SmartSwitch; this can be achieved by deploying SR-IOV or by properly configuring VLANs in the software switch in the hypervisor or the hosting operating system. Each VM or container can be tagged explicitly; for example, by adding a VLAN tag, or it can be tagged implicitly by its source MAC address.

Please note that this solution does not require VXLAN encapsulation because the DSNs in the ToR are in the same layer 2 domain as the servers.

Comparing SmartSwitches with SmartNICs

The following table compares SmartSwitches and SmartNICs according to several different parameters:

	SmartNIC	**SmartSwitch**
Server applicability	Greenfield	Brownfield and Greenfield
Bare metal support	Yes	Yes
Requires drivers on server	Yes	No
Performance	Highest	High
Cost per server	High	Low
Support for RDMA and storage	Yes	No
Certification by OEM	Yes	No

The "applicability" of the SmartSwitch includes brownfield, since it does not require opening servers to change the NIC, an operation that is rarely performed on existing servers.

The "Bare Metal Support" is excellent in the case of a SmartSwitch, since it does not require touching the server at all. The "server" can be any device with an Ethernet port, for example, an industrial PLC (Programmable Logic Controller).

SmartNICs require drivers in the server OS. This requirement may not be a big deal if the server is running a recent version of a primary OS, but in some cases, bare-metal servers run old or customized OSes for which drivers are not available or certified.

The performance of a SmartNIC is considered the highest since a DSN is fully dedicated to a server. Advanced SmartSwitch implementations may overcome this obstacle by load balancing sessions coming from a server across multiple DSNs.

The downside of SmartNICs is high cost since each server needs to have a full DSN, even if it is lightly utilized.

SmartSwitches are designed to implement services for classical network traffic effectively. The same does not apply for Storage and RDMA services that require functionalities on the initiators and the targets. For these kinds of applications, SmartNICs maintain an advantage.

Enterprise and Cloud customers alike want NICs to be certified by the vendor of the servers they deploy them into, which is not trivial and often requires a long time. Since SmartSwitches do not require any hardware to be installed on servers, they do not incur in this issue.

The Key Difference in Architecture: The Pensando Flow-Based Engine

We have previously mentioned that:

SmartSwitches have a different forwarding mechanism that is stateful. Each packet between host A and host B is processed as a part of a session between A and B. The session comprises two unidirectional flows, one from A to B and one from B to A.

Sessions and flows are identified using an extended five-tuple that includes the basic five fields (IP source and destination addresses, protocol type, TCP/UDP source destination ports) and other information like input interface, MAC addresses, TCP flags, etc.

When you build hardware capable of implementing this new forwarding mechanism at wire speed, many new features become possible.

This hardware is the *Pensando flow-based engine*, which:

- identifies and tracks all the flows for all the sessions.
- uses this information to forward packets and to apply services
- operates at wire rate
- tracks million of flows simultaneously

Traditional switching platforms have no concept of session state and flow tracking and this drastically limits their capability to implement services. Another limitation of traditional switches is that ACLs lack scale typically due to a shared TCAM (ternary content-addressable memory) implementation and a flat policy model. They are also limited to the classical five fields of the 5-tuple, without relation to previous packets (no flow tracking).

The Pensando flow-based engine stores each flow in a flow table. Each flow table entry includes counters and timers. For the first time in the history of networking, it is possible to provide precise information on how each flow behaves, including parameters like bytes and packet exchange, delays, jitter, and bandwidth. Previously this information was available only at the aggregate level, for example, at the interface level. This new granularity is what we call "Telemetry," It allows us to pinpoint performance issues at the network and application level.

This telemetry information may be coupled with thresholds so that the news of flows misbehaving according to some policy definition can be streamed directly from the SmartSwitch hardware to an external application for further analysis, without the intervention of the control or management plane.

This capability to track sessions is the foundation upon which all stateful services are built.

A first example is NAT in masquerading mode. Multiple IP clients that use private IP addresses use a single public IP address. Stateful forwarding solves this problem very elegantly: when the outgoing flow is permitted and the outgoing flow entry is created in the flow table, the associated incoming flow entry is also created. The two entries are bound in a session and provide the appropriate five-tuple mapping to select the right private IP address.

A second example is a stateful firewall. Whenever a firewall is deployed, it is essential to audit its operation to detect incoming attacks and identify other problematic situations due to misconfiguration. This is usually accomplished by analyzing the "firewall log." The Pensando flow-based engine naturally creates the firewall log entries each time it allows, denies, or modifies a session and the associated flows. It also supplements this information with detailed telemetry data that gives a clear time view of the events.

Services

The Pensando flow-based engine is the foundation to create all the stateful services that are the ultimate reason why customers are interested in SmartSwitches. All of these services share the concepts of sessions and flows and draw performance and scale benefits because the Pensando flow-based engine is implemented in hardware with the programmable P4 architecture described later.

Business and Economic Benefits of SmartSwitches

The modern networking paradigm is shifting away from legacy designs that support multiple networking tiers and disparate network and security appliances (both physical and virtual). Moreover, many enterprise organizations are breaking down separate siloed teams and adopting more "cloud like" operational and organizational models. SmartSwitches provide the most efficient solution to offer services in this context, while at the same time leveraging the most up to date technology the industry has to offer. In particular, SmartSwitches are uniquely positioned to satisfy modern data center security requirements:

- Data center workloads should only be allowed to communicate as needed (zero trust networking)
- East-West network traffic visibility
- Adapt to changes

Software Licensing and Support Costs

In a scenario where software appliances are deployed as agents on hosts to implement a security and/or firewall functionality, software licensing and support fees are typically both one-time and ongoing. Specifically, perpetual host or socket licenses, along with year over year growth because of typical server expansion, will greatly impact any financial analysis over a three year period. Yearly support and service associated with these licenses must also be considered when comparing the total cost of such solutions to a SmartSwitch deployment.

Network Infrastructure Costs

Growing organizations that want to expand their existing infrastructure will find that SmartSwitches decrease the need for additional switching infrastructure and security hardware, potentially saving millions over a couple of years.

In the traditional approach services require additional infrastructure to connect service appliances, which is not just additional ports, but also additional capacity to enable tromboning of traffic to and from such appliances. Moreover, offering visibility services may require to build complete parallel networks, typically referred to as packet broker networks or TAP networks, to transport mirrored packets to analytics applications.

SmartSwitches, not only eliminate the need to acquire service appliances, but also the network infrastructure to support them because they are deployed in compute domains directly as ToR switches and provide the needed services right as the traffic enters and exits the servers.

As they expand and upgrade their data center, organizations scale down their infrastructure and avoid service appliance purchases, which leads to significant capital and operational cost avoidance compared to maintaining legacy network and appliance hardware and software.

Moreover, customers can divest from their existing infrastructure, incrementally retiring existing equipment (both switches and appliances) and avoiding the corresponding costs (licenses, support, power, cooling, etc.), as SmartSwitches subsume the functionality.

Stability and Manageability

Leveraging the full capability of SmartSwitches to address security posture (patching, updating, administering) greatly reduces the cost of maintenance and support typically associated with the legacy architectures.

End users of the applications supported by SmartSwitches benefit from more predictable and stable environments. Elimination traffic steering and tromboning directly translates into more predictable, better performing data center environments for whatever application type. Moreover, SmartSwitches are compatible with modern network design (i.e., a collapsed spine/leaf two-tier Clos topology of layer 3 routers) which results in a more stable fabric when compared to the three-tier legacy design with layer 2 functionality on some devices, layer 3 on others, service redirection functions, etc.. With everything collapsed and "meshed" in the spine/leaf design, the network is also tolerant of device failure, and overall less complex from a configuration and protocol management standpoint.

As infrastructure and use cases become more complicated, with traditional solutions administrators can find complications in providing the appropriate level of performance and availability to end users. This equates to end users experiencing a performance degradation or complete lack of application availability due to the inability for administrators and critical services, such as firewalls and load balancers, to provide resources in a timely manner.

Putting It All Together

In summary, there are multiple criteria to evaluate when considering the return on investment (ROI) or Total Cost of Ownership (TCO) of SmartSwitches:

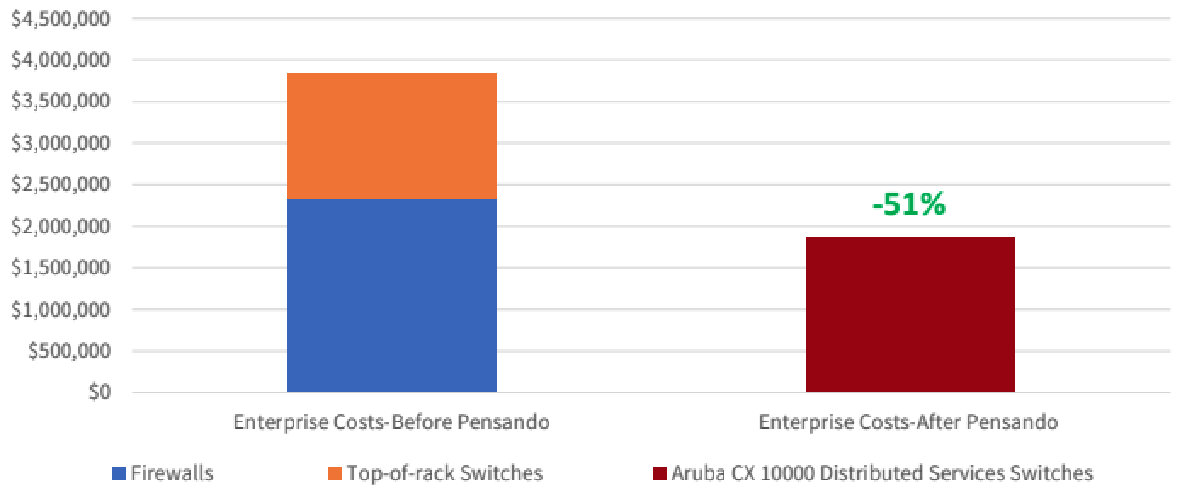
- Reduction in overall infrastructure (network devices and appliances)
- Reduction in power and cooling requirements
- Reduction in software and license costs
- Reduction in service fees and support contracts
- Reduction in management tools
- Reduction in operational expenses
- Highest security and performance benefits

When all of the above are taken into account, the TCO of a SmartSwitch-based deployment can be significantly lower than traditional solutions.

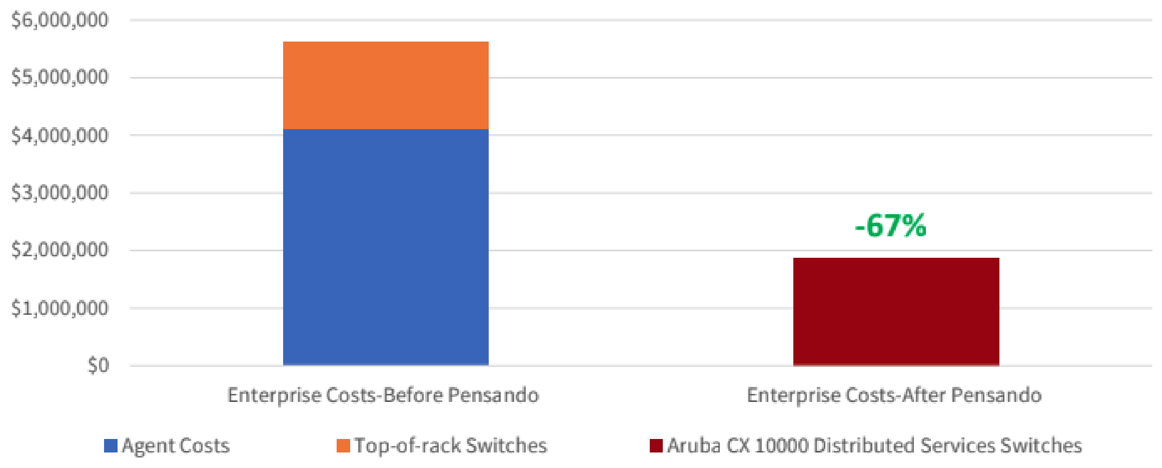
TCO Example

For example, Pensando has partnered with Aruba (a Hewlett Packard Enterprise company) to build a SmartSwitch, the Aruba CX 10000. The following charts (*) compare typical three year costs of the various hardware and software components needed in traditional solutions (hardware appliances and software agents, respectively), as well as their support fees, to the cost of a solution based on Aruba CX 10000 SmartSwitches.

As shown in the figure below, the cost of a solution based on hardware firewalls and ToR switches in an enterprise data center with around 2,000 servers over three years is roughly $2 million higher than when SmartSwitches are deployed in the same scenario, which represents a 51% cost saving.



The following figure shows that the cost is even higher if in the same 2,000 server data center firewall and flow-based telemetry services are implemented through software agents. In this case a SmartSwitch-based solution provides 67% cost savings.



Source: Enterprise Strategy Group

(*) Data and charts extracted from the whitepaper "Economic Benefits of Deploying Aruba CX 10000 with Pensando" available at https://pensando.io/wp-content/uploads/2022/02/CX-10000-economic-benefits.pdf