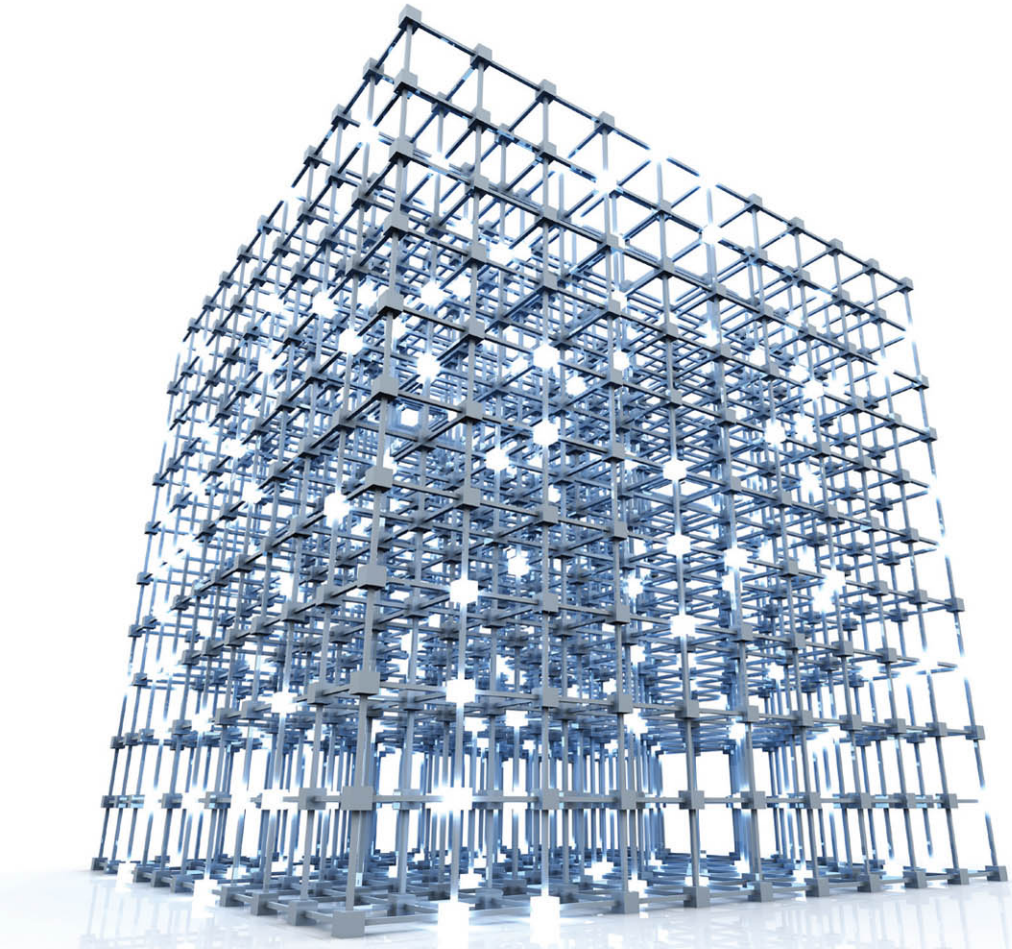
William Stallings

Foundations of Modern Networking

SDN, NFV, QoE, IoT, and Cloud



THE WILLIAM STALLINGS BOOKS ON COMPUTER AND DATA COMMUNICATIONS TECHNOLOGY

DATA AND COMPUTER COMMUNICATIONS, TENTH EDITION
A comprehensive survey that has become the standard in the field, covering (1) data communications, including transmission, media, signal encoding, link control, and multiplexing; (2) communication networks, including circuit and packet switched, Frame Relay, ATM, and LANs; (3) the TCP/IP protocol suite, including IPv6, TCP, MIME, and HTTP, as well as a detailed treatment of network security. **Received the 2007 Text and Academic Authors Association (TAA) award for the best Computer Science and Engineering Textbook of the year.**

***WIRELESS COMMUNICATION NETWORKS AND SYSTEMS* (with Cory Beard)**
A comprehensive, state-of-the art survey. Covers fundamental wireless communications topics, including antennas and propagation, signal encoding techniques, spread spectrum, and error-correction techniques. Examines satellite, cellular, wireless local loop networks, and wireless LANs, including Bluetooth and 802.11. Covers wireless mobile networks and applications.

***COMPUTER SECURITY, THIRD EDITION* (with Lawrie Brown)**
A comprehensive treatment of computer security technology, including algorithms, protocols, and applications. Covers cryptography, authentication, access control, database security, cloud security, intrusion detection and prevention, malicious software, denial of service, firewalls, software security, physical security, human factors, auditing, legal and ethical aspects, and trusted systems. **Received the 2008 TAA award for the best Computer Science and Engineering Textbook of the year.**

OPERATING SYSTEMS, EIGHTH EDITION
A state-of-the art survey of operating system principles. Covers fundamental technology as well as contemporary design issues, such as threads, SMPs, multicore, real-time systems, multiprocessor scheduling, embedded OSs, distributed systems, clusters, security, and object-oriented design. **Third, fourth and sixth editions received the TAA award for the best Computer Science and Engineering Textbook of the year.**

CRYPTOGRAPHY AND NETWORK SECURITY, SIXTH EDITION
A tutorial and survey on network security technology. Each of the basic building blocks of network security, including conventional and public-key cryptography, authentication, and digital signatures, are covered. Provides a thorough mathematical background for such algorithms as AES and RSA. The book covers important network security tools and applications, including S/MIME, IP Security, Kerberos, SSL/TLS, network access control, and Wi-Fi security. In addition, methods for countering hackers and viruses are explored. **Second edition received the TAA award for the best Computer Science and Engineering Textbook of 1999.**

NETWORK SECURITY ESSENTIALS, FIFTH EDITION
A tutorial and survey on network security technology. The book covers important network security tools and applications, including S/MIME, IP security, Kerberos, SSL/TLS, network access control, and Wi-Fi security. In addition, methods for countering hackers and viruses are explored.

***BUSINESS DATA COMMUNICATIONS, SEVENTH EDITION* (with Tom Case)**
A comprehensive presentation of data communications and telecommunications from a business perspective. Covers voice, data, image, and video communications and applications technology and includes a number of case studies. Topics covered include data communications, TCP/IP, cloud computing, Internet protocols and applications, LANs and WANs, network security, and network management.

COMPUTER ORGANIZATION AND ARCHITECTURE, TENTH EDITION
A unified view of this broad field. Covers fundamentals such as CPU, control unit, microprogramming, instruction set, I/O, and memory. Also covers advanced topics such as multicore, superscalar, and parallel organization. **Five-time winner of the TAA award for the best Computer Science and Engineering Textbook of the year.**

With these limitations in mind, this chapter provides an overview of the SDN application plane. Section 6.1 begins with an overview of the SDN application plane architecture. Section 6.2 looks at a key component of that architecture, the network services abstraction layer. The remaining sections look at six major application areas that can be supported by SDN. These sections also describe a number of specific examples. The examples were chosen to give the reader a feel for the range of applications that can benefit from an SDN infrastructure.

6.1 SDN Application Plane Architecture

The application plane contains applications and services that define, monitor, and control network resources and behavior. These applications interact with the SDN control plane via application-control interfaces, for the SDN control layer to automatically customize the behavior and the properties of network resources. The programming of an SDN application makes use of the abstracted view of network resources provided by the SDN control layer by means of information and data models exposed via the application-control interface.

This section provides an overview of application plane functionality, depicted in Figure 6.1. The elements in this figure are analyzed through a bottom-up approach, and subsequent sections provide detail on specific application areas.

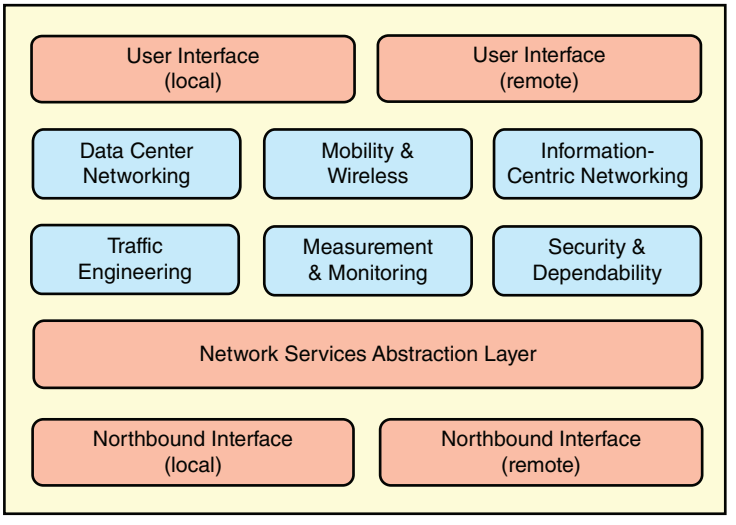


FIGURE 6.1 SDN Application Plane Functions and Interfaces

Northbound Interface

As described in Chapter 5, "SDN Control Plane," the northbound interface enables applications to access control plane functions and services without needing to know the details of the underlying network switches. Typically, the northbound interface provides an abstract view of network resources controlled by the software in the SDN control plane.

Figure 6.1 indicates that the northbound interface can be a local or remote interface. For a local interface, the SDN applications are running on the same server as the control plane software (controller network operating system). Alternatively, the applications could be run on remote systems and the northbound interface is a protocol or application programming interface (API) that connects the applications to the controller network operating system (NOS) running on central server. Both architectures are likely to be implemented.

An example of a northbound interface is the REST API for the Ryu SDN network operating system, described in Section 5.4.

Network Services Abstraction Layer

RFC 7426 defines a network services abstraction layer between the control and application planes and describes it as a layer that provides service abstractions that can be used by applications and services. Several functional concepts are suggested by the placement of this layer in the SDN architecture:

- This layer could provide an abstract view of network resources that hides the details of the underlying data plane devices.
- This layer could provide a generalized view of control plane functionality, so that applications could be written that would operate across a range of controller network operating systems.
- This functionality is similar to that of a hypervisor or virtual machine monitor that decouples applications from the underlying OS and underlying hardware.
- This layer could provide a network virtualization capability that allows different views of the underlying data plane infrastructure.

Arguably, the network services abstraction layer could be considered to be part of the northbound interface, with the functionality incorporated in the control plane or the application plane.

A wide range of schemes have been developed that roughly fall into this layer, and a full treatment is beyond our scope. Section 6.2 provides several examples for a better understanding.

Network Applications

There are many network applications that could be implemented for an SDN. Different published surveys of SDN have come up with different lists and even different general categories of SDN-based network applications. Figure 6.1 includes six categories that encompass the majority of SDN applications. Later sections of this chapter provide an overview of each area.

User Interface

The user interface enables a user to configure parameters in SDN applications and to interact with applications that support user interaction. Again, there are two possible interfaces. A user that is collocated with the SDN application server (which may or may not include the control plane) can use the server's keyboard/display. More typically, the user would log on to the application server over a network or communications facility.

6.2 Network Services Abstraction Layer

In the context of the discussion, **abstraction** refers to the amount detail about lower levels of the model that is visible to higher levels. More abstraction means less detail; less abstraction means more detail. An **abstraction layer** is a mechanism that translates a high-level request into the low-level commands required to perform the request. An API is one such mechanism. It shields the implementation details of a lower level of abstraction from software at a higher level. A network abstraction represents the basic properties or characteristics of network entities (such as switches, links, ports, and flows) is such a way that network programs can focus on the desired functionality without having to program the detailed actions.

Abstractions in SDN

Scott Shenker, an Open Networking Foundation (ONF) board member and OpenFlow researcher, indicates that SDN can be defined by three fundamental abstractions [SHEN11]: forwarding, distribution, and specification, as illustrated in Figure 6.2 and described further in the sections that follow.

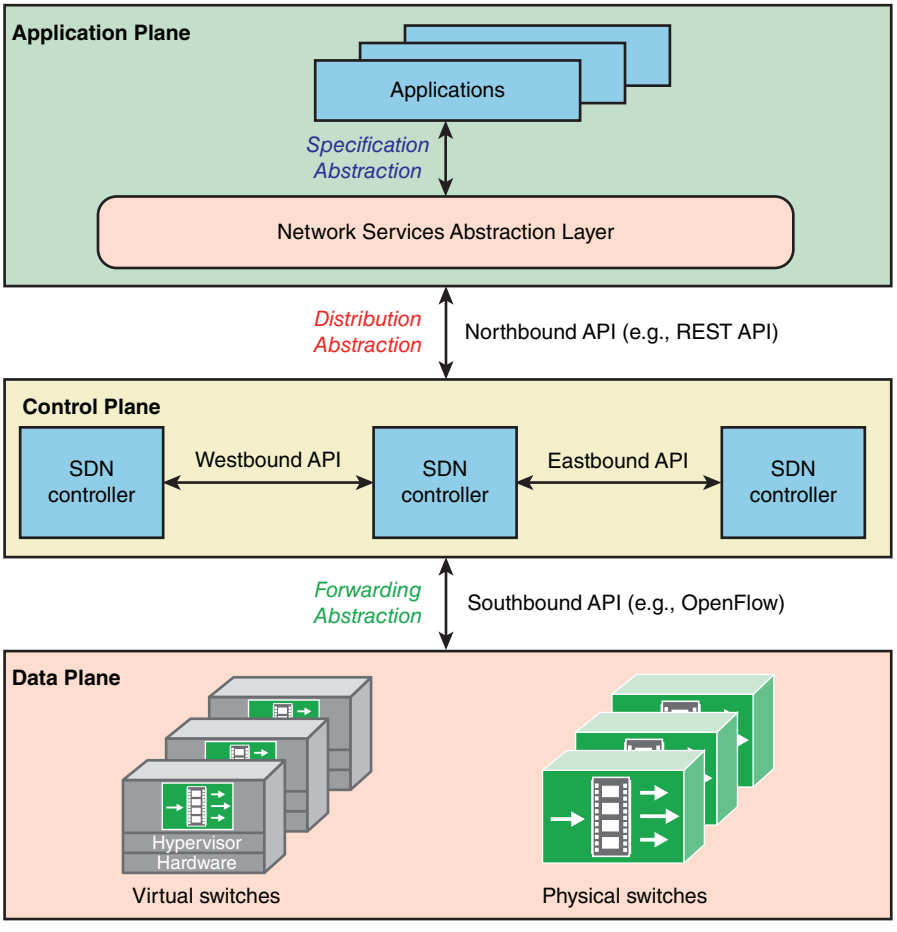


FIGURE 6.2 SDN Architecture and Abstractions

Forwarding Abstraction

← *See Sections 4.1, "SDN Data Plane," 4.2, "OpenFlow Logical Network Device"*

The forwarding abstraction allows a control program to specify data plane forwarding behavior while hiding details of the underlying switching hardware. This abstraction supports the data plane forwarding function. By abstracting away from the forwarding hardware, it provides flexibility and vender neutrality.

The OpenFlow API is an example of a forwarding abstraction.

Distribution Abstraction

This abstraction arises in the context of distributed controllers. A cooperating set of distributed controllers maintains a state description of the network and routes through the networks. The distributed state of the entire network may involve partitioned data sets, with controller instances exchanging routing information, or a replicated data set, so that the controllers must cooperate to maintain a consistent view of the global network.

This abstraction aims at hiding complex distributed mechanisms (used today in many networks) and separating state management from protocol design and implementation. It allows providing a single coherent global view of the network through an annotated network graph accessible for control via an API. An implementation of such an abstraction is an NOS, such as OpenDaylight or Ryu.

Specification Abstraction

The distribution abstraction provides a global view of the network as if there is a single central controller, even if multiple cooperating controllers are used. The specification abstraction then provides an abstract view of the global network. This view provides just enough detail for the application to specify goals, such as routing or security policy, without providing the information needed to implement the goals. The presentation by Shenker [SHEN11] summarizes these abstractions as follows:

- **Forwarding interface:** An abstract forwarding model that shields higher layers from forwarding hardware.
- **Distribution interface:** A global network view that shields higher layers from state dissemination/collection.
- **Specification interface:** An abstract network view that shields application program from details of physical network.

Figure 6.3 is a simple example of a specification abstraction. The physical network is a collection of interconnected SDN data plane switches. The abstract view is a single virtual switch. The physical network may consist of a single SDN domain. Ports on edge switches that connect to other domains and to hosts are mapped into ports on the virtual switch. At the application level, a module can be executed to learn the media access control (MAC) address of hosts. When a previously unknown host sends a packet, the application module can associate that address with the input port and direct future traffic direct to this host to this port. Similarly, if a packet arrives at one of the virtual switch ports with an unknown destination address, the module floods that packet to all output ports. The abstraction layer translates these actions into actions on the entire physical network, performing the internal forwarding with the domain.

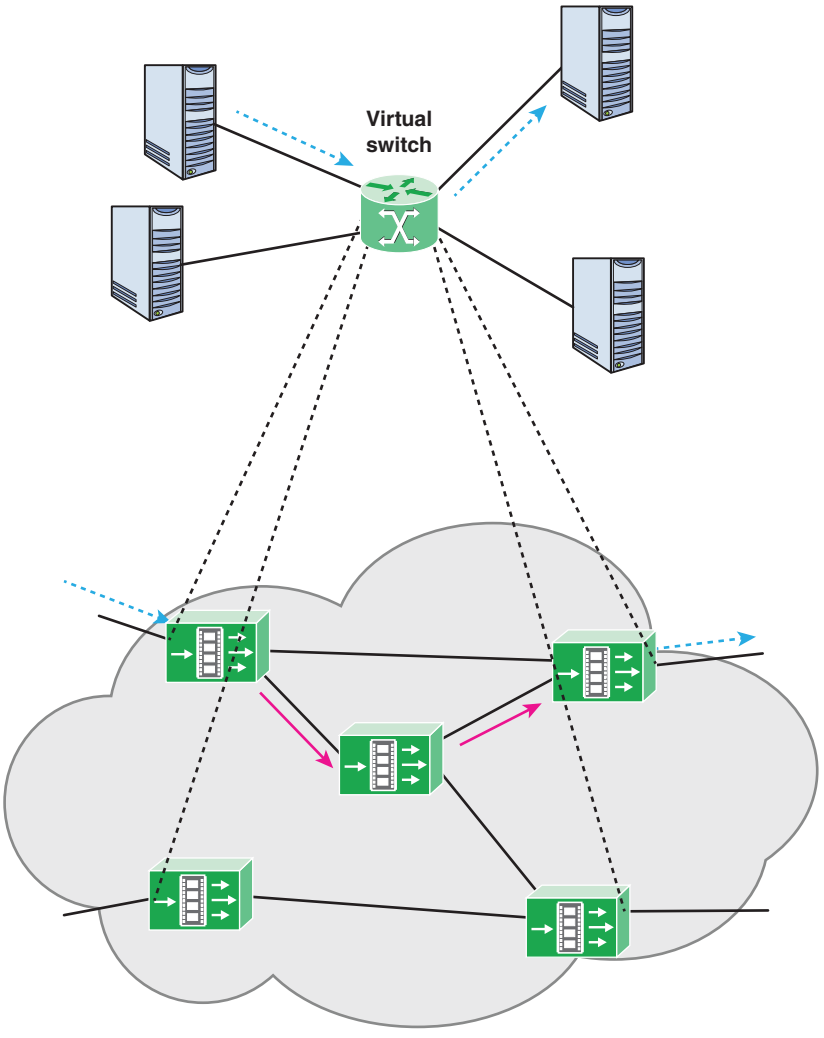


FIGURE 6.3 Virtualization of a Switching Fabric for MAC Learning

Frenetic

An example of a network services abstraction layer is the programming language Frenetic. Frenetic enables networks operators to program the network as a whole instead of manually configuring individual network elements. Frenetic was designed to solve challenges with the use of OpenFlow-based models by working with an abstraction at the network level as opposed to OpenFlow, which directly goes down to the network element level.

Frenetic includes an embedded query language that provides effective abstractions for reading network state. This language is similar to SQL and includes segments for