Computer Fraud and Security
ISSN (online): 1873-7056

Virtual Switches and Network Overlays: The Foundation of Modern SDNs

Rishi Kanth Alapati
University of Southern California, Los Angeles, USA

Abstract

Software-Defined Networking (SDN) has radically redefined the network architecture of the XXI century by decoupling the control plane and the data plane, allowing unprecedented centralization, control, and automation. This technical article explores two foundational building blocks that underlie modern SDN deployments: virtual switches and network overlays. Virtual switches are software-defined Layer 2 devices deployed inside hypervisors, which deliver connectivity to virtualized workloads and have programmable forwarding tables that are responsive to centralized controllers. Multi-tenancy Network overlay technologies, such as VXLAN and GENEVE, address multi-tenancy issues by applying an encapsulation of Ethernet frames into IP packets to form logical network topologies that have no physical boundaries. A combination of these technologies provides a highly scalable architecture in which thousands of isolated virtual networks may co-exist on the shared physical infrastructure. It discusses the internal layouts of these components and how they integrate with each other, and design principles are employed that facilitate cloud-scale deployments. Through analysis of the interaction between distributed elements of forwarding and centralized control, the article gives network engineers the necessary information to design and operate modern cloud infrastructure.

Keywords: Network Virtualization, Software-Defined Networking, Overlay Networks, Virtual Switches, Multi-Tenancy

1. Introduction

In the cloud-enabled infrastructure era, SDN has transformed network design, deployment, and management. Through separating the control plane from the data plane, SDN supports centralized network control and policy compliance that cannot be achieved by traditional networking methods. This technical article examines two pivotal building blocks central to current SDN solutions: virtual switches and network overlays.

The emergence of SDN represents one of the most significant architectural shifts in networking over the past decade. According to the Open Networking Foundation, this paradigm fundamentally changes network architecture by creating a logical separation between network control functions and forwarding operations, making networks more dynamic, manageable, and adaptable to the high-bandwidth requirements of modern applications [1]. Overlays and virtual switches are now the foundation technologies facilitating such a shift, offering the flexibility and scale necessary to cloud environments, yet still maintaining compatibility with existing infrastructure.

Virtual switches serve as the main network interface to virtualized workloads, using advanced packet processing pipelines that reflect and augment the capability of traditional switches. These switch technologies, based on software, form the base for network virtualization through interconnecting virtual machines and containers to the rest of the network infrastructure. At the same time, overlay technologies such as VXLAN and GENEVE solve the issue of limitations in legacy network segmentation by encapsulating Layer 2 frames in Layer 3 packets, establishing logical tunnels between endpoints independent of physical topology. As Koponen et al. illustrated in their multi-tenant datacenter research, this method allows independent evolution of physical and logical networks as well as scale demands many orders of magnitude beyond traditional networking limitations [2].

This paper examines the architecture, deployment, and operational characteristics of the virtual switches and network overlays within the existing SDN implementations. The awareness of these underlying technologies can assist the network architects and engineers in the creation, implementation, and operation of rich network infrastructure that is demanded in current cloud and enterprise environments.

2. The Architecture of Software-Defined Networks

Software-Defined Networking is a network design paradigm shift. The traditional networks integrate both control (routing, policy) and data transfer within every network device. SDN dissociates these functions, creating a centralized control plane that deploys programs on a large number of data plane devices. The resulting decoupling is incredibly flexible, automatable, and scalable.

The fundamental design of SDN centers on three fundamental planes, namely, the data, control, and management planes. This separation, as described by Kreutz et al., produces a "logically centralized control model" that shatters the vertical integration of legacy networks, in effect turning the underlying infrastructure into an extensible platform that can be manipulated with well-specified APIs [3]. Their in-depth survey outlines how this architecture allows network operators to provision sophisticated traffic management schemes without being bound by the limitations imposed by distributed control protocols such as OSPF or BGP. The centralized control plane has a global understanding of network resources, enabling optimized path choice, dynamic allocation of resources, and uniform policy application to heterogeneous infrastructure.

In modern enterprise network virtualization platforms, this design is realized through hypervisor-level virtual switches and overlay networks that virtualize the physical network infrastructure. Collectively, these technologies form a rational network fabric capable of interconnecting multiple data centers with consistent policy enforcement and segmentation. Casado et al. introduced this architectural vision as the natural extension of SDN, stating that hypervisor-based switching offers the optimal insertion point for network virtualization [4]. Their article, "Fabric: A Retrospective on Evolving SDN," explains how edge virtualization using hypervisor switches together with a global control plane produces a scale-out network abstraction layer. The model makes the physical network operate as a mere IP fabric while complex network services are handled at the edge, allowing organizations to have thousands of logical networks deployed without touching the underlying infrastructure. This design has been especially useful in multi-tenant environments where policy consistency and network isolation are overriding needs.

Feature	Traditional Networks	Software-Defined Networks
Control Plane Location	Distributed (in each device)	Centralized
Policy Enforcement	Device-by-device configuration	Global application
Network Visibility	Limited to the local device	Complete network view
Programming Interface	Vendor-specific CLIs	Open APIs
Service Implementation	Throughout network	Primarily at the network edge
Configuration Model	Manual, device-specific	Automated, abstracted
Scalability Approach	Add more physical devices	Logical network segmentation
Traffic Engineering	Based on distributed protocols (OSPF, BGP)	Centralized optimization
Multi-tenancy Support	Limited by VLAN constraints	Extensive through overlays
Resource Allocation	Static, predetermined	Dynamic, on-demand

Table 1: SDN Architectural Components vs. Traditional Networking [3, 4]

3. Virtual Switches: The Network Interface of the Hypervisor

vSwitches act as the main network interface for virtualized workloads. In contrast to hardware switches, these are purely software-based, often deployed inside hypervisors or container runtimes. They provide Layer-2 connectivity for virtual machines and containers and connect them into the broader network infrastructure.

The internal structure of a virtual switch mirrors many features of physical switches: you have virtual port management, MAC forwarding tables, VLAN tagging, uplink aggregation, and more. Pfaff et al. describe how **Open vSwitch** (OVS)

implements a flexible packet-processing architecture using multiple flow tables that are programmable via OpenFlow and other control protocols [5]. The OVS design separates the **fast path** (kernel-level packet forwarding) from the **slow path** (feature-rich processing in userspace), achieving a balance between performance and extensibility. This split allows OVS to maintain forwarding state at scale (thousands of VMs) while adapting to changes such as live migrations and dynamic policy updates.

Beyond those core capabilities, modern OVS deployments incorporate significant enhancements to address performance and scalability demands:

- **DPDK-based datapath**: OVS can be built with DPDK support to operate fully in userspace, bypassing the kernel network stack for higher per-packet performance [13].
- **Multi-core scaling**: In OVS 2.16 and above, new mechanisms for spreading PMD (Poll Mode Driver) threads across CPU cores improve throughput and reduce per-thread bottlenecks [14].
- **Improved DPDK statistics**: From OVS 2.17 onward, per-queue statistics for DPDK ports have been exposed via OVSDB, enabling finer-grained observability of how traffic is distributed across queues and cores [15].
- **Hardware offload and NIC/DPUs acceleration**: OVS supports offloading portions of its datapath to NIC hardware or DPU engines to reduce host CPU usage. For example, NVIDIA's OVS-DOCA offload mode allows certain flows to be executed directly on NIC hardware while preserving compatibility with the OVS control plane [16].
- **Kernel offload / hardware offload**: OVS 2.8+ supports "OVS Hardware Offload," in which data-plane operations are offloaded to NIC hardware (e.g. SR-IOV/VF representors) while leaving the control-plane logic in software [17]

Virtual switches therefore have evolved to support **dynamic, programmable forwarding tables** that SDN controllers can update in real time across the network fabric. Pettit et al. examine how modern virtual switches balance **performance**, **feature richness**, and **management simplicity**, often leveraging hardware offload paths and hybrid datapaths to reconcile competing demands [6]. By enabling high-level match-action tables at the hypervisor edge, virtual switches reduce east–west traffic inside the data center and enforce security policies that follow workloads as they migrate across hosts.

Feature	Traditional Switch Implementation	Open vSwitch Implementation	Benefits
Packet Processing	Monolithic architecture	Split fast-path (kernel) and slow-path (userspace)	Improved performance with feature flexibility
State Management	Limited MAC table entries	Support for thousands of VM states	Accommodates large-scale virtualization
Policy Enforcement	Network perimeter only	Adjacent to workloads	Reduced east-west traffic overhead
Configuration	Static, device-based	Programmable via OpenFlow	Dynamic policy updates
Traffic Management	VLAN-based (4094 limit)	Flow-based tables	More granular control
Hardware Integration	Purpose-built ASICs	Software with hardware offloading	Balance of flexibility and performance
Mobility Support	Limited	VM migration tracking	Consistent policy during workload movement
Forwarding Decision	Fixed pipeline	Programmable match-action tables	Customizable packet handling

Scalability	New hardware required	Software-defined capacity	Cost-effective expansion
Edge Services	Basic L2 forwarding	Advanced security and routing	Consolidated network functions

Table 2: Virtual Switch Architecture: Kernel vs. Userspace Components [5, 6]

4. Overlay Networks: Building Logical Topologies

Network overlays address one of the hardest multi-tenant problems: how to build isolated network segments that may span physical boundaries. This is supported by technologies like VXLAN (Virtual Extensible LAN) and GENEVE (Generic Network Virtualization Encapsulation), which encapsulate Layer 2 Ethernet packets into Layer 3 packets.

Such an encapsulation process creates logical tunnels among the virtual switches, such that the Layer 2 adjacency can be accomplished regardless of the underlying physical network structure. Mahalingam et al. specify the VXLAN architecture in RFC 7348, explaining how the protocol utilizes a 24-bit Virtual Network Identifier (VNI) to enable up to 16 million logical networks on common infrastructure [7]. Their specification outlines how VXLAN encapsulation appends a header with the VNI and other control data to every frame, allowing the physical network to forward traffic between tunnel endpoints without mixing tenant network traffic. This design decouples the logical network topology seen by workloads from the physical network, solving the scalability constraint of traditional VLANs while still offering Layer 2 semantics to applications that need them.

The technology behind overlay networks includes a number of key elements: encapsulation headers, tunnel endpoints, and tunnel management systems. Gross et al. cover these elements in their in-depth discussion of GENEVE, which takes the overlay approach further with increased protocol extensibility and flexibility [8]. Their publication outlines how tunnel endpoints, normally realized inside virtual switches, are responsible for decapsulation and encapsulation processing at the edge of the network. These endpoints share virtual-to-physical addressing mappings to direct traffic between workloads across physical locations. GENEVE's extensible header format supports the inclusion of rich routing and policy-enforced metadata, facilitating use cases beyond straightforward network segmentation. Tunnel setup and management by the tunnel management system creates and maintains the connectivity between endpoints, distributing information about workload reachability and location throughout the overlay network. This system commonly employs a centralized controller to spread configuration data and provide a common view of the network topology, allowing for unimpeded workload mobility and dynamic network reconfiguration without losing application connectivity.

Feature	VXLAN	GENEVE	Traditional VLAN
Network Identifier	24-bit VNI	Flexible length	12-bit VLAN ID
Maximum Segments	16 million	Extensible	4,094
Encapsulation Type	MAC-in-UDP	Extensible format	802.1Q tag
Header Size	Fixed	Variable	Fixed
Metadata Support	Limited	Extensive	None
Physical Network Requirement	IP routing	IP routing	L2 adjacency
Controller Integration	Required for scale	Required for scale	Optional
Workload Mobility	Supported across subnets	Supported across subnets	Limited to the L2 domain
Cross-Datacenter Support	Native	Native	Requires L2 extension
Protocol Extensibility	Limited	High	None
Standardization	RFC 7348	IETF Draft	IEEE 802.1Q

Table 3: Overlay Protocol Segmentation Capabilities Comparison [7, 8]

5. Virtual Switch and Overlay Integration

In data center production environments, virtual switches and overlay networks operate together to provide an end-to-end network fabric. The virtual switch maps VMs to the correct overlay segments based on control plane configuration. When a VM transmits traffic to a remote endpoint, the virtual switch encapsulates the frame in the correct overlay headers and sends it over the physical network to the destination tunnel endpoint.

The combination of virtual switches and overlay networks forms a distributed system, which has to ensure consistency throughout potentially thousands of endpoints. Based on GeeksforGeeks' thorough review of network overlays in distributed systems, the combination brings a number of important benefits such as location independence, network isolation, and easy migration [9]. Their technical overview describes how overlay networks produce an abstraction layer that isolates the logical network view from the physical network topology, allowing workloads to communicate as if they were co-located on the same local network, independent of their physical location. When a virtual machine initiates communication, the virtual switch first checks if the destination is local or remote. For local destinations, traffic is simply switched within the host. For distant locations, the switch encapsulates, placing the overlay headers as required prior to sending the packet over the physical network. The encapsulation would entail inserting the overlay network identifier (e.g., the VNI in VXLAN) and other metadata into the packet in a way that it is delivered to the target destination endpoint without causing the networks of different tenants to interact.

Such systems provide previous ARP (Address Resolution Protocol) and ND (Neighbor Discovery) information in IP discovery modules, which aid in advanced capabilities such as ARP suppression, IP spoofing control, and distributed routing. NVIDIA's tech blog explains how these features make network performance and security much better in virtualized systems [10]. Their reasoning describes ARP suppression as one of the most effective optimizations available in overlay networks, as it significantly minimizes broadcast traffic that otherwise would take up bandwidth throughout the entire fabric. By responding to ARP requests locally from distributed MAC-to-IP mappings stored by the control plane, virtual switches remove extraneous broadcast traffic while also serving up quicker responses to virtual machines. Likewise, IP spoofing detection takes advantage of the virtual switch's location at the network edge to check source addresses against anticipated values for malicious impersonation avoidance. Distributed routing is presented as especially useful in contemporary data centers where east-west traffic is the norm, allowing Layer 3 forwarding to be done at the virtual switch level without having to make the traffic travel through centralized physical routers. This ability decreases latency by a large margin and enhances overall network performance with steady policy enforcement via the centralized control plane.

Function	Traditional Implementation	Integrated SDN Implementation	Operational Impact
VM-to-Network Assignment	VLAN tagging at the access port	Logical segment mapping in vSwitch	Dynamic provisioning
Traffic Path Determination	Physical switch MAC tables	Local/remote endpoint detection	Optimized forwarding
Broadcast Handling	Network-wide flooding	ARP suppression at the edge	Reduced fabric traffic
Security Enforcement	Perimeter-based	Source validation at the first hop	Earlier threat prevention
Layer 3 Routing	Centralized router devices	Distributed at the virtual switch	Reduced traffic hairpinning
Workload Mobility	Complex VLAN/subnet changes	Maintained overlay membership	Seamless migration
Multi-tenancy	Limited by VLAN scope	Overlay encapsulation isolation	Scalable tenant separation
Address Resolution	Broadcast-based	Control plane distribution	Performance

			optimization
Traffic Engineering	Physical path configuration	Logical overlay path selection	Simplified management
Cross-DC Communication	Extended L2 domains	IP-based overlay tunneling	Greater flexibility
Policy Enforcement	Network device configuration	Control plane distribution	Consistent application
Failure Detection	Physical link monitoring	Tunnel endpoint monitoring	Comprehensive visibility

Table 4: Integrated Edge Services in Virtual Network Fabrics [9, 10]

6. Scaling in Enterprise and Cloud Scenarios

The above architecture allows for an unprecedented scale of network deployments. Huge cloud environments are easily supported with thousands of virtual networks running on shared infrastructure, each with its own isolated address space and security policies.

This scalability is realized through a number of design principles, which are distributed data plane operations, centralized control, stateless design, and hierarchical management. As per Microsoft Research's insightful analysis of Azure's network infrastructure, these principles are necessary to run cloud networks at a global scale [11]. In their paper, they explain how the blend of local forwarding decisions by virtual switches and centralized policy definition via the control plane results in a highly scalable architecture. The distributed data plane supports per-host independent forwarding decisions from locally cached forwarding tables and policies, avoiding potential bottlenecks in centralized packet processing. Yet these distributed elements are kept in sync through a logically centralized control plane that preserves consistent policies across the system. This logically centralized control plane distributes configuration changes, updates forwarding tables, and preserves security policies being applied consistently wherever the workload resides. The authors mention that Azure employs a stateless design pattern wherever feasible, keeping the demand for complicated state synchronization between elements to a minimum and enhancing system dependability during fault conditions.

Hierarchical administration also increases scalability by structuring network resources in logical domains that can be managed and scaled autonomously. Singh et al. explore this technique in their study of Google's Jupiter network topology, which hosts one of the globe's largest cloud computing environments [12]. Their work illustrates how hierarchical management facilitates controlled scaling through the decomposition of the network into manageable units that can be upgraded and expanded independently. This technique permits cloud providers to grow their networks incrementally without impacting current services, a requirement imperative to production deployments. The paper outlines the way that Google's deployment structures network resources into hierarchical tiers, with varying policies and management styles at each tier. This approach allows network operators to manage resources at suitable levels of abstraction, ranging from individual virtual networks through to whole data center fabrics. The authors point out how this design enables Google to have tens of thousands of virtual networks on its infrastructure, each with a distinct isolated address space and security policies. By integrating these design principles—distributed data plane, centralized control, stateless design, and hierarchical management—cloud providers can create network fabrics that scale up to millions of endpoints but have consistent performance and security.

Conclusion

The fundamental technologies that can be used to effect the transition to software-defined infrastructure are virtual switches and network overlays. These technologies remove the constraints imposed on the traditional networking methods by virtualizing network functions at the hypervisor edge and building logical topologies with encapsulation, and are compatible with existing infrastructure. It is the architectural patterns presented, including distributed data plane functionality coupled with centralized control, stateless design principles, and hierarchical control, that form the basis of a network fabric that can be scaled up to meet application needs in the present day. With organizations still working their way toward cloud-native architectures, it is becoming more and more important that network professionals are familiar with these basic elements. The advanced interaction of virtual switches, overlay networks, and the SDN control plane allows features never before achievable with traditional networking paradigms, including granular security policies that

accompany workloads as they move between hosts and dynamic network reconfigurations without affecting service availability. These technologies are going to stay as key parts of network design as infrastructure continues to grow more abstract, more automatable, and more scalable.

References

[1] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," 2012. [Online]. Available: https://opennetworking.org/wp-content/uploads/2011/09/wp-sdn-newnorm.pdf

[2] Teemu Koponen et al., "Network virtualization in multi-tenant datacenters," NSDI'14: Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2616448.2616468

[3] Diego Kreutz et al., "Software-Defined Networking: A Comprehensive Survey," ResearchGate, 2014. [Online]. Available: https://www.researchgate.net/publication/262805723_Software-Defined_Networking_A_Comprehensive_Survey

[4] Martín Casado et al., "Fabric: A Retrospective on Evolving SDN," ACM, 2012. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2342441.2342459

[5] Ben Pfaff et al., "The Design and Implementation of Open vSwitch," in the Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15), 2015. [Online]. Available: https://www.usenix.org/system/files/conference/nsdi15/nsdi15-paper-pfaff.pdf

[6] Justin Pettit et al., "Virtual Switching in an Era of Advanced Edges,". [Online]. Available: https://www.openvswitch.org/support/papers/dccaves2010.pdf

[7] M. Mahalingam et al., "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Aug. 2014. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7348

[8] J. Gross et al., "Geneve: Generic Network Virtualization Encapsulation," Internet Engineering Task Force, 2020. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-nvo3-geneve-16

[9] GeeksforGeeks, "Network Overlays in Distributed Systems," 2025. [Online]. Available: https://www.geeksforgeeks.org/system-design/network-overlays-in-distributed-systems/

[10] Rama Darbha, "Optimizing Your Data Center Network," NVIDIA Developer Blog, 2022. [Online]. Available: https://developer.nvidia.com/blog/optimizing-your-data-center-network/

[11] Albert Greenberg et al., "VL2: A Scalable and Flexible Data Center Network," Microsoft, 2009. [Online]. Available: https://www.microsoft.com/en-us/research/publication/vl2-a-scalable-and-flexible-data-center-network/

[12] Arjun Singh et al., "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," SIGCOMM '15: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2785956.2787508

[13] OpenvSwitch, "Open vSwitch with DPDK,". [Online]. Available: https://docs.openvswitch.org/en/latest/intro/install/dpdk/

[14] Kevin Traynor, "Improve multicore scaling in Open vSwitch DPDK," Red Hat Developer, 2021. [Online]. Available: https://developers.redhat.com/articles/2021/11/19/improve-multicore-scaling-open-vswitch-dpdk

[15] David Marchand, "A statistics update in Open vSwitch user space datapath," Red Hat Developer, 2023. [Online]. Available: https://developers.redhat.com/articles/2023/09/18/statistics-update-open-vswitch-user-space-datapath

[16] Nvidia, "A statistics update in Open vSwitch user space datapath," DOCA Documentation v2.5.3 LTS. [Online]. Available: https://docs.nvidia.com/doca/archive/2-5-3/openvswitch+offload/index.html

[17] OVN Kubernetes, "OVS Acceleration with Kernel datapath,". [Online]. Available: https://ovn-kubernetes.io/features/hardware-offload/ovs-kernel/