Chapter 5. SDN Control Plane

> The organization for the control and guidance of the trade should therefore be of so complete a character that the trade may be either dispersed about the ocean or concentrated along particular routes; or in some places dispersed and in others concentrated; and that changes from one policy to the other can be made when necessary at any time.
>
> —*The World Crisis*, Winston Churchill, 1923

Chapter Objectives: **After studying this chapter, you should be able to**

- List and explain the key functions of the SDN control plane.
- Discuss the routing function in the SDN controller.
- Understand the ITU-T Y.3300 layered SDN model.
- Present an overview of OpenDaylight.
- Present an overview of REST.
- Compare centralized and distributed SDN controller architectures.
- Explain the role of BGP in an SDN network.

This chapter continues our study of software-defined networking (SDN), focusing on the control plane (see Figure 5.1). Section 5.1 provides an overview of SDN control plane architecture, discussing the functions and interface capabilities of a typical SDN control plane implementation. Next, we summarize the ITU-T layered SDN model, which provides additional insight into the role of the control plane. This is followed by a description of one of the most significant open source SDN controller efforts, known as OpenDaylight. Then Section 5.4 describes the REST northbound interface, which has become common in SDN implementations. Finally, Section 5.5 discusses issues relating to cooperation and coordination among multiple SDN controllers.

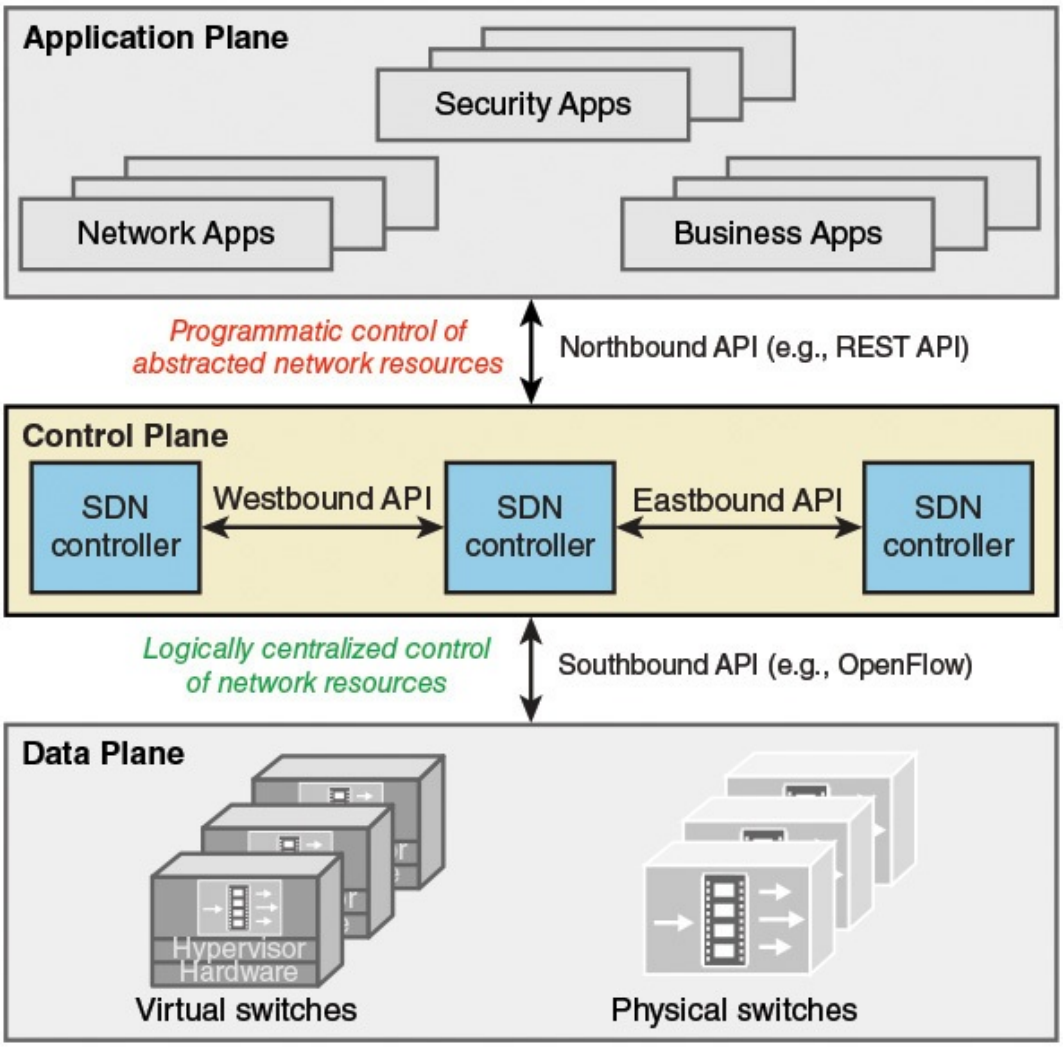

FIGURE 5.1 SDN Architecture

5.1 SDN Control Plane Architecture

The SDN control layer maps application layer service requests into specific commands and directives to data plane switches and supplies applications with information about data plane topology and activity. The control layer is implemented as a server or cooperating set of servers known as SDN controllers. This section provides an overview of control plane functionality. Later, we look at specific protocols and standards implemented within the control plane.

Control Plane Functions

Figure 5.2 illustrates the functions performed by SDN controllers. The figure illustrates the essential functions that any controller should provide, as suggested in a paper by Kreutz [KREU15], which include the following:

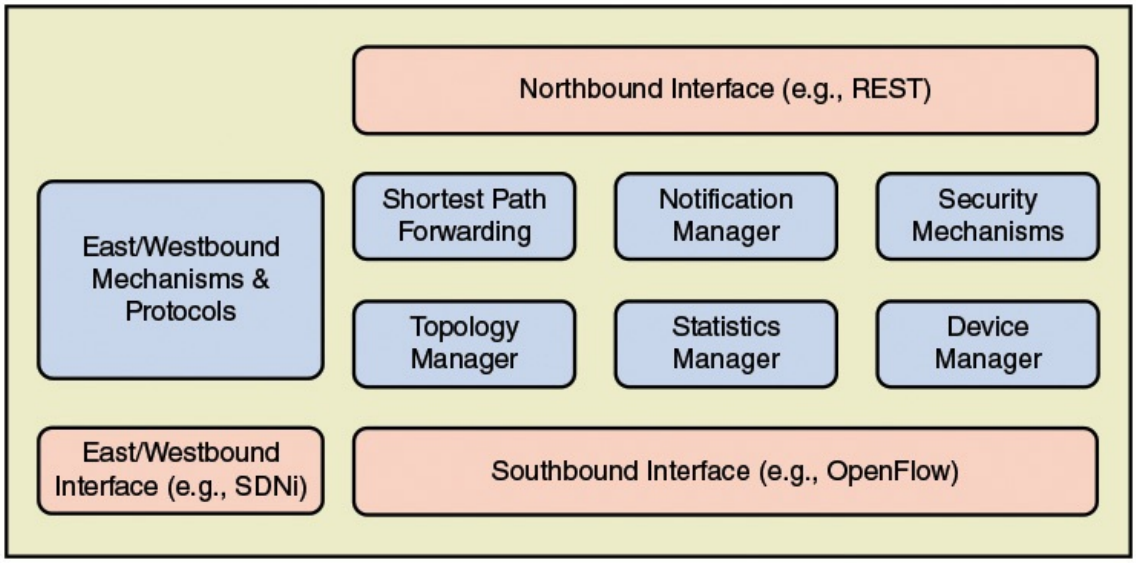

FIGURE 5.2 SDN Control Plane Functions and Interfaces

- **Shortest path forwarding:** Uses routing information collected from switches to establish preferred routes.
- **Notification manager:** Receives, processes, and forwards to an application events, such as alarm notifications, security alarms, and state changes.
- **Security mechanisms:** Provides isolation and security enforcement between applications and services.
- **Topology manager:** Builds and maintains switch interconnection topology information.
- **Statistics manager:** Collects data on traffic through the switches.
- **Device manager:** Configures switch parameters and attributes and manages flow tables.

The functionality provided by the SDN controller can be viewed as a **network operating system (NOS)**. As with a conventional OS, an NOS provides

essential services, common application programming interfaces (APIs), and an abstraction of lower-layer elements to developers. The functions of an SDN NOS, such as those in the preceding list, enable developers to define network policies and manage networks without concern for the details of the network device characteristics, which may be heterogeneous and dynamic. The northbound interface, discussed subsequently, provides a uniform means for application developers and network managers to access SDN service and perform network management tasks. Further, well-defined northbound interfaces enable developers to create software that is independent not only of data plane details but to a great extent usable with a variety of SDN controller servers.

A number of different initiatives, both commercial and open source, have resulted in SDN controller implementations. The following list describes a few prominent ones:

- **OpenDaylight:** An open source platform for network programmability to enable SDN, written in Java. OpenDaylight was founded by Cisco and IBM, and its membership is heavily weighted toward network vendors. OpenDaylight can be implemented as a single centralized controller, but enables controllers to be distributed where one or multiple instances may run on one or more clustered servers in the network.
- **Open Network Operating System (ONOS):** An open source SDN NOS, initially released in 2014. It is a nonprofit effort funded and developed by a number of carriers, such as AT&T and NTT, and other service providers. Significantly, ONOS is supported by the Open Networking Foundation, making it likely that ONOS will be a major factor in SDN deployment. ONOS is designed to be used as a distributed controller and provides abstractions for partitioning and distributing network state onto multiple distributed controllers.
- **POX:** An open source OpenFlow controller that has been implemented by a number of SDN developers and engineers. POX has a well written API and documentation. It also provides a web-based graphical user interface (GUI) and is written in Python, which typically shortens its experimental and developmental cycles compared to some other implementation languages, such as C++.
- **Beacon:** An open source package developed at Stanford. Written in Java and highly integrated into the Eclipse integrated development environment (IDE). Beacon was the first controller that made it possible

for beginner programmers to work with and create a working SDN environment.

- **Floodlight:** An open source package developed by Big Switch Networks. Although its beginning was based on Beacon, it was built using Apache Ant, which is a very popular software build tool that makes the development of Floodlight easier and more flexible. Floodlight has an active community and has a large number of features that can be added to create a system that best meets the requirements of a specific organization. Both a web-based and Java-based GUI are available and most of its functionality is exposed through a REST API.
- **Ryu:** An open source component-based SDN framework developed by NTT Labs. It is open sourced and fully developed in python.
- **Onix:** Another distributed controller, jointly developed by VMWare, Google, and NTT. Onix is a commercially available SDN controller.

➔ *See Section 5.3, "Open-Daylight"*

Perhaps the most significant controller on this list is OpenDaylight, described subsequently in Section 5.3.

Southbound Interface

The southbound interface provides the logical connection between the SDN controller and the data plane switches (see Figure 5.3). Some controller products and configurations support only a single southbound protocol. A more flexible approach is the use of a southbound abstraction layer that provides a common interface for the control plane functions while supporting multiple southbound APIs.

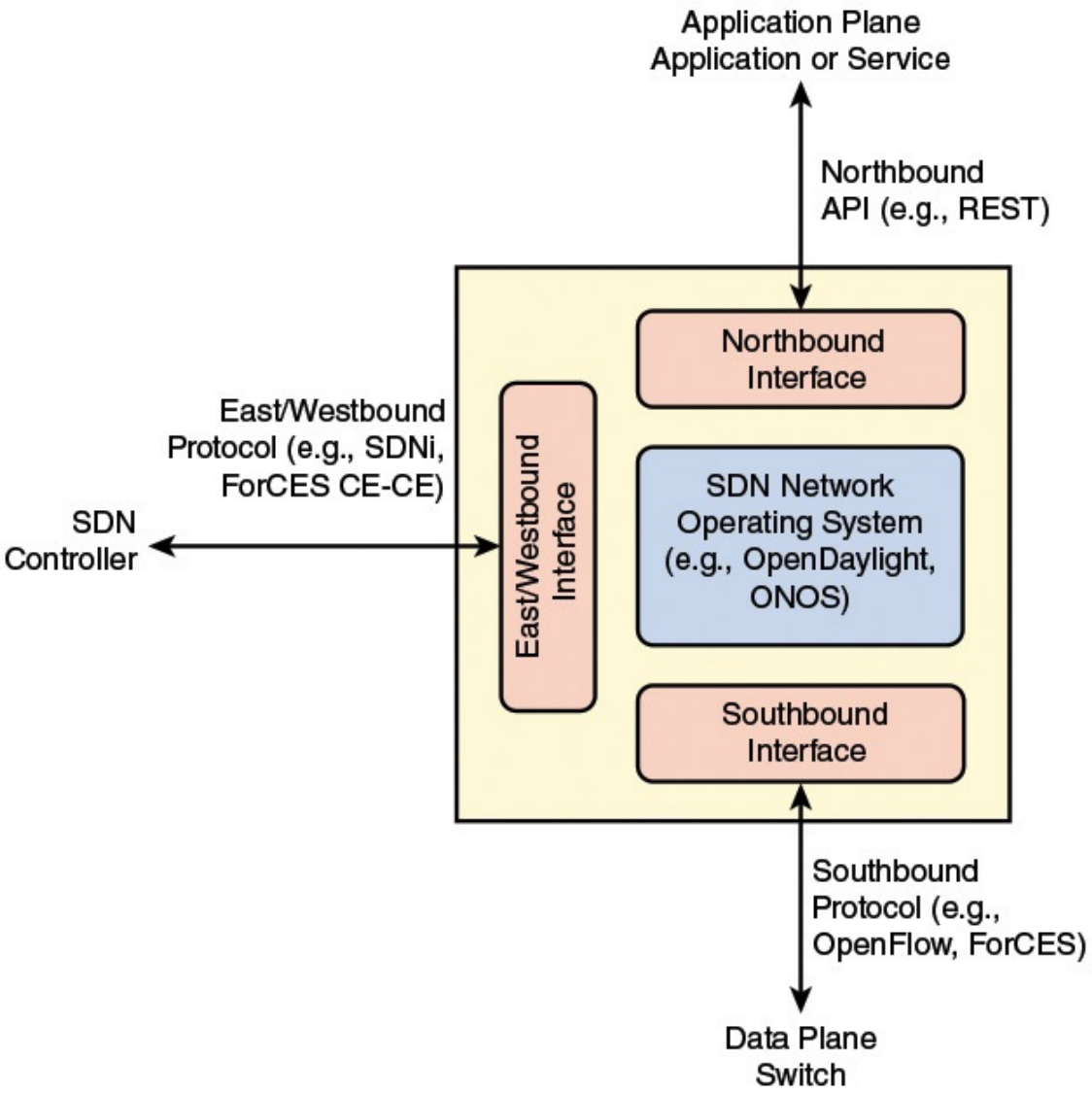

FIGURE 5.3 SDN Controller Interfaces

The most commonly implemented southbound API is OpenFlow, covered in some detail in Chapter 4, "SDN Data Plane and OpenFlow." Other southbound interfaces include the following:

- **Open vSwitch Database Management Protocol (OVSDB):** Open vSwitch (OVS) an open source software project which implements virtual switching that is interoperable with almost all popular hypervisors. OVS uses OpenFlow for message forwarding in the control plane for both virtual and physical ports. OVSDB is the protocol used to manage and configure OVS instances.

- **Forwarding and Control Element Separation (ForCES):** An IETF effort that standardizes the interface between the control plane and the data plane for IP routers.
- **Protocol Oblivious Forwarding (POF):** This is advertised as an enhancement to OpenFlow that simplifies the logic in the data plane to a very generic forwarding element that need not understand the protocol data unit (PDU) format in terms of fields at various protocol levels. Rather, matching is done by means of (offset, length) blocks within a packet. Intelligence about packet format resides at the control plane level.

Northbound Interface

The northbound interface enables applications to access control plane functions and services without needing to know the details of the underlying network switches. The northbound interface is more typically viewed as a software API rather than a protocol.

Unlike the southbound and eastbound/westbound interfaces, where a number of heterogeneous interfaces have been defined, there is no widely accepted standard for the northbound interface. The result has been that a number of unique APIs have been developed for various controllers, complicating the effort to develop SDN applications. To address this issue the Open Networking Foundation formed the Northbound Interface Working Group (NBI-WG) in 2013, with the objective of defining and standardizing a number of broadly useful northbound APIs. As of this writing, the working group has not issued any standards.

A useful insight of the NBI-WG is that even in an individual SDN controller instance, APIs are needed at different "latitudes." That is, some APIs may be "further north" than others, and access to one, several, or all of these different APIs could be a requirement for a given application.

Figure 5.4, from the NBI-WG charter document (October 2013), illustrates the concept of multiple API latitudes. For example, an application may need one or more APIs that directly expose the functionality of the controller, to manage a network domain, and use APIs that invoke analytic or reporting services residing on the controller.

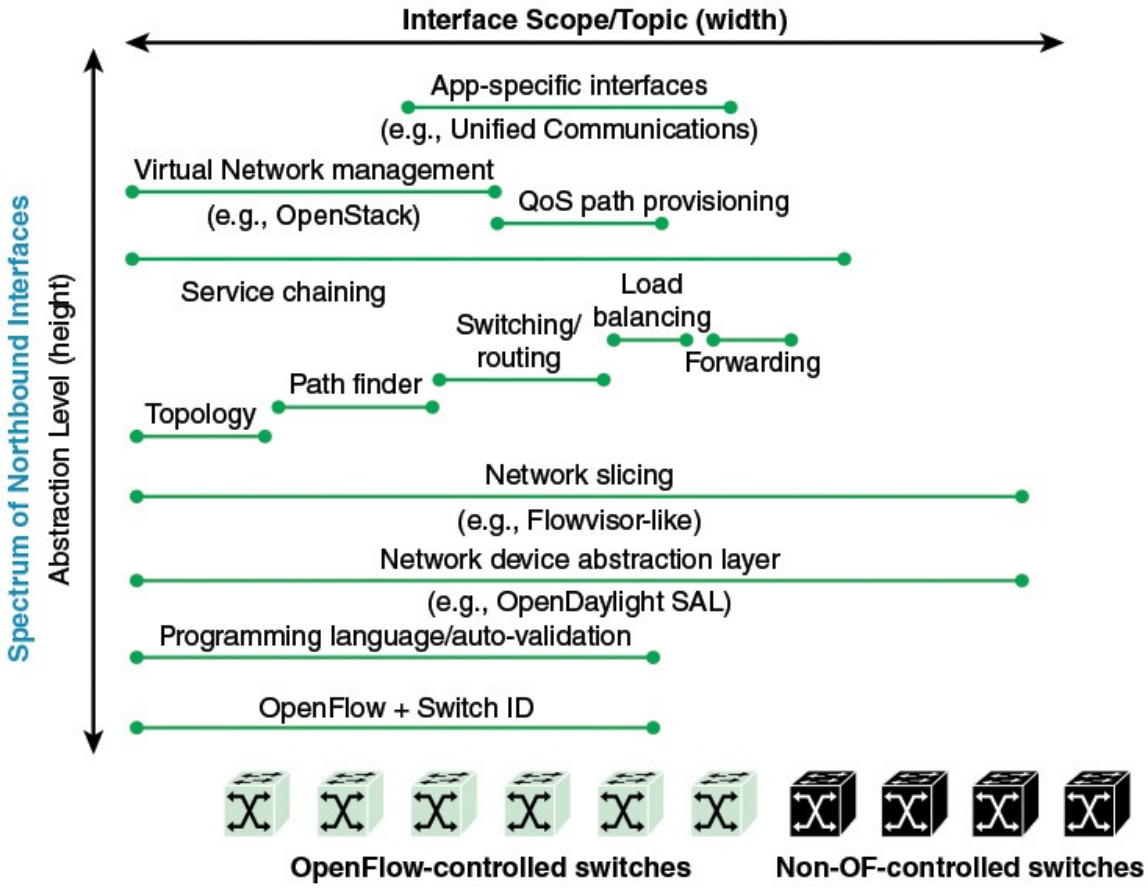

FIGURE 5.4 Latitude of Northbound Interfaces

Figure 5.5 shows a simplified example of an architecture with multiple levels of northbound APIs, the levels of which are described in the list that follows.

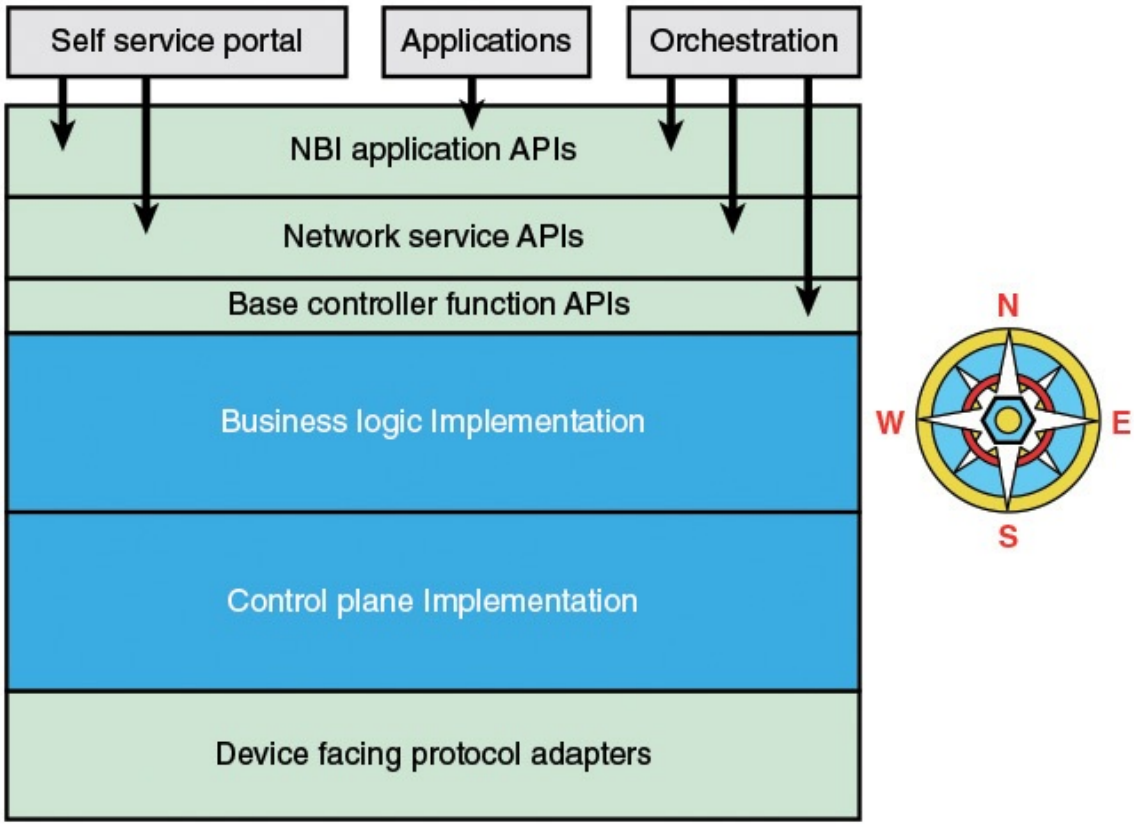

FIGURE 5.5 SDN Controller APIs

- **Base controller function APIs:** These APIs expose the basic functions of the controller and are used by developers to create network services.
- **Network service APIs:** These APIs expose network services to the north.
- **Northbound interface application APIs:** These APIs expose application-related services that are built on top of network services.

A common architectural style used for defining northbound APIs is REpresentational State Transfer (REST). Section 5.4 discusses REST.

➔ *See Section 5.4, "REST"*

Routing

As with any network or internet, an SDN network requires a routing function. In general terms, the routing function comprises a protocol for collecting information about the topology and traffic conditions of the network, and an