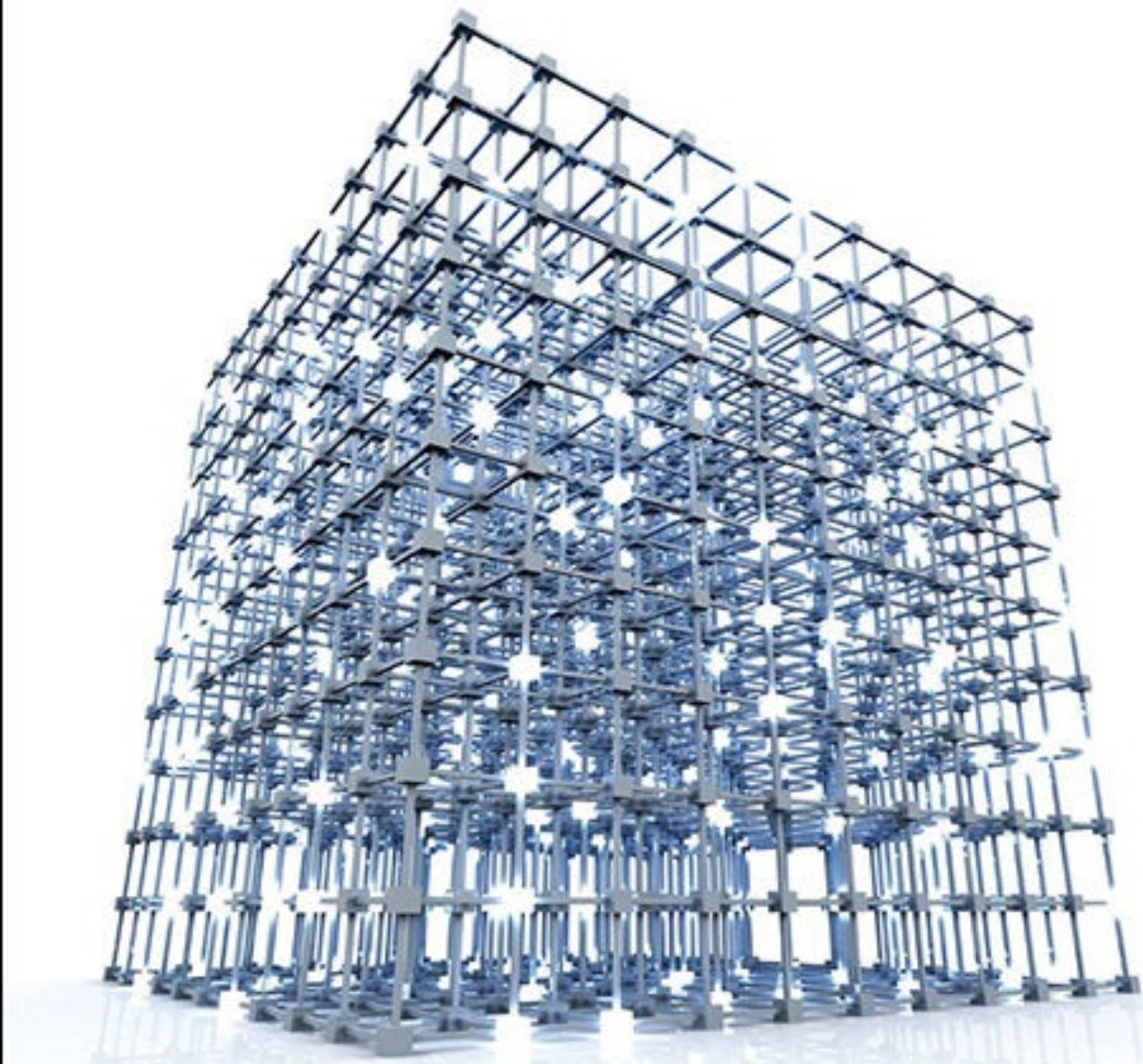
William Stallings

Foundations of Modern Networking

SDN, NFV, QoE, IoT, and Cloud



About This eBook

ePUB is an open, industry-standard format for eBooks. However, support of ePUB and its many features varies across reading devices and applications. Use your device or app settings to customize the presentation to your liking. Settings that you can customize often include font, font size, single or double column, landscape or portrait mode, and figures that you can click or tap to enlarge. For additional information about the settings and features on your reading device or app, visit the device manufacturer's Web site.

Many titles include programming code or configuration examples. To optimize the presentation of these elements, view the eBook in single-column, landscape mode and adjust the font size to the smallest setting. In addition to presenting code and configurations in the reflowable text format, we have included images of the code that mimic the presentation found in the print book; therefore, where the reflowable format may compromise the presentation of the code listing, you will see a "Click here to view code image" link. Click the link to view the print-fidelity code image. To return to the previous page viewed, click the Back button on your device or app.

Foundations of Modern Networking

SDN, NFV, QoE, IoT, and Cloud

William Stallings

With contributions by:

Florence Agboma
British Sky Broadcasting

Sofiene Jelassi
Assistant Professor
University of Monastir, Tunisia

PEARSON

800 East 96th Street, Indianapolis, Indiana 46240 USA

Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud

ISBN-13: 978-0-13-417539-3
ISBN-10: 0-13-417539-5

Library of Congress Control Number: 2015950673

Text printed in the United States on recycled paper at RR Donnelley, Crawfordsville, IN
First printing: November 2015

Associate Publisher
Dave Dusthimer

Executive Editor
Brett Bartow

Senior Development Editor
Christopher Cleveland

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Keith Cline

Indexer
Publishing Works

Proofreader
Katie Matejka

Technical Reviewers
Wendell Odom
Tim Szigeti

Editorial Assistant
Vanessa Evans

Designer
Alan Clements

Compositor
Mary Sudul

Trademarks

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Warning and Disclaimer

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Contents at a Glance

Table of Contents

About the Author

Dr. William Stallings has made a unique contribution to understanding the broad sweep of technical developments in computer security, computer networking, and computer architecture. He has authored 18 textbooks, and, counting revised editions, a total of 70 books on various aspects of these subjects. His writings have appeared in numerous ACM and IEEE publications, including the *Proceedings of the IEEE* and *ACM Computing Reviews*. He has 13 times received the award for the best computer science textbook of the year from the Text and Academic Authors Association.

In over 30 years in the field, he has been a technical contributor, technical manager, and an executive with several high-technology firms. He has designed and implemented both TCP/IP-based and OSI-based protocol suites on a variety of computers and operating systems, ranging from microcomputers to mainframes. Currently, he is an independent consultant whose clients have included computer and networking manufacturers and customers, software development firms, and leading-edge government research institutions.

He created and maintains the Computer Science Student Resource Site at ComputerScienceStudent.com/. This site provides documents and links on a variety of subjects of general interest to computer science students (and professionals). He is a member of the editorial board of *Cryptologia*, a scholarly journal devoted to all aspects of cryptology.

Dr. Stallings holds a Ph.D. from M.I.T. in Computer Science and a B.S. from Notre Dame in electrical engineering.

About the Contributing Authors

Florence Agboma currently works as a Technology Analyst at British Sky Broadcasting (BSkyB), London. Her work includes streaming video quality improvements for different video platforms such as linear OTT, VoD, and broadcast. She is a member of the Video Quality Experts Group (VQEG). Dr. Agboma holds a Ph.D. from the University of Essex, United Kingdom, and her research focused on quality of experience for mobile content delivery systems.

Dr. Agboma has published a number of peer-reviewed articles in journal papers, book chapters, and international conference proceedings. Her interests include video quality assessments, psychophysical methods, pay TV analytics, quality of experience management, and emerging broadcast TV technologies such as high dynamic range and ultra HD.

Sofiene Jelassi received a Bachelor of Science and a Master of Science from the University of Monastir, Tunisia, in June 2003 and December 2005, respectively. He obtained a Ph.D. in Computer Science from the University of Pierre and Marie Curie, Paris, France, in February 2010. His doctoral thesis was titled *Adaptive Quality Control of Packetized Voice Conversations over Mobile Ad-Hoc Networks*. From June 2010 to December 2013, he worked as an R&D engineer at Inria within DIONYSOS research group. From January to December 2014, he worked as a post-doctoral fellow at GTA/UFRJ in Rio de Janeiro, Brazil. Since January 2015, he has been working as Assistant Professor at University of Monastir, Tunisia. His research

includes wired and wireless software-defined networks (SDNs), server and network virtualization, network monitoring, content-oriented management of mobile networks and services, mobile virtual network operators (MVNO), customized voice and video systems, quality of user experience (QoE) measurement and modeling, in-lab and in-field usability testing, crowdsourcing, user profiling, context sensing, service gamification, and social-driven emergency services. Dr. Jelassi has more than 20 papers published in international journals and conferences.

Dedication

To Tricia, my loving wife, the kindest and gentlest person.

Acknowledgments

This book has benefited from review by a number of people who gave generously of their time and expertise. I especially thank Wendell Odom (Certskills, LLC) and Tim Szigeti (Cisco Systems), who each devoted an enormous amount of time to a detailed review of the entire manuscript.

Thanks also to the many people who provided detailed technical reviews of one or more chapters: Christian Adell (Corporació Catalana de Mitjans Audiovisuals), Eduard Dulharu (AT&T Germany), Cemal Duman (Ericsson), David L. Foote (NFV Forum (ATIS)), Harold Fritts, Scott Hogg (Global Technology Resources), Justin Kang (Accenture), Sergey Katsev (Fortinet), Raymond Kelly (Telecoms Now Ltd), Faisal Khan (Mobily Saudi Arabia), Epameinondas Kontothanasis (Unifys), Sashi Kumar (Intel), Hongwei Li (Hewlett-Packard), Cynthia Lopes (Maya Technologies), Simone Mangiante (EMC), Roberto Fuentes Martinez (Tecnocom), Mali Naghavi (Ericsson), Fatih Eyup Nar (Ericsson USA), Jimmy Ng (Huawei Technologies), Mark Noble (Salix Technology Services), Luke Reid (Sytel Reply UK), David Schuckman (State Farm Insurance), Vivek Srivastava (Zscaler), Istvan Teglas (Cisco Systems), and Paul Zanna (Northbound Networks).

Finally, I want to thank the many people at Pearson responsible for the publication of the book. This includes the staff at Pearson, particularly Senior Development Editor Chris Cleveland; Executive Editor Brett Bartow, and his assistant Vanessa Evans; and Project Editor Mandie Frank. Thanks also to the marketing and sales staffs at Pearson, without whose efforts this book would not be in front of you.

With all this assistance, little remains for which I can take full credit. However, I am proud to say that, with no help whatsoever, I selected all the quotations.

Preface

> There is the book, Inspector. I leave it with you, and you cannot doubt that it contains a full explanation.
>
> —*The Adventure of the Lion's Mane*, Sir Arthur Conan Doyle

Background

A host of factors have converged to produce the latest revolution in computer and communications networking:

- **Demand:** Enterprises are faced with a surge of demands that focus their attention on the need to design, evaluate, manage, and maintain sophisticated network infrastructures. These trends include the following:
- **Big data:** Enterprises large and small increasingly rely on processing and analyzing massive amounts of data. To process large quantities of data within tolerable time periods, big data may need distributed file systems, distributed databases, cloud computing platforms, Internet storage, and other scalable storage technologies.
- **Cloud computing:** There is an increasingly prominent trend in many organizations to move a substantial portion or even all information technology (IT) operations to an Internet-connected infrastructure known as enterprise cloud computing. This drastic shift in IT data processing is accompanied by an equally drastic shift in networking requirements.
- **Internet of Things (IoT):** The IoT involves large numbers of objects that use standard communications architectures to provide services to end users. Billions of such devices will be interconnected in industrial, business, and government networks, providing new interactions between the physical world and computing, digital content, analysis, applications, and services. IoT provides unprecedented opportunities for users, manufacturers, and service providers in a wide variety of sectors. Areas that will benefit from IoT data collection, analysis, and automation capabilities include health and fitness, healthcare, home monitoring and automation, energy savings and smart grid, farming, transportation, environmental monitoring, inventory and product management, security, surveillance, education, and many others.
- **Mobile devices:** Mobile devices are now an indispensable part of every enterprise IT infrastructure, including employer supplied and bring your own device (BYOD). The large population of mobile devices generates unique new demands on network planning and management.
- **Capacity:** Two interlocking trends have generated new and urgent requirements for intelligent and efficient network design and management:
- **Gigabit data rate networks:** Ethernet offerings have reached 100 Gbps with further

increases in the works. Wi-Fi products at almost 7 Gbps are available. And 4G and 5G networks bring gigabit speeds to cellular networks.

- **High-speed, high-capacity servers:** Massive blade servers and other high-performance servers have evolved to meet the increasing multimedia and data processing requirements of enterprises, calling for a need for efficiently designed and managed networks.
- **Complexity:** Network designers and managers operate in a complex, dynamic environment, in which a range of requirements, most especially quality of service (QoS) and quality of experience (QoE) require flexible, manageable networking hardware and services.
- **Security:** With increasing reliance on networked resources, an increasing need emerges for networks that provide a range of security services.

With the development of new network technologies in response to these factors, it is imperative for system engineers, system analysts, IT managers, network designers, and product marketing specialists to have a firm grasp on modern networking. These professionals need to understand the implications of the factors listed above and how network designers have responded. Dominating this landscape are (1) two complementary technologies that are rapidly being developed and deployed (software-defined networking [SDN] and network functions virtualization [NFV]) and (2) the need to satisfy QoS and QoE requirements.

This book provides the reader with a thorough understanding of SDN and NFV and their practical deployment and use in today's enterprises. In addition, the book provides clear explanations of QoS/QoE and the whole range of related issues, such as cloud networking and IoT. This is a technical book, intended for readers with some technical background, but is sufficiently self-contained to be a valuable resource for IT managers and product marketing personnel, in addition to system engineers, network maintenance personnel, and network and protocol designers.

Organization of the Book

The book consists of six parts:

- **Modern Networking:** Provides an overview of modern networking and a context for the remainder of the book. Chapter 1 is a survey of the elements that make up the networking ecosystem, including network technologies, network architecture, services, and applications. Chapter 2 examines the requirements that have evolved for the current networking environment and provides a preview of key technologies for modern networking.
- **Software-Defined Networks:** Devoted to a broad and thorough presentation of SDN concepts, technology, and applications. Chapter 3 begins the discussion by laying out what the SDN approach is and why it is needed, and provides an overview of the SDN architecture. This chapter also looks at the organizations that are issuing specifications and standards for SDN. Chapter 4 is a detailed look at the SDN data plane, including the key components, how they interact, and how they are managed. Much of the chapter is devoted to OpenFlow, a vital data plane technology and an interface to the control plane. The chapter explains why OpenFlow is needed and then proceeds to provide a detailed

technical explanation. Chapter 5 is devoted to the SDN control plane. It includes a discussion of OpenDaylight, an important open source implementation of the control plane. Chapter 6 covers the SDN application plane. In addition to examining the general SDN application plane architecture, the chapter discusses six major application areas that can be supported by SDN and provides a number of examples of SDN applications.

- **Virtualization:** Devoted to a broad and thorough presentation of network functions virtualization (NFV) concepts, technology, and applications, as well as a discussion of network virtualization. Chapter 7 introduces the concept of virtual machine, and then looks at the use of virtual machine technology to develop NFV-based networking environments. Chapter 8 provides a detailed discussion of NFV functionality. Chapter 9 looks at traditional concepts of virtual networks, then at the more modern approach to network virtualization, and finally introduces the concept of software defined infrastructure.
- **Defining and Supporting User Needs:** Equally as significant as the emergence of the SDN and NFV is the evolution of quality of service (QoS) and quality of experience (QoE) to determine customer needs and network design responses to those needs. Chapter 10 provides an overview of QoS concepts and standards. Recently QoS has been augmented with the concept of QoE, which is particularly relevant to interactive video and multimedia network traffic. Chapter 11 provides an overview of QoE and discusses a number of practical aspects of implementing QoE mechanisms. Chapter 12 looks further into the network design implications of the combined use of QoS and QoE.
- **Modern Network Architecture: Clouds and Fog:** The two dominant modern network architectures are cloud computing and the Internet of things (IoT), sometimes referred to as fog computing. The technologies and applications discussed in the preceding parts all provide a foundation for cloud computing and IoT. Chapter 13 is a survey of cloud computing. The chapter begins with a definition of basic concepts, and then covers cloud services, deployment models, and architecture. The chapter then discusses the relationship between cloud computing and SDN and NFV. Chapter 14 introduces IoT and provides a detailed look at the key components of IoT-enabled devices. Chapter 15 looks at several model IoT architectures and then describes three example IoT implementations.
- **Related Topics:** Discusses two additional topics that, although important, do not conveniently fit into the other Parts. Chapter 16 provides an analysis of security issues that have emerged with the evolution of modern networking. Separate sections deal with SDN, NFV, cloud, and IoT security, respectively. Chapter 17 discusses career-related issues, including the changing role of various network-related jobs, new skill requirements, and how the reader can continue his or her education to prepare for a career in modern networking.

Supporting Websites

I maintain a companion website at WilliamStallings.com/Network that includes a list of relevant links organized by chapter and an errata sheet for the book.

Companion website

I also maintain the Computer Science Student Resource Site, at ComputerScienceStudent.com. The purpose of this site is to provide documents, information, and links for computer science students and professionals. Links and documents are organized into seven categories:

Computer Science Student Resource Site

- **Math:** Includes a basic math refresher, a queuing analysis primer, a number system primer, and links to numerous math sites.
- **How-to:** Advice and guidance for solving homework problems, writing technical reports, and preparing technical presentations.
- **Research resources:** Links to important collections of papers, technical reports, and bibliographies.
- **Other useful:** A variety of other useful documents and links.
- **Computer science careers:** Useful links and documents for those considering a career in computer science.
- **Writing help:** Help in becoming a clearer, more effective writer.
- **Miscellaneous topics and humor:** You have to take your mind off your work once in a while.

Part I: Modern Networking

The whole of this operation is described in minute detail in the official British Naval History, and should be studied with its excellent charts by those who are interested in its technical aspect. So complicated is the full story that the lay reader cannot see the wood for the trees. I have endeavored to render intelligible the broad effects.

—*The World Crisis*, Winston Churchill

Part I provides an overview of modern networking and a context for the remainder of the book. Chapter 1 is a survey of the elements that make up the networking ecosystem, including network technologies, network architecture, services, and applications. In Chapter 2, we examine the requirements that have evolved for the current networking environment and provide a preview of key technologies for modern networking.

Chapter 1. Elements of Modern Networking

> There is some evidence that computer networks will have a large impact on society. Likely areas are the economy, resources, small computers, human-to-human interaction, and computer research.
>
> —*What Can Be Automated?*
> The Computer Science and Engineering Research Study, National Science Foundation, 1980

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

- Explain the key elements and their relationships of a modern networking ecosystem, including end users, network providers, application providers and application service providers.
- Discuss the motivation for the typical network hierarchy of access networks, distribution networks, and core networks.
- Present an overview of Ethernet, including a discussion of its application areas and common data rates.
- Present an overview of Wi-Fi, including a discussion of its application areas and common data rates.
- Understand the differences between the five generations of cellular networks.
- Present an overview of cloud computing concepts.
- Describe the Internet of Things.
- Explain the concepts of network convergence and unified communications.

Long gone are the days when a single vendor, such as IBM, could provide an enterprise with all the products and services required by their information technology (IT) department, including computer hardware, system software, applications software, and communications and networking equipment and services. Today, users and enterprises face complex, heterogeneous and diverse environments that require sophisticated and advanced solutions.

The focus of this book is twofold:

- The networking technologies that enable the design, development, deployment, and operation of complex modern networks, including and especially software-defined networks (SDN), network functions virtualization (NFV), quality of service (QoS), and quality of experience (QoE).
- The network architectures that have come to dominate modern networking, which are cloud networking and the Internet of Things (IoT), also known as fog networking.

But before diving into the details of these technologies, we need an overview of the current networking environment and the challenges it brings.

This chapter provides a brief survey of the key elements of modern networking. We begin with a top-level description of what might be considered the typical networking ecosystem. Then, Section 1.2 looks in more detail at the way in which the network elements are organized. Next, Sections 1.3 through 1.5 examine the key high-speed network technologies that support the modern networking ecosystem. The remainder of this chapter introduces important architectures and applications that are part of this ecosystem.

1.1 The Networking Ecosystem

Figure 1.1 depicts the modern networking ecosystem in very general terms. The entire ecosystem exists to provide services to end users. The term **end user**, or simply *user*, is used here as a very general term, to encompass users working within an enterprise or in a public setting or at home. The user platform can be fixed (for example, PC or workstation), portable (for example, laptop), or mobile (for example, tablet or smartphone).

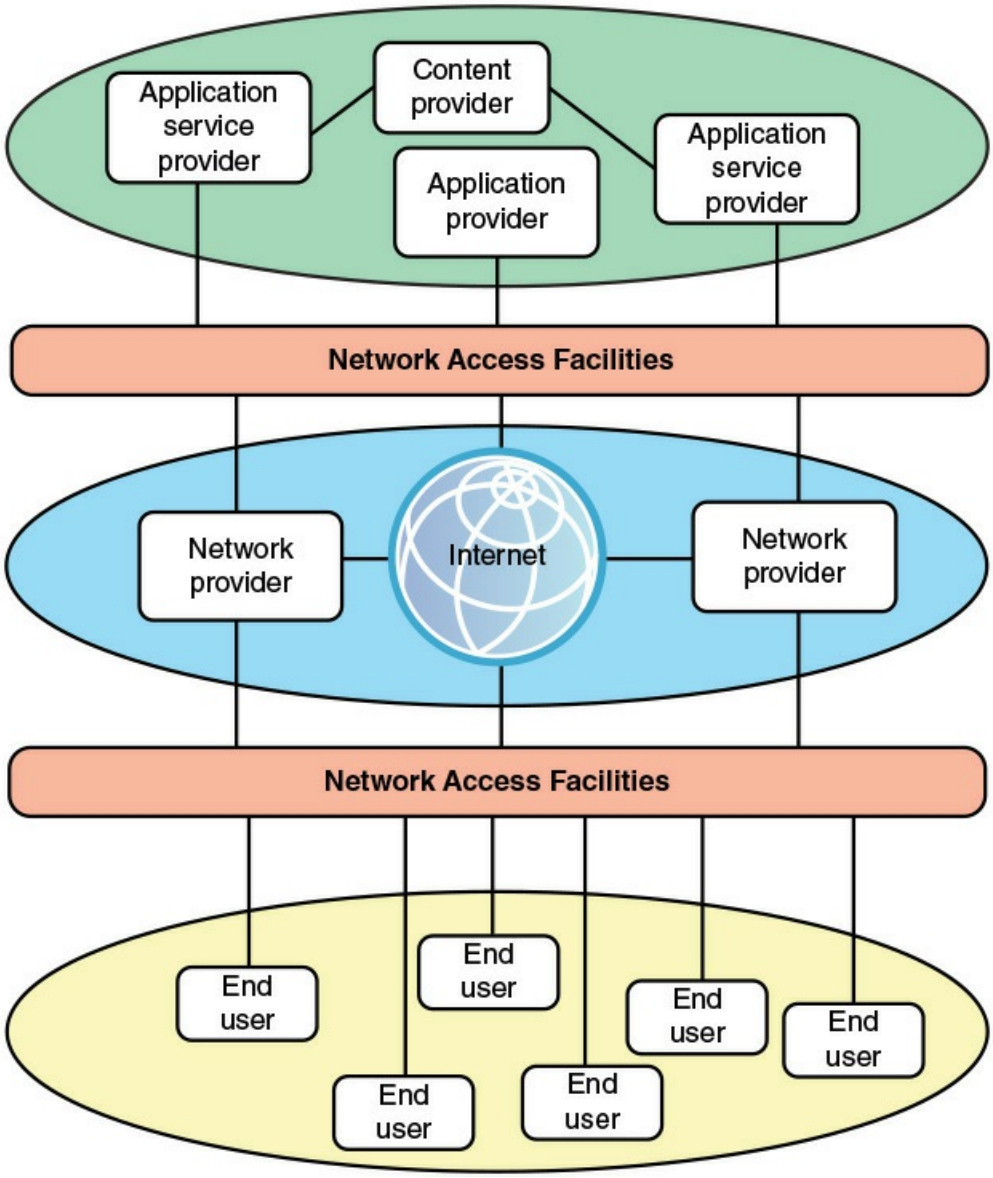


FIGURE 1.1 The Modern Networking Ecosystem

Users connect to network-based services and content through a wide variety of network access facilities. These include digital subscriber line (DSL) and cable modems, Wi-Fi and Worldwide Interoperability for Microwave Access (WiMAX) wireless modems, and cellular modems. Such network access facilities enable the use to connect directly to the Internet or to a variety of network providers, including Wi-Fi networks, cellular networks, and both private and shared network facilities, such as a premises enterprise network.

Ultimately, of course, users want to use network facilities to access applications and content. Figure 1.1 indicates three broad categories of interest to users. **Application providers** provide applications, or apps, that run on the user's platform, which is typically a mobile platform. More recently, the concept of an app store has become available for fixed and portable platforms as well.

A distinct category of provider is the **application service provider**. Whereas the application provider downloads software to the user's platform, the application service provider acts as a server or host of application software that is executed on the provider's platforms. Traditional examples of such software include web servers, e-mail servers, and database servers. The most prominent example now is the cloud computing provider. We discuss this latter category subsequently in this chapter and in Chapter 13, "Cloud Computing."

The final element shown in Figure 1.1 is the **content provider**. A content provider serves the data to be consumed on the user device (for example, e-mail, music, video). This data may be commercially provided intellectual property. In some instances, an enterprise may be an application or content provider. Examples of content providers are music record labels and movie studios.

Figure 1.1 is intended to provide a very general depiction of the networking ecosystem. It is worth pointing out here two major elements of modern networking not explicitly depicted in this figure:

- **Data center networking:** Both large enterprise data centers and cloud provider data centers consist of very large numbers of interconnected servers. Typically, as much as 80 percent of the data traffic is within the data center network, and only 20 percent relies on external networks to reach users.
- **IoT or fog networking:** An Internet of Things deployed by an enterprise may consist of hundreds, thousands, even millions of devices. The vast bulk of the data traffic to and from these devices is machine to machine, rather than user to machine.

Each of these networking environments creates its own particular requirements, which are discussed as the book progresses.

1.2 Example Network Architectures

This section introduces two example network architectures, and with them some of the networking terminology in common use. These examples give some idea of the range of network architectures covered in this book.

A Global Network Architecture

We begin with an architecture that could represent an enterprise network of national or global extent, or a portion of the Internet with some of its associated networks. Figure 1.2 illustrates some of the typical communications and network elements in use in such a context.

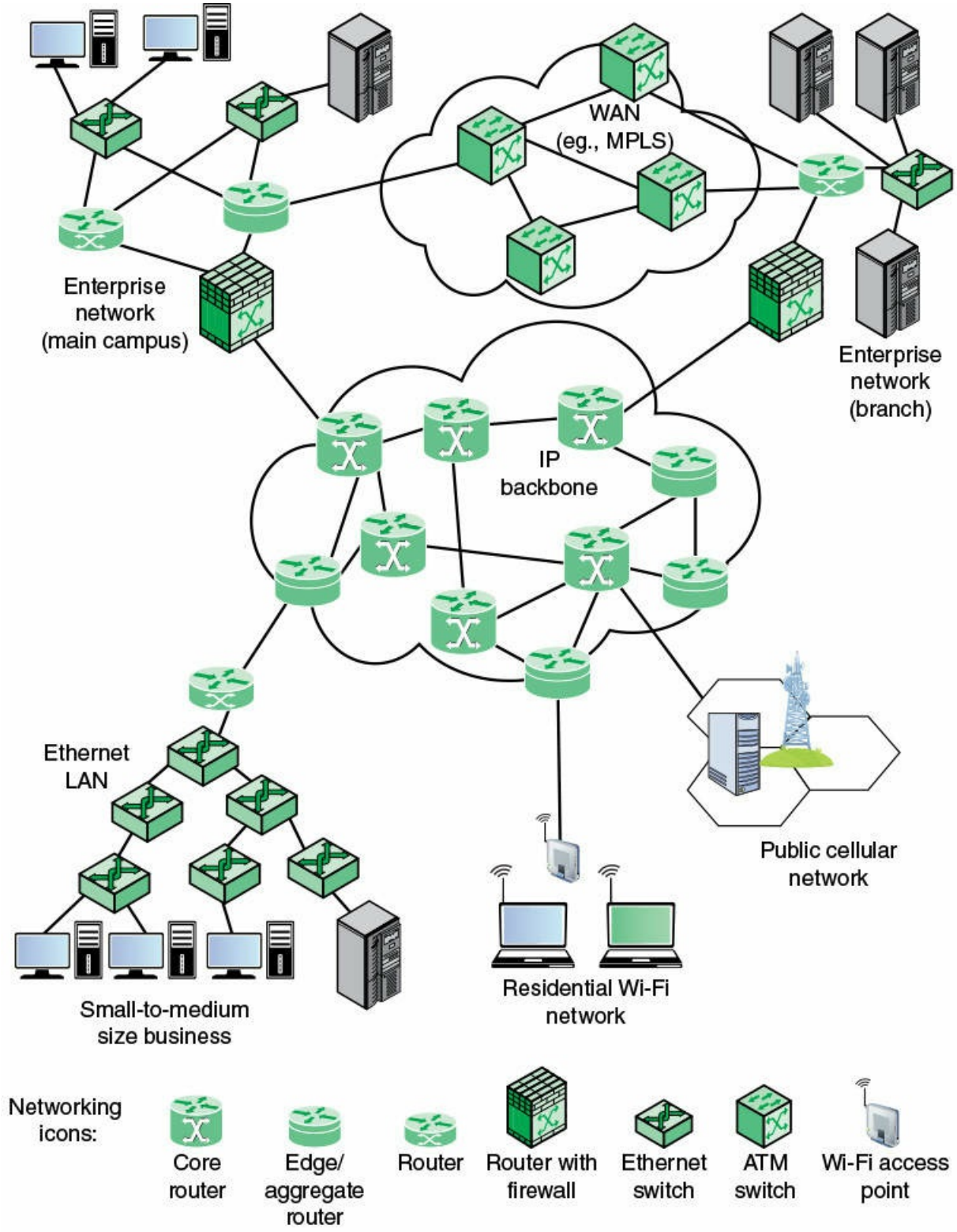


FIGURE 1.2 A Global Networking Architecture

At the center of the figure is an IP backbone, or core, network, which could represent a portion of the Internet or an enterprise IP network. Typically, the backbone consists of high-performance

routers, called **core routers**, interconnected with high-volume optical links. The optical links often make use of what is known as wavelength-division multiplexing (WDM), such that each link has multiple logical channels occupying different portions of the optical bandwidth.

At the periphery of an IP backbone are routers that provide connectivity to external networks and users. These routers are sometimes referred to as **edge routers** or **aggregation routers**. Aggregation routers are also used within an enterprise network to connect a number of routers and switches, to external resources, such as an IP backbone or a high-speed WAN. As an indication of the capacity requirements for core and aggregation routers, the IEEE Ethernet Bandwidth Assessments Group [XI11] reports on an analysis that projects these requirements for Internet backbone providers and large enterprise networks in China. The analysis concludes that aggregation router requirements will be in the range of 200 Gbps to 400 Gbps per optical link by 2020, and 400 Gbps to 1 Tbps per optical link for core routers by 2020.

The upper part of Figure 1.2 depicts a portion of what might be a large enterprise network. The figure shows two sections of the network connected via a private high-speed WAN, with switches interconnected with optical links. MPLS using IP is a common switching protocol used for such WANs; wide-area Ethernet is another option. Enterprise assets are connected to, and protected from, an IP backbone or the Internet via routers with firewall capability, a not uncommon arrangement for implementing the firewall.

The lower left of the figure depicts what might be a layout for a small- or medium-size business, which relies on an Ethernet LAN. Connection to the Internet through a router could be through a cable or DSL connection or a dedicated high-speed link.

The lower portion of Figure 1.2 also shows an individual residential user connected to an Internet service provider (ISP) through some sort of subscriber connection. Common examples of such a connection are a DSL, which provides a high-speed link over telephone lines and requires a special DSL modem, and a cable TV facility, which requires a cable modem, or some type of wireless connection. In each case, there are separate issues concerning signal encoding, error control, and the internal structure of the subscriber network.

Finally, mobile devices, such as smartphones and tablets, can connect to the Internet through the public cellular network, which has a high-speed connection, typically optical, to the Internet.

A Typical Network Hierarchy

This section focuses in on a network architecture that, with some variation, is common in many enterprises. As Figure 1.3 illustrates, enterprises often design their network facilities in a three-tier hierarchy: access, distribution, and core.

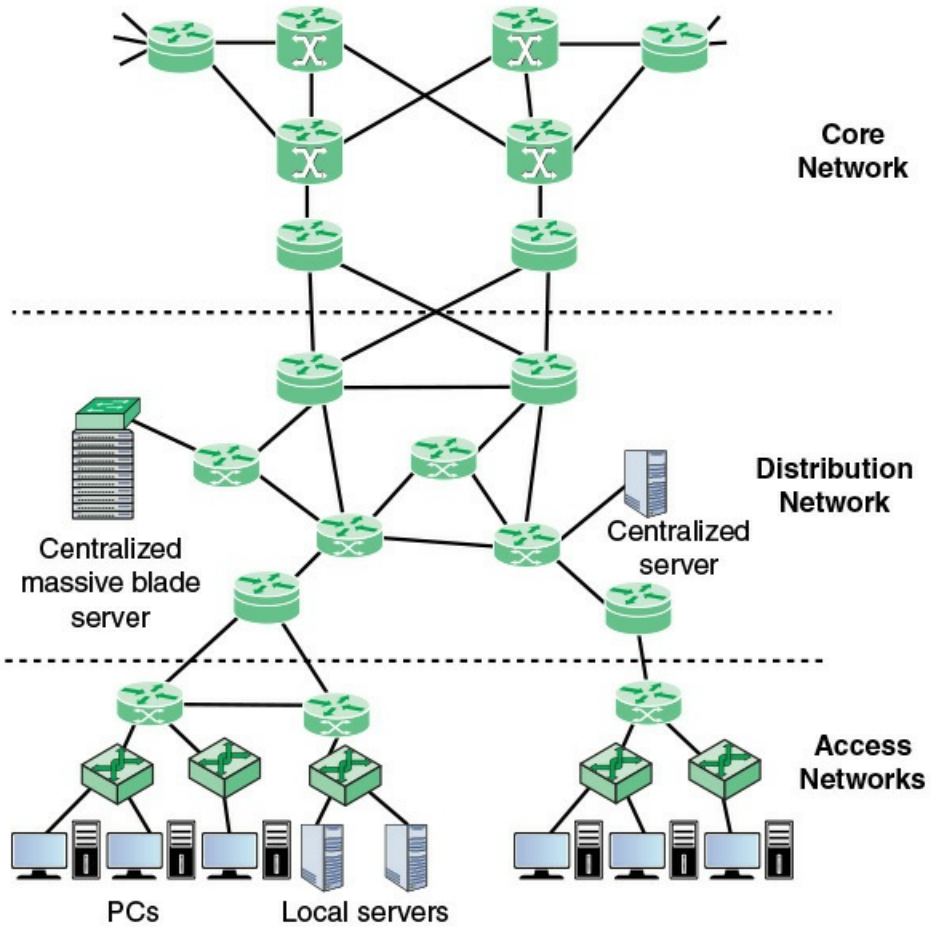


FIGURE 1.3 A Typical Network Hierarchy

Closest to the end user is the **access network**. Typically, an access network is a local-area network (LAN) or campus-wide network that consisting of LAN switches (typically Ethernet switches) and, in larger LANs, IP routers that provide connectivity among the switches. **Layer 3 switches** (not shown) are also commonly used within an LAN. The access network supports end user equipment, such as desktop and laptop computers and mobile devices. The access network also supports local servers that primarily or exclusively serve the users on the local access network.

One or more access routers connect the local assets to the next higher level of the hierarchy, the distribution network. This connection may be via the Internet or some other public or private communications facility. Thus, as described in the preceding subsection, these access routers function as edge routers that forward traffic into and out of the access network. For a large local facility, there might be additional access routers that provide internal routing but do not function as edge routers (not shown in Figure 1.2).

The **distribution network** connects access networks with each other and with the core network. An edge router in the distribution network connects to an edge router in an access network to provide connectivity. The two routers are configured to recognize each other and will generally exchange routing and connectivity information and, typically, some traffic-related information. This cooperation between routers is referred to as **peering**. The distribution network also serves

to aggregate traffic destined for the core router, which protects the core from high-density peering. That is, the use of a distribution network limits the number of routers that establish peer relationships with edge routers in the core, saving memory, processing, and transmission capacity. A distribution network may also directly connect servers that are of use to multiple access networks, such as database servers and network management servers.

Again, as with access networks, some of the distribution routers may be purely internal and do not provide an edge router function.

The **core network**, also referred to as a **backbone network**, connects geographically dispersed distribution networks as well as providing access to other networks that are not part of the enterprise network. Typically, the core network will use very high performance routers, high-capacity transmission lines, and multiple interconnected routers for increased redundancy and capacity. The core network may also connect to high-performance, high-capacity servers, such as large database servers and private cloud facilities. Some of the core routers may be purely internal, providing redundancy and additional capacity without serving as edge routers.

A hierarchical network architecture is an example of a good modular design. With this design, the capacity, features, and functionality of network equipment (routers, switches, network management servers) can be optimized for their position in the hierarchy and the requirements at a given hierarchical level.

1.3 Ethernet

Continuing the top-down approach of the preceding two sections, the next three sections focus on key network transmission technologies of **Ethernet**, Wi-Fi, and 4G/5G cellular networks. Each of these technologies has evolved to support very high data rates. These data rates support the many multimedia applications required by enterprises and consumers and, at the same time, place great demands on network switching equipment and network management facilities. A full discussion of these network technologies is beyond the scope of this book. Here, we provide a brief survey.

This section begins with discussion of Ethernet applications, and then looks at standards and performance.

Applications of Ethernet

Ethernet is the predominant wired networking technology, used in homes, offices, data centers, enterprises, and WANs. As Ethernet has evolved to support data rates up to 100 Gbps and distances from a few meters to tens of kilometers, it has become essential for supporting personal computers, workstations, servers, and massive data storage devices in organizations large and small.

Ethernet in the Home

Ethernet has long been used in the home to create a local network of computers with access to the Internet via a broadband modem/router. With the increasing availability of high-speed, low-

cost Wi-Fi on computers, tablets, smartphones, modem/routers, and other devices, home reliance on Ethernet has declined. Nevertheless almost all home networking setups include some use of Ethernet.

Two recent extensions of Ethernet technology have enhanced and broadened the use of Ethernet in the home: powerline carrier (PLC) and Power over Ethernet (PoE). Powerline modems take advantage of existing power lines and use the power wire as a communication channel to transmit Ethernet packets on top of the power signal. This makes it easy to include Ethernet-capable devices throughout the home into the Ethernet network. PoE acts in a complementary fashion, distributing power over the Ethernet data cable. PoE uses the existing Ethernet cables to distribute power to devices on the network, thus simplifying the wiring for devices such as computers and televisions.

With all of these Ethernet options, Ethernet will retain a strong presence in home networking, complementing the advantages of Wi-Fi.

Ethernet in the Office

Ethernet has also long been the dominant network technology for wired local-area networks (LANs) in the office environment. Early on there were some competitors, such as IBM's Token Ring LAN and the Fiber Distributed Data Interface (FDDI), but the simplicity, performance, and wide availability of Ethernet hardware eventually made Ethernet the winner. Today, as with home networks, the wired Ethernet technology exists side by side with the wireless Wi-Fi technology. Much of the traffic in a typical office environment now travels on Wi-Fi, particularly to support mobile devices. Ethernet retains its popularity because it can support many devices at high speeds, is not subject to interference, and provides a security advantage because it is resistant to eavesdropping. Therefore, a combination of Ethernet and Wi-Fi is the most common architecture.

Figure 1.4 provides a simplified example of an enterprise LAN architecture. The LAN connects to the Internet/WANs via a firewall. A hierarchical arrangement of routers and switches provides the interconnection of servers, fixed user devices, and wireless devices. Typically, wireless devices are only attached at the edge or bottom of the hierarchical architecture; the rest of the campus infrastructure is all Ethernet. There may also be an IP telephony server that provides call control functions (voice switching) for the telephony operations in an enterprise network, with connectivity to the public switched telephone network (PTSN).

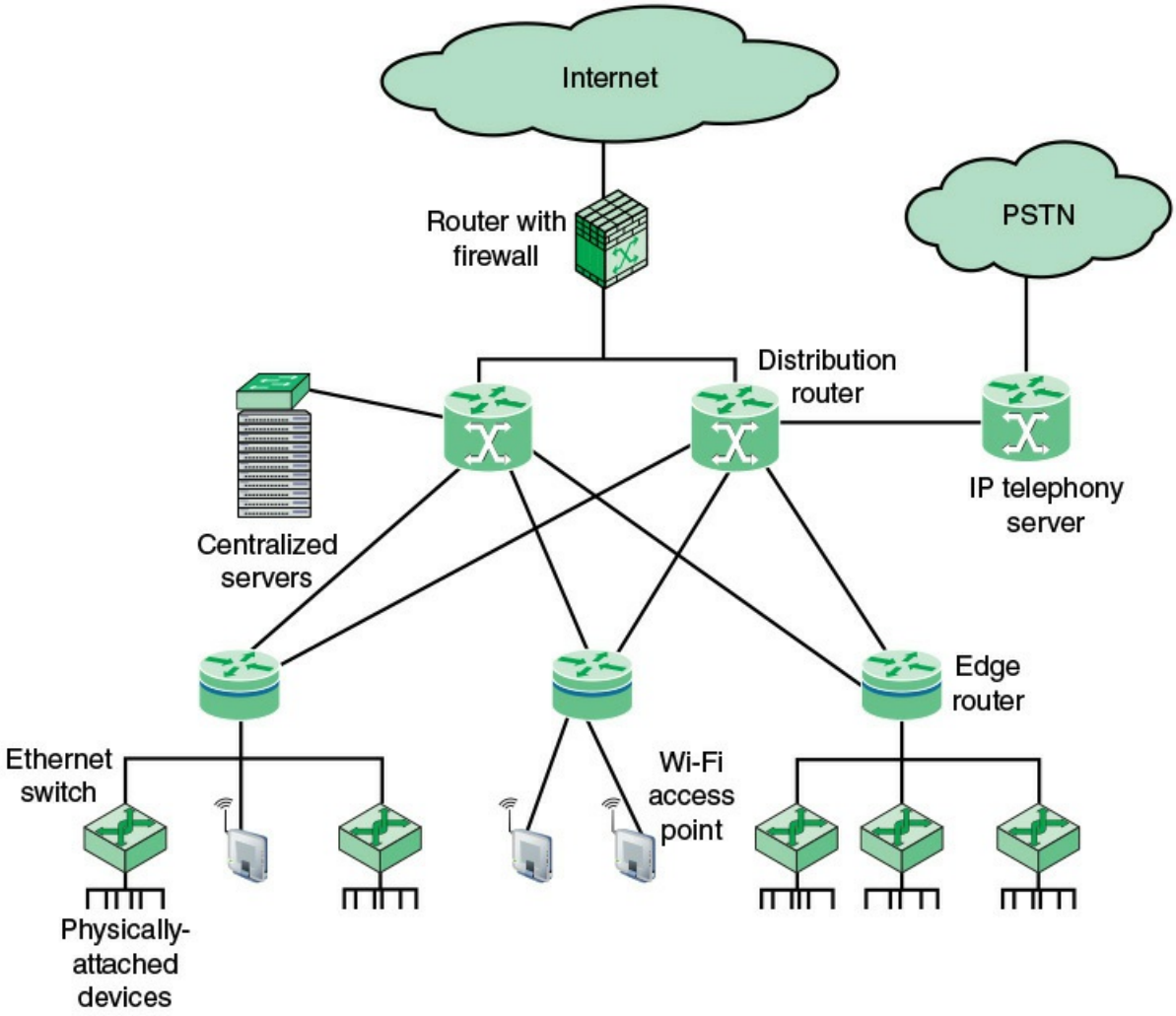


FIGURE 1.4 A Basic Enterprise LAN Architecture

Ethernet in the Enterprise

A tremendous advantage of Ethernet is that it is possible to scale the network, both in terms of distance and data rate, with the same Ethernet protocol and associated quality of service (QoS) and security standards. An enterprise can easily extend an Ethernet network among a number of buildings on the same campus or even some distance apart, with links ranging from 10 Mbps to 100 Gbps, using a mixture of cable types and Ethernet hardware. Because all the hardware and communications software conform to the same standard, it is easy to mix different speeds and different vendor equipment. The same protocol is used for intensive high-speed interconnections of data servers in a single room, workstations and servers distributed throughout the building, and links to Ethernet networks in other buildings up to 100 km away.

Ethernet in the Data Center

As in other areas, Ethernet has come to dominate in the data center, where very high data rates

are needed to handle massive volumes of data among networked servers and storage units. Historically, data centers have employed various technologies to support high-volume, short-distance needs, including InfiniBand and Fiber Channel. But now that Ethernet can scale up to 100 Gbps, with 400 Gbps on the horizon, the case for a unified protocol approach throughout the enterprise is compelling.

Two features of the new Ethernet approach are noteworthy. For co-located servers and storage units, high-speed Ethernet fiber links and switches provided the needed networking infrastructure. Another important version of Ethernet is known as backplane Ethernet. Backplane Ethernet runs over copper jumper cables that can provide up to 100 Gbps over very short distances. This technology is ideal for **blade servers**, in which multiple server modules are housed in a single chassis.

Ethernet for Wide-Area Networking

Until fairly recently, Ethernet was not a significant factor in wide-area networking. But gradually, more telecommunications and network providers have switched to Ethernet from alternative schemes to support wide-area access (also referred to as first mile or last mile). Ethernet is supplanting a variety of other wide-area options, such as dedicated T1 lines, synchronous digital hierarchy (SDH) lines, and Asynchronous Transfer Mode (ATM). When used in this fashion, the term *carrier Ethernet* is applied. The term *metro Ethernet*, or *metropolitan-area network (MAN) Ethernet*, is also used. Ethernet has the advantage that it seamlessly fits into the enterprise network for which it provides wide-area access. But a more important advantage is that carrier Ethernet provides much more flexibility in terms of the data rate capacity that is used, compared to traditional wide-area alternatives.

Carrier Ethernet is one of the fastest-growing Ethernet technologies, destined to become the dominant means by which enterprises access wide-area networking and Internet facilities.

Standards

Within the **IEEE 802** LAN standards committee, the 802.3 group is responsible for issuing standards for LANs that are referred to commercially as Ethernet. Complementary to the efforts of the 802.3 committee, the industry consortium known as The Ethernet Alliance supports and originates activities that span from incubation of new Ethernet technologies to interoperability testing to demonstrations to education.

IEEE 802.3 Committee

Ethernet Data Rates

Currently, Ethernet systems are available at speeds up to 100 Gbps. Here's a brief chronology.

- **1983:** 10 Mbps (megabit per second, million bits per second)
- **1995:** 100 Mbps
- **1998:** 1 Gbps (gigabits per second, billion bits per second)
- **2003:** 10 Gbps
- **2010:** 40 Gbps and 100 Gbps

The Ethernet Alliance

Coming soon (as of this writing) are standards at 2.5, 5, 25, 50, and 400 Gbps (see Figure 1.5).

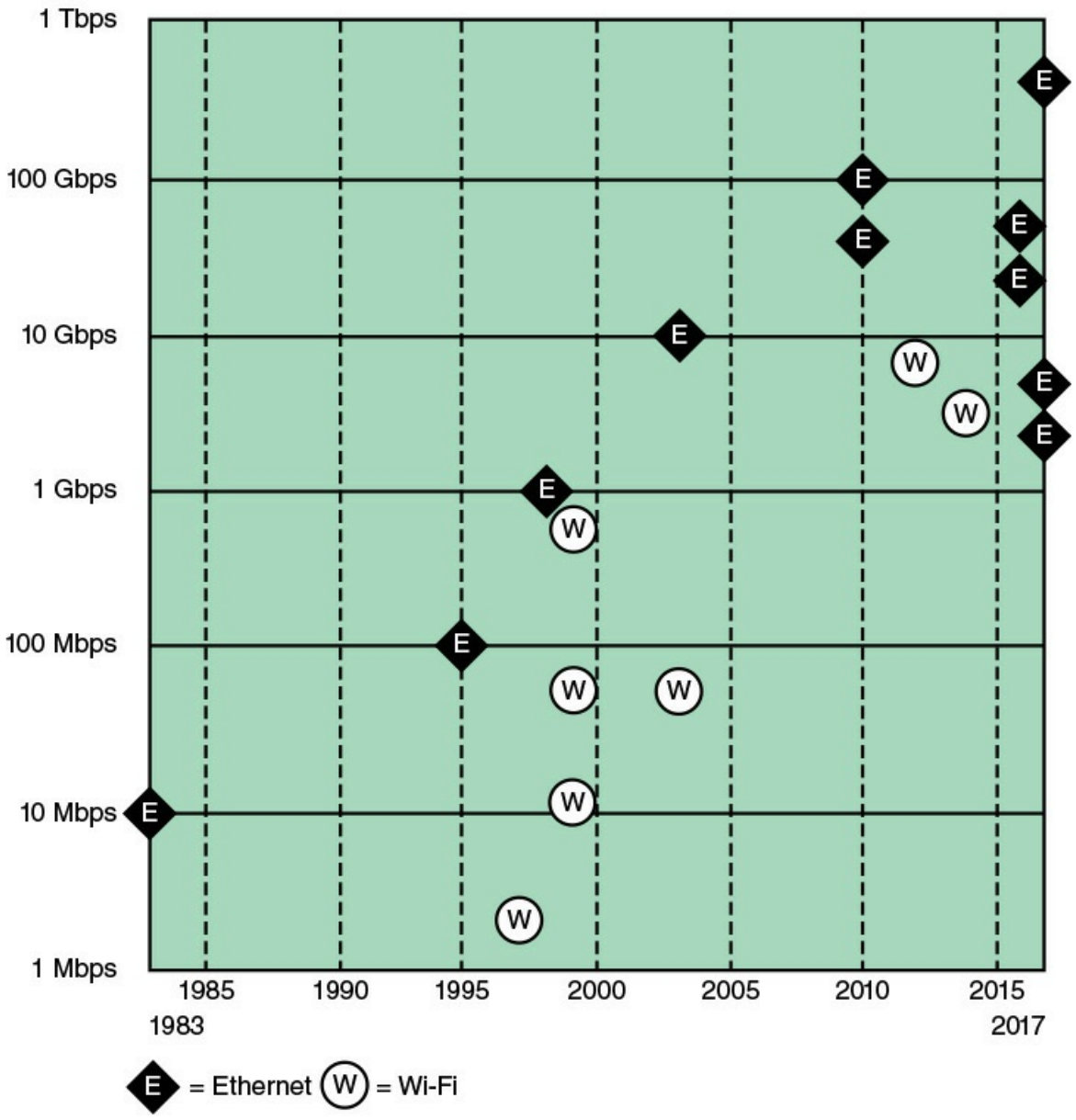


FIGURE 1.5 Ethernet and Wi-Fi Timelines

1-Gbps Ethernet

For a number of years, the initial standard of Ethernet, at 10 Mbps, was adequate for most office environments. By the early 1990s, it was clear that higher data rates were needed to support the growing traffic load on the typical LAN. Key drivers included the following:

- **Centralized server farms:** In many multimedia applications, there is a need for client system to be able to draw huge amounts of data from multiple, centralized servers, called server farms. As the performance of the servers has increased, the network becomes the bottleneck.

- **Power workgroups:** These groups typically consist of a small number of cooperating users who need to exchange massive data files across the network. Example applications are software development and computer-aided design.
- **High-speed local backbone:** As processing demand grows, enterprises develop an architecture of multiple LANs interconnected with a high-speed backbone network.

To meet such needs, the IEEE 802.3 committee developed a set of specifications for Ethernet at 100 Mbps, followed a few years later by a 1-Gbps family of standards. In each case, the new specifications defined transmission media and transmission encoding schemes built on the basic Ethernet framework, making the transition easier than if a completely new specification were issued.

10-Gbps Ethernet

Even as the ink was drying on the 1-Gbps specification, the continuing increase in local traffic made this specification inadequate for needs in the short-term future. Accordingly, the IEEE 802.3 committee soon issued a standard for 10-Gbps Ethernet. The principle driving requirement for 10-Gbps Ethernet was the increase in intranet (local interconnected networks) and Internet traffic. A number of factors contribute to the explosive growth in both Internet and intranet traffic:

- An increase in the number of network connections
- An increase in the connection speed of each end-station (for example, 10-Mbps users moving to 100 Mbps, analog 56k users moving to DSL and cable modems)
- An increase in the deployment of bandwidth-intensive applications such as high-quality video
- An increase in web hosting and application hosting traffic

Initially, network managers used 10-Gbps Ethernet to provide high-speed, local backbone interconnection between large-capacity switches. As the demand for bandwidth increased, 10-Gbps Ethernet began to be deployed throughout the entire network, to include server farm, backbone, and campus-wide connectivity. This technology enables ISPs and network service providers (NSPs) to create very high-speed links at a very low cost between co-located carrier-class switches and routers.

The technology also allows the construction of MANs and WANs that connect geographically dispersed LANs between campuses or points of presence (PoPs).

100-Gbps Ethernet

The IEEE 802.3 committee soon realized the need for a greater data rate capacity than 10-Gbps Ethernet offers, to support Internet exchanges, high-performance computing, and video-on-demand delivery. The authorization request justified the need for two different data rates in the new standard (40 Gbps and 100 Gbps) by recognizing that aggregate network requirements and end-station requirements are increasing at different rates.

The following are market drivers for 100-Gbps Ethernet:

- **Data center/Internet media providers:** To support the growth of Internet multimedia content and web applications, content providers have been expanding data centers, pushing 10-Gbps Ethernet to its limits. Likely to be high-volume early adopters of 100-Gbps Ethernet.
- **Metro video/service providers:** Video on demand has been driving a new generation of 10-Gbps Ethernet metropolitan/core network buildouts. Likely to be high-volume adopters in the medium term.
- **Enterprise LANs:** Continuing growth in convergence of voice/video/data and in unified communications is driving up network switch demands. However, most enterprises still rely on 1-Gbps or a mix of 1-Gbps and 10-Gbps Ethernet, and adoption of 100-Gbps Ethernet is likely to be slow.
- **Internet exchanges/ISP core routing:** With the massive amount of traffic flowing through these nodes, these installations are likely to be early adopters of 100-Gbps Ethernet.

Figure 1.6 shows an example of the application of 100-Gbps Ethernet. The trend at large data centers, with substantial banks of blade servers, is the deployment of 10-Gbps ports on individual servers to handle the massive multimedia traffic provided by these servers. Typically, a single blade server rack will contain multiple servers and one or two 10-Gbps Ethernet switches to interconnect all the servers and provide connectivity to the rest of the facility. The switches are often mounted in the rack and referred to as top-of-rack (ToR) switches. The term *ToR* has become synonymous with server access switch, even if it is not located "top of rack." For very large data centers, such as cloud providers, the interconnection of multiple blade server racks with additional 10-Gbps switches is increasingly inadequate. To handle the increased traffic load, switches operating at greater than 10 Gbps are needed to support the interconnection of server racks and to provide adequate capacity for connecting offsite through network interface controllers (NICs).

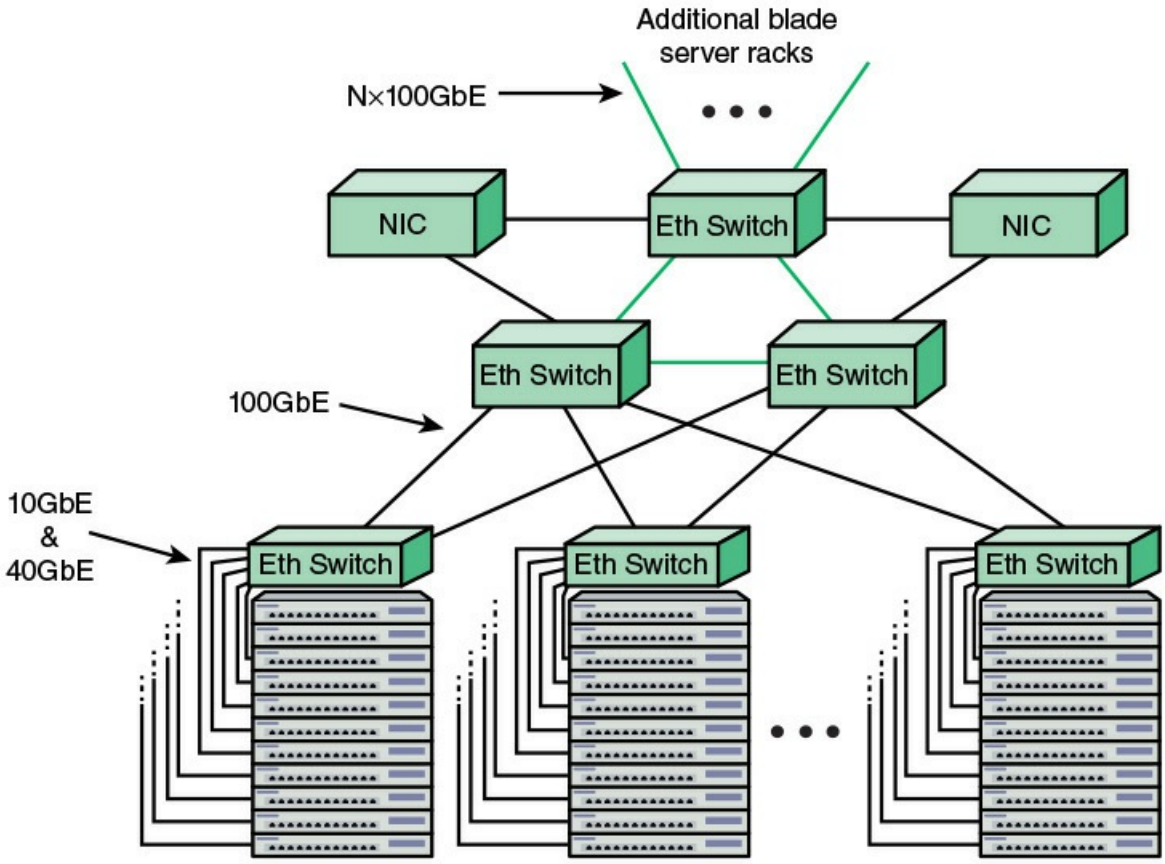


FIGURE 1.6 Configuration for Massive Blade Server Cloud Site

25/50-Gbps Ethernet

One of the options for implementing 100-Gbps is as four 25-Gbps physical lanes. Therefore, it would be relatively easy to develop standards for 25-Gbps and 50-Gbps Ethernet, using one or two lanes, respectively. Having these two lower-speed alternatives, based on the 100-Gbps technology, would give users more flexibility in meeting existing and near-term demands with a solution that would scale easily to higher data rates.

Such considerations have led to the form of the 25 Gigabit Ethernet Consortium by a number of leading cloud networking providers, including Google and Microsoft. The objective of the Consortium is to support an industry-standard, interoperable Ethernet specification that boosts the performance and slashes the interconnect cost per Gbps between the NIC and ToR switch. The specification adopted by the Consortium prescribes a single-lane 25-Gbps Ethernet and dual-lane 50-Gbps Ethernet link protocol, enabling up to 2.5 times higher performance per physical lane on twinax copper wire between the rack endpoint and switch compared to 10-Gbps and 40-Gbps Ethernet links. The IEEE 802.3 committee is at work developing the needed standards for 25 Gbps and may include 50 Gbps.

It is too early to say how these various options (25, 40, 50, 100 Gbps) will play out in the marketplace. In the intermediate term, the 100-Gbps switch is likely to predominate at large sites, but the availability of these slower and cheaper alternatives gives enterprises a number of paths

for scaling up to meet increasing demand.

400-Gbps Ethernet

The growth in demand never lets up. IEEE 802.3 is currently exploring technology options for producing a 400-Gbps Ethernet standard, although no timetable is yet in place. Looking beyond that milestone, there is widespread acknowledgment that a 1-Tbps (terabits per second, trillion bits per second) standard will eventually be produced.

2.5/5-Gbps Ethernet

As a testament to the versatility and ubiquity of Ethernet, and at the same time that ever higher data rates are being standardized, consensus is developing to standardize two lower rates: 2.5 Gbps and 5 Gbps. These relatively low speeds are also known as Multirate Gigabit BASE-T (MGBASE-T). Currently, the MGBASE-T Alliance is overseeing the development of these standards outside of IEEE. It is likely that the IEEE 802.3 committee will ultimately issue standards based on these industry efforts.

These new data rates are mainly intended to support IEEE 802.11ac wireless traffic into a wired network. IEEE 802.11ac is a 3.2-Gbps Wi-Fi standard that is gaining acceptance where more than 1 Gbps of throughput is needed, such as to support mobile users in the office environment. This new wireless standard overruns 1-Gbps Ethernet link support but may not require the next step up, which is 10 Gbps. Assuming that 2.5 and 5 Gbps can be made to work over the same cable that supports 1 Gbps, this would provide a much needed uplink speed improvement for access points supporting 802.11ac radios with their high bandwidth capabilities.

1.4 Wi-Fi

Just as Ethernet has become the dominant technology for wired LANs, so **Wi-Fi**, standardized by the IEEE 802.11 committee, has become the dominant technology for wireless LANs. This overview section discusses applications of Wi-Fi and then looks at standards and performance.

Applications of Wi-Fi

Wi-Fi is the predominant wireless Internet access technology, used in homes, offices, and public spaces. Wi-Fi in the home now connects computers, tablets, smartphones, and a host of electronic devices, such as video cameras, TVs, and thermostats. Wi-Fi in the enterprise has become an essential means of enhancing worker productivity and network effectiveness. And public Wi-Fi hotspots have expanded dramatically to provide free Internet access in must public places.

Wi-Fi in the Home

The first important use of Wi-Fi in the home was to replace Ethernet cabling for connecting desktop and laptop computers with each other and with the Internet. A typical layout is a desktop

computer with an attached router/modem that provides an interface to the Internet. Other desktop and laptop computers connect either via Ethernet or Wi-Fi to the central router, so that all the home computers can communicate with each other and with the Internet. Wi-Fi greatly simplified the hookup. Not only is there no need for a physical cable hookup, but the laptops can be moved easily from room to room or even outside the house.

Today, the importance of Wi-Fi in the home has expanded tremendously. Wi-Fi remains the default scheme for interconnecting a home computer network. Because both Wi-Fi and cellular capability are now standard on both smartphones and tablets, the home Wi-Fi provides a cost-effective way to the Internet. The smartphone or tablet will automatically use a Wi-Fi connection to the Internet if available, and only switch to the more expensive cellular connection if the Wi-Fi connection is not available. And Wi-Fi is essential to implementing the latest evolution of the Internet: the Internet of Things.

Public Wi-Fi

Access to the Internet via Wi-Fi has expanded dramatically in recent years, as more and more facilities provide a Wi-Fi hotspot, which enables any Wi-Fi device to attach. Wi-Fi hotspots are provided in coffee shops, restaurants, train stations, airports, libraries, hotels, hospitals, department stores, RV parks, and many other places. So many hotspots are available that it is rare to be too far from one. There are now numerous tablet and smartphone apps that increase their convenience.

Even very remote places will be able to support hotspots with the development of the satellite Wi-Fi hotspot. The first company to develop such a product is the satellite communications company Iridium. The satellite modem will initially provide a relatively low-speed connection, but the data rates will inevitably increase.

Enterprise Wi-Fi

The economic benefit of Wi-Fi is most clearly seen in the enterprise. Wi-Fi connections to the enterprise network have been offered by many organizations of all sizes, including public and private sector. But in recent years, the use of Wi-Fi has expanded dramatically, to the point that now approximately half of all enterprise network traffic is via Wi-Fi rather then the traditional Ethernet. Two trends have driven the transition to a Wi-Fi-centered enterprise. First, the demand has increased, with more and more employees preferring to use laptops, tablets, and smartphones to connect to the enterprise network, rather than a desktop computer. Second, the arrival of Gigabit Ethernet, especially the IEEE 802.ac standard, allows the enterprise network to support high-speed connections to many mobile devices simultaneously.

Whereas Wi-Fi once merely provided an accessory network designed to cover meetings and public areas, enterprise Wi-Fi deployment now generally provides ubiquitous coverage, to include main offices and remote facilities, and both indoor locations and outdoor spaces surrounding them. Enterprises accepted the need for, and then began to encourage, the practice known as bring your own device (BYOD). The almost universal availability of Wi-Fi capability on laptops, tablets, and smartphones, in addition to the wide availability of home and public Wi-Fi networks, has greatly benefited the organization. Employees can use the same devices and the same applications to continue their work or check their e-mail from wherever they are—home, at

their local coffee shop, or while traveling. From the enterprise perspective, this means higher productivity and efficiency and lower costs.

Standards

Essential to the success of Wi-Fi is interoperability. Wi-Fi-enabled devices must be able to communicate with Wi-Fi access points, such as the home router, the enterprise access point, and public hotspots, regardless of the manufacturer of the device or access point. Such interoperability is guaranteed by two organizations. First, the IEEE 802.11 wireless LAN committee develops the protocol and signaling standards for Wi-Fi. Then, the Wi-Fi Alliance creates test suites to certify interoperability for commercial products that conform to various IEEE 802.11 standards. The term *Wi-Fi* (wireless fidelity) is used for products certified by the Alliance.

IEEE 802.11 Wireless LAN Working Group

Wi-Fi Alliance

Wi-Fi Data Rates

Just as businesses and home users have generated a need to extend the Ethernet standard to speeds in the gigabits per second (Gbps) range, the same requirement exists for Wi-Fi. As the technology of antennas, wireless transmission techniques, and wireless protocol design has evolved, the IEEE 802.11 committee has been able to introduce standards for new versions of Wi-Fi at ever-higher speeds. Once the standard is issued, industry quickly develops the products. Here's a brief chronology, starting with the original standard, which was simply called IEEE 802.11, and showing the maximum data rate for each version (Figure 1.5):

- **802.11 (1997):** 2 Mbps (megabits per second, million bits per second)
- **802.11a (1999):** 54 Mbps
- **802.11b (1999):** 11 Mbps
- **802.11n (1999):** 600 Mbps
- **802.11g (2003):** 54 Mbps

- **802.11ad (2012):** 6.76 Gbps (billion bits per second)
- **802.11ac (2014):** 3.2 Gbps

IEEE 802.11ac operates in the 5-GHz band, as does the older and slower standards 802.11a and 802.11n. It is designed to provide a smooth evolution from 802.11n. This new standard makes use of advanced technologies in antenna design and signal processing to achieve much greater data rates, at lower battery consumption, all within the same frequency band as the older versions of Wi-Fi.

IEEE 802.11ad is a version of 802.11 operating in the 60-GHz frequency band. This band offers the potential for much wider channel bandwidth than the 5-GHz band, enabling high data rates with relatively simple signal encoding and antenna characteristics. Few devices operate in the 60-GHz band, which means communication experiences less interference than in the other bands used for Wi-Fi.

Because of the inherent transmission limitations of the 60-GHz band, 802.11ad is likely to be useful only within a single room. Because it can support high data rates and, for example, could easily transmit uncompressed high-definition video, it is suitable for applications such as replacing wires in a home entertainment system, or streaming high-definition movies from your cell phone to your television.

Gigabit Wi-Fi holds attractions for both office and residential environments and commercial products are beginning to roll out. In the office environment, the demand for ever greater data rates has led to Ethernet offerings at 10 Gbps, 40 Gbps, and most recently 100 Gbps. These stupendous capacities are needed to support blade servers, heavy reliance on video and multimedia, and multiple broadband connections offsite. At the same time, the use of wireless LANs has grown dramatically in the office setting to meet needs for mobility and flexibility. With the gigabit-range data rates available on the fixed portion of the office LAN, gigabit Wi-Fi is needed to enable mobile users to effectively use the office resources. IEEE 802.11ac is likely to be the preferred gigabit Wi-Fi option for this environment.

In the consumer and residential market, IEEE 802.11ad is likely to be popular as a low-power, short-distance wireless LAN capability with little likelihood of interfering with other devices. IEEE 802.11ad is also an attractive option in professional media production environments in which massive amounts of data need to be moved short distances.

1.5 4G/5G Cellular

Cellular technology is the foundation of mobile wireless communications and supports users in locations that are not easily served by wired networks. Cellular technology is the underlying technology for mobile telephones, personal communications systems, wireless Internet and wireless web applications, and much more. This section looks at how cellular technology has evolved through four generations and is poised for a fifth generation.

First Generation

The original cellular networks, now dubbed 1G, provided analog traffic channels and were designed to be an extension of the public switched telephone networks. Users with brick-sized

cell phones placed and received calls in the same fashion as landline subscribers. The most widely deployed 1G system was the Advanced Mobile Phone Service (AMPS), developed by AT&T. Voice transmission was purely analog and control signals were sent over a 10-kbps analog channel.

Second Generation

First-generation cellular networks quickly became highly popular, threatening to swamp available capacity. Second-generation (2G) systems were developed to provide higher-quality signals, higher data rates for support of digital services, and greater capacity. Key differences between 1G and 2G networks include the following:

- **Digital traffic channels:** The most notable difference between the two generations is that 1G systems are almost purely analog, whereas 2G systems are digital. In particular, 1G systems are designed to support voice channels; digital traffic is supported only by the use of a modem that converts the digital data into analog form. 2G systems provide digital traffic channels. These systems readily support digital data; voice traffic is first encoded in digital form before transmitting.
- **Encryption:** Because all the user traffic, and the control traffic, is digitized in 2G systems, it is a relatively simple matter to encrypt all the traffic to prevent eavesdropping. All 2G systems provide this capability, whereas 1G systems send user traffic in the clear, providing no security.
- **Error detection and correction:** The digital traffic stream of 2G systems also lends itself to the use of error detection and correction techniques. The result can be very clear voice reception.
- **Channel access:** In 1G systems, each cell supports a number of channels. At any given time a channel is allocated to only one user. 2G systems also provide multiple channels per cell, but each channel is dynamically shared by a number of users.

Third Generation

The objective of the third generation (3G) of wireless communication is to provide fairly high-speed wireless communications to support multimedia, data, and video in addition to voice. 3G systems share the following design features:

- **Bandwidth:** An important design goal for all 3G systems is to limit channel usage to 5 MHz. There are several reasons for this goal. On the one hand, a bandwidth of 5 MHz or more improves the receiver's ability to resolve multipath when compared to narrower bandwidths. On the other hand, the available spectrum is limited by competing needs, and 5 MHz is a reasonable upper limit on what can be allocated for 3G. Finally, 5 MHz is adequate for supporting data rates of 144 and 384 kbps, the main targets for 3G services.
- **Data rate:** Target data rates are 144 and 384 kbps. Some 3G systems also provide support up to 2 Mbps for office use.
- **Multirate:** The term *multirate* refers to the provision of multiple fixed-data-rate logical

channels to a given user, in which different data rates are provided on different logical channels. Further, the traffic on each logical channel can be switched independently through the wireless and fixed networks to different destinations. The advantage of multirate is that the system can flexibly support multiple simultaneous applications from a given user and can efficiently use available capacity by only providing the capacity required for each service.

Fourth Generation

The evolution of smartphones and cellular networks has ushered in a new generation of capabilities and standards, which is collectively called 4G. 4G systems provide ultra-broadband Internet access for a variety of mobile devices including laptops, smartphones, and tablets. 4G networks support Mobile web access and high-bandwidth applications such as high-definition mobile TV, mobile video conferencing, and gaming services.

These requirements have led to the development of a fourth generation (4G) of mobile wireless technology that is designed to maximize bandwidth and throughput while also maximizing spectral efficiency. 4G systems have the following characteristics:

- Based on an all-IP packet switched network
- Support peak data rates of up to approximately 100 Mbps for high-mobility mobile access and up to approximately 1 Gbps for low-mobility access such as local wireless access
- Dynamically share and use the network resources to support more simultaneous users per cell
- Support smooth handovers across heterogeneous networks
- Support high QoS for next-generation multimedia applications

In contrast to earlier generations, 4G systems do not support traditional circuit-switched telephony service, providing only IP telephony services.

Fifth Generation

5G systems are still some years away (perhaps 2020), but 5G technologies are likely an area of active research. By 2020, the huge amounts of data traffic generated by tablets and smartphones will be augmented by an equally huge, and perhaps much larger, amount of traffic from the *Internet of Things*, which includes shoes, watches, appliances, cars, thermostats, door locks, and much more.

With 4G, we may have reached a point of diminishing returns on network efficiency. There will be incremental improvements in the future, but significant increases in transmission efficiency seem unlikely. Instead, the focus for 5G will be on building more intelligence into the network, to meet service quality demands by dynamic use of priorities, adaptive network reconfiguration, and other network management techniques.

1.6 Cloud Computing

This section provides a brief overview of cloud computing, which is dealt with in greater detail later in the book.

Although the general concepts for cloud computing go back to the 1950s, cloud computing services first became available in the early 2000s, particularly targeted at large enterprises. Since then, cloud computing has spread to small- and medium-size businesses, and most recently to consumers. Apple's iCloud was launched in 2012 and had 20 million users within a week of launch. Evernote, the cloud-based note-taking and archiving service, launched in 2008, approached 100 million users in less than six years. In late 2014, Google announced that Google Drive had almost a quarter of a billion active users. Here, we look at the key elements of clouds, including cloud computing, cloud networking, and cloud storage.

➔ *See Chapter 13, "Cloud Computing"*

Cloud Computing Concepts

There is an increasingly prominent trend in many organizations to move a substantial portion or even all IT operations to an Internet-connected infrastructure known as enterprise **cloud computing**. At the same time, individual users of PCs and mobile devices are relying more and more on cloud computing services to back up data, sync devices, and share, using personal cloud computing.

The National Institute of Standards and Technology (NIST) defines the essential characteristics of cloud computing as follows:

- **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (for example, mobile phones, laptops, and personal digital assistants [PDAs]) and other traditional or cloud-based software services.
- **Rapid elasticity:** Cloud computing enables you to expand and reduce resources according to your specific service requirement. For example, you may need a large number of server resources for the duration of a specific task. You can then release these resources upon completion of the task.
- **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (for example, storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
- **On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider. Because the service is on demand, the resources are not permanent parts of your IT infrastructure.
- **Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a degree of location independence in that the customer generally has no control or knowledge over the

exact location of the provided resources, but may be able to specify location at a higher level of abstraction (for example, country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. Even private clouds tend to pool resources between different parts of the same organization.

Figure 1.7 illustrates the typical cloud service context. An enterprise maintains workstations within an enterprise LAN or set of LANs, which are connected by a router through a network or the Internet to the cloud service provider. The cloud service provider maintains a massive collection of servers, which it manages with a variety of network management, redundancy, and security tools. In the figure, the cloud infrastructure is shown as a collection of blade servers, which is a common architecture.

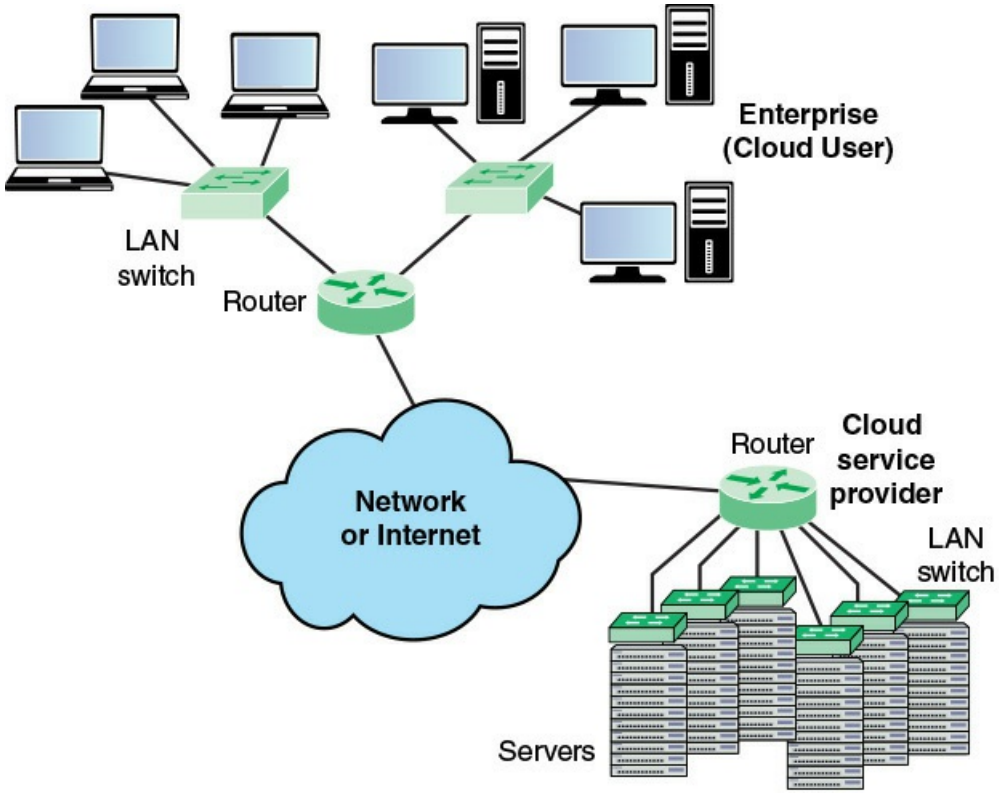


FIGURE 1.7 Cloud Computing Context

The Benefits of Cloud Computing

Cloud computing provides economies of scale, professional network management, and professional security management. These features can be attractive to companies large and small, government agencies, and individual PC and mobile users. The individual or company needs to pay only for the storage capacity and services they need. The user, be it company or individual, does not have the hassle of setting up a database system, acquiring the hardware they need, doing maintenance, and backup up the data; all this is part of the cloud service.

In theory, another big advantage of using cloud computing to store your data and share it with others is that the cloud provider takes care of security. Alas, the customer is not always protected. There have been a number of security failures among cloud providers. Evernote made

headlines in early 2013 when it told all of its users to reset their passwords after an intrusion was discovered. Cloud security is addressed in Chapter 16, "Security."

Cloud Networking

Cloud networking refers to the networks and network management functionality that must be in place to enable cloud computing. Many cloud computing solutions rely on the Internet, but that is only a piece of the networking infrastructure.

One example of cloud networking is the provisioning high-performance/high-reliability networking between the provider and subscriber. In this case, some or all of the traffic between an enterprise and the cloud bypasses the Internet and uses dedicated private network facilities owned or leased by the cloud service provider. More generally, cloud networking refers to the collection of network capabilities required to access a cloud, including making use of specialized services over the Internet, linking enterprise data centers to a cloud, and using firewalls and other network security devices at critical points to enforce access security policies.

Cloud Storage

We can think of cloud storage as a subset of cloud computing. In essence, cloud storage consists of database storage and database applications hosted remotely on cloud servers. Cloud storage enables small businesses and individual users to take advantage of data storage that scales with their needs and to take advantage of a variety of database applications without having to buy, maintain, and manage the storage assets.

1.7 Internet of Things

The **Internet of Things (IoT)** is the latest development in the long and continuing revolution of computing and communications. Its size, ubiquity, and influence on everyday lives, business, and government dwarf any technical advance that has gone before. This section provides a brief overview of the IoT, which is dealt with in greater detail later in the book.

Things on the Internet of Things

The *Internet of Things (IoT)* is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors. A dominant theme is the embedding of short-range mobile transceivers into a wide array of gadgets and everyday items, enabling new forms of communication between people and things, and between things themselves. The Internet now supports the interconnection of billions of industrial and personal objects, usually through cloud systems. The objects deliver sensor information, act on their environment, and in some cases modify themselves, to create overall management of a larger system, like a factory or city.

➔ *See Chapter 14, "The Internet of Things"*

The IoT is primarily driven by deeply embedded devices. These devices are low-bandwidth, low-repetition data-capture and low-bandwidth data-usage appliances that communicate with each

other and provide data via user interfaces. Embedded appliances, such as high-resolution video security cameras, Video over IP (VoIP) phones, and a handful of others, require high-bandwidth streaming capabilities. Yet countless products simply require packets of data to be intermittently delivered.

Evolution

With reference to the end systems supported, the Internet has gone through roughly four generations of deployment culminating in the IoT:

1. **Information technology (IT):** PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people, primarily using wired connectivity.
2. **Operational technology (OT):** Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA (supervisory control and data acquisition), process control, and kiosks, bought as appliances by enterprise OT people and primarily using wired connectivity.
3. **Personal technology:** Smartphones, tablets, and ebook readers bought as IT devices by consumers (employees) exclusively using wireless connectivity and often multiple forms of wireless connectivity.
4. **Sensor/actuator technology:** Single-purpose devices bought by consumers, IT, and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems.

It is the fourth generation that is usually thought of as the IoT, and which is marked by the use of billions of embedded devices.

Layers of the Internet of Things

Both the business and technical literature often focus on two elements of the Internet of Things —the "things" that are connected, and the Internet that interconnects them. It is better to view the IoT as a massive system, which consists of five layers:

- **Sensors and actuators:** These are the things. Sensors observe their environment and report back quantitative measurements of such variables as temperature, humidity, presence or absence of some observable, and so on. Actuators operate on their environment, such as changing a thermostat setting or operating a valve.
- **Connectivity:** A device may connect via either a wireless or wired link into a network to send collected data to the appropriate data center (sensor) or receive operational commands from a controller site (actuator).
- **Capacity:** The network supporting the devices must be able to handle a potentially huge flow of data.
- **Storage:** There needs to be a large storage facility to store and maintain backups of all the collected data. This is typically a cloud capability.
- **Data analytics**. For large collections of devices, "big data" is generated, requiring a data

analytics capability to process the data flow.

All of these layers are essential to an effective use of the IoT concept.

1.8 Network Convergence

Network convergence refers to the merger of previously distinct telephony and information technologies and markets. You can think of this convergence in terms of a three-layer model of enterprise communications:

- **Application convergence:** These are seen by the end users of a business. Convergence integrates communications applications, such as voice calling (telephone), voice mail, e-mail, and instant messaging, with business applications, such as workgroup collaboration, customer relationship management, and back-office functions. With convergence, applications provide rich features that incorporate voice, data, and video in a seamless, organized, and value-added manner. One example is multimedia messaging, which enables a user to use a single interface to access messages from a variety of sources (for example, office voice mail, e-mail, SMS text messages, and mobile voice mail).
- **Enterprise services:** At this level, the manager deals with the information network in terms of the services that must be available to ensure that users can take full advantage of the applications that they use. For example, network managers need to make sure that appropriate privacy mechanisms and authentication services are in place to support convergence-based applications. They may also be able to track user locations to support remote print services and network storage facilities for mobile workers. Enterprise network management services may also include setting up collaborative environments for various users, groups, and applications and QoS provision.
- **Infrastructure:** The network and communications infrastructure consists of the communication links, LANs, WANs, and Internet connections available to the enterprise. Increasingly, enterprise network infrastructure also includes private/public cloud connections to data centers that host high-volume data storage and web services. A key aspect of convergence at this level is the ability to carry voice, image, and video over networks that were originally designed to carry data traffic. Infrastructure convergence has also occurred for networks that were designed for voice traffic. For example, video, image, text, and data are routinely delivered to smartphone users over cell phone networks.

Figure 1.8 illustrates the major attributes of the three-layer model of enterprise communications. In simple terms, convergence involves moving an organization's voice, video, and image traffic to a single network infrastructure. This often involves integrating distinct voice and data networks into a single network infrastructure and extending the infrastructure to support mobile users. The foundation of this convergence is packet-based transmission using the Internet Protocol (IP). Using IP packets to deliver all varieties of communications traffic, sometimes referred to as everything over IP, enables the underlying infrastructure to deliver a wide range of useful applications to business users.

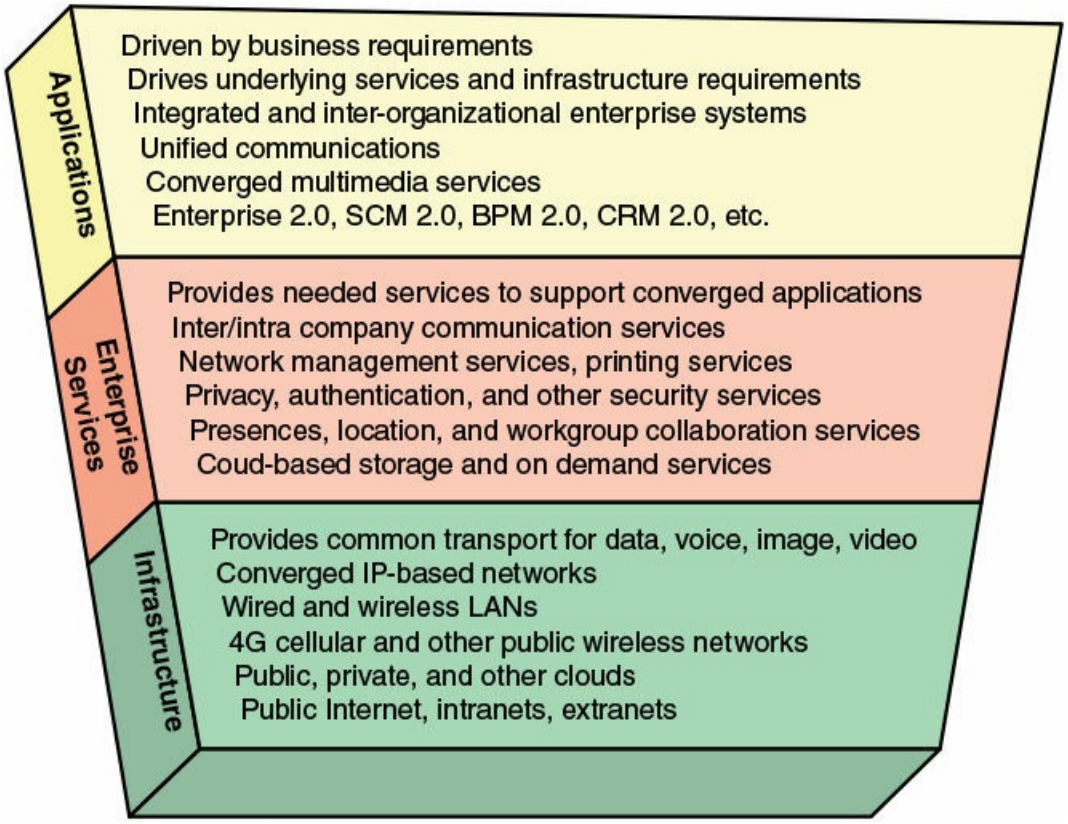


FIGURE 1.8 Business-Driven Convergence

Convergence brings many benefits, including simplified network management, increased efficiency, and greater flexibility at the application level. For example, a converged network infrastructure provides a predictable platform on which to build new add applications that combine video, data, and voice. This makes it easier for developers to create innovative mash-ups and other value-added business applications and services. The following list summarizes three key benefits of IP network convergence:

- **Cost savings:** A converged network can provide significant double-digit percent reductions in network administration, maintenance, and operating costs; converging legacy networks onto a single IP network enables better use of existing resources, and implementation of centralized capacity planning, asset management, and policy management.
- **Effectiveness:** The converged environment has the potential to provide users with great flexibility, irrespective of where they are. IP convergence allows companies to create a more mobile workforce. Mobile workers can use a virtual private network (VPN) to remotely access business applications and communication services on the corporate network. A VPN helps maintain enterprise network security by separating business traffic from other Internet traffic.
- **Transformation:** Because they are modifiable and interoperable, converged IP networks can easily adapt to new functions and features as they become available through technological advancements without having to install new infrastructure. Convergence also enables the enterprise-wide adoption of global standards and best practices, thus

providing better data, enhanced real-time decision making, and improved execution of key business processes and operations. The end result is enhanced agility and innovation, the key ingredients of business innovation.

These compelling business benefits are motivating companies to invest in converged network infrastructures. Businesses, however, are keenly aware of the downside of convergence: having a single network means a single point of failure. Given their reliance on ICT (information and communications technology), today's converged enterprise network infrastructures typically include redundant components and back up systems to increase network resiliency and lessen the severity of network outages.

1.9 Unified Communications

A concept related to network convergence is **unified communications** (UC). Whereas enterprise network convergence focuses on the consolidation of traditionally distinct voice, video, and data communications networks into a common infrastructure, UC focuses on the integration of real-time communication services to optimize business processes. As with converged enterprise networks, IP is the cornerstone on which UC systems are built. Key elements of UC include the following:

1. UC systems typically provide a unified user interface and consistent user experience across multiple devices and media.
2. UC merges real-time communications services with non-real-time services and business process applications.

Figure 1.9 shows the typical components of a UC architecture and how they relate to one another.

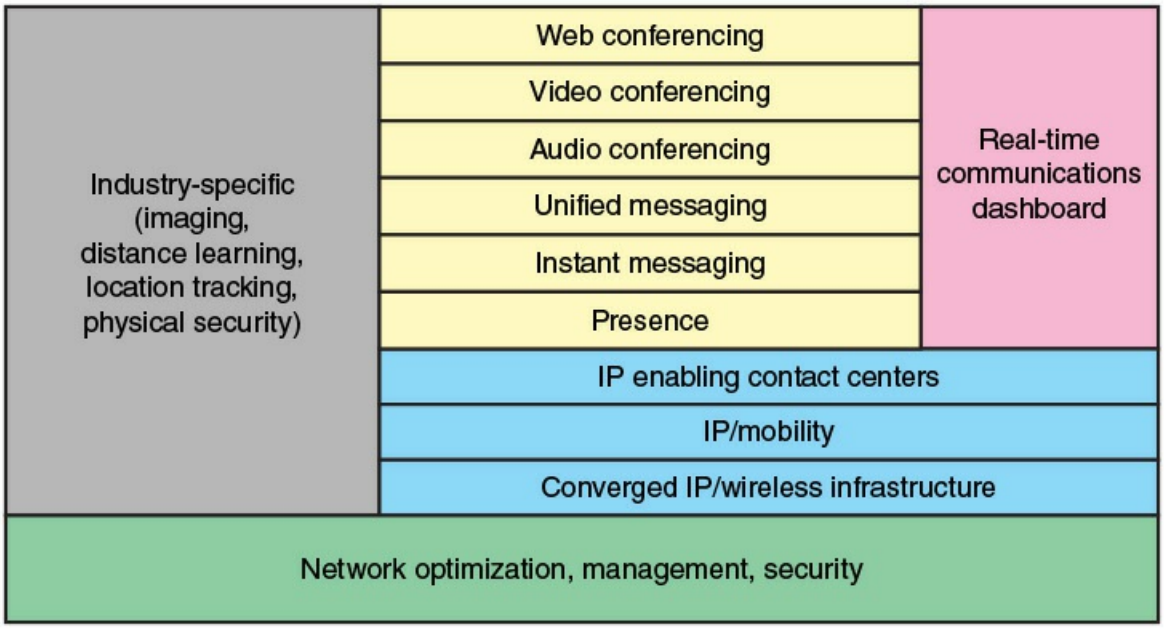


FIGURE 1.9 Elements of a Unified Communications Architecture

The key elements of this architecture are as follows:

- **Real-time communications (RTC) dashboard:** An RTC dashboard is a key component of UC architecture. This is the element that provides UC users with a unified user interface across communication devices. Ideally, the user has a consistent interface no matter what communication device the user is currently using, whether it is a cell phone, wireless tablet computer, desktop system, or office telephone attached to the corporate private branch exchange (PBX). As you can see in Figure 1.9, RTC dashboards provide access to real-time communication services such as instant messaging, audio and video conferencing, and interactive whiteboards; RTC dashboards also provide access to non-real-time services such as unified messaging (e-mail, voice mail, fax, and SMS) in unified view. An RTC dashboard includes presence information about co-workers and partners so that users can know on the fly which colleagues are available to communicate or join a collaborative communication session. RTC dashboards have become necessities in organizations that require high levels of communication and collaboration to support business processes.
- **Web conferencing:** Refers to live meetings or presentations in which participants access the meeting or presentation via a mobile device or the web, either over the Internet, or corporate intranet. Web conferences often include data sharing through web-connected interactive white boards (IWBs).
- **Audio conferencing:** Also called conference calling, refers to a live meeting in which participants are linked together for audio transmission and reception. A participant may be on a landline, mobile phone, or at a "softphone"—a computer equipped with microphone and speaker.
- **Unified messaging:** Unified messaging systems provide a common repository for messages from multiple sources. It allows users to retrieve saved e-mail, voice mail, and fax messages from a computer, telephone, or mobile device. Computer users can select and play voice-mail recordings that appear in their unified messaging inboxes. Telephone users can both retrieve voice mail and hear text-to-voice translations of e-mail messages. Messages of any type can be saved, answered, filed, sorted, and forwarded. Unified messaging systems relieve business users from having to monitor multiple voice mailboxes by enabling voicemail messages received by both office phones and cell phones to be saved to the same mailbox. With UC, users can use any device at any time to retrieve e-mail or voice-mail from unified messaging mailboxes.
- **Instant messaging (IM):** Real-time text-based messaging between two or more participants. IM is similar to online chat because it is text-based and exchanged bidirectionally in real time. IM is distinct from chat in that IM clients use contact (or buddy) lists to facilitate connections between known users, whereas online chat can include text-based exchanges between anonymous users.
- **Video teleconferencing (VTC):** Videoconferencing allows users in two or more locations to interact simultaneously via two-way video and audio transmission. UC systems enable users to participate in video conferences via desktop computers, smartphones, and mobile devices.
- **Presence:** The capability to determine, in real time, where someone is, how that person

prefers to be reached, and even what the person is currently doing. Presence information shows the individual's availability state before co-workers attempt to contact them person. It was once considered simply an underlying technology to instant messaging (for example, "available to chat" or "busy") but has been broadened to include whether co-workers are currently on office or mobile phones, logged in to a computer, involved in a video call or in a meeting, or out of the office for lunch or vacation. A co-worker's geographic location is becoming more common as an element in presence information for a number of business reasons, including the capability to quickly respond to customer emergencies. Business has embraced presence information because it facilitates more efficient and effective communication. It helps eliminate inefficiencies associated with "phone tag" or composing and sending e-mails to someone who could more quickly answer a question over the phone or with a quick meeting.

- **IP enabling contact centers:** Refers to the use of IP-based unified communications to enhance customer contact center functionality and performance. The unified communications infrastructure makes use of presence technology to enable customers and internal enterprise employees to be quickly connected to the required expert or support person. In addition, this technology supports mobility, so that call center personnel need not be located at a particular office or remain in a particular place. Finally, the UC infrastructure enables the call center employee to quickly access other employees and information assets, including data, video, image, and audio.
- **IP/mobility:** Refers to the delivery of information to and collection of information from enterprise personnel who are usually mobile, using an IP network infrastructure. In a typical enterprise, upward of 30 percent of employees use some form of weekly remote access technology in the performance of their jobs.
- **Converged IP/wireless infrastructure:** A unified networking and communications-based IP packet transfer to support voice, data, and video transmission and can be extended to include local- and wide-area wireless communications. UC-enabled mobile devices are able to switch between Wi-Fi and cellular systems in the middle of a communication session. For example, a UC user could receive a co-worker's call via a smartphone connected to Wi-Fi network at home, continue the conversation while driving to work over a cellular network connection, and could end the call at the office while connected to the business's Wi-Fi network. Both handoffs (home Wi-Fi to cellular and cellular to office Wi-Fi) would take place seamlessly and transparently without dropping the call.

The importance of UC is not only that it integrates communication channels but also that it offers a way to integrate communication functions and business applications. Three major categories of benefits are typically realized by organizations that use UC:

- **Personal productivity gains:** Presence information helps employees find each other and choose the most effective way to communicate in real time. Less time is wasted calling multiple numbers to locate co-workers or checking multiple worked-related voice mailboxes. Calls from VIP contacts can be routed simultaneously to all of a UC user's phone devices (office phone, softphone, smartphone, home phone) to ensure faster responsiveness to customers, partners, and co-workers. With mobile presence information capabilities, employees who are geographically closest can be dispatched to address a problem.

- **Workgroup performance gains:** UC systems support real-time collaboration among team members, which facilitates workgroup performance improvements. Examples include the use of presence information to speed identification of an available individual with the right skills a work team needs to address a problem. Enhanced conferencing capabilities with desktop VTC and interactive white boards and automated business rules to route or escalate communications also help to increase workgroup performance.
- **Enterprise-level process improvements:** IP convergence enables UC to be integrated with enterprise-wide and departmental-level applications, business processes, and workflows. UC-enabled enhanced communications with customers, suppliers, and business partners are redefining best practices for customer relationship management (CRM), supply chain management (SCM), and other enterprise-wide applications and are transforming relationships among members of business networks. Communication-enabled business processes (CEBP) are fueling competition in several industries, including financial services, healthcare, and retail.

1.10 Key Terms

After completing this chapter, you should be able to define the following terms.

3G

4G

5G

access network

aggregation router

application provider

application service provider

backbone network

blade server

cloud computing

cloud networking

cloud storage

content provider

core network

core router

distribution network

edge router

end users

IEEE 802.3

IEEE 802.11

Internet of Things (IoT)

Ethernet

network convergence

network provider

peering

Power over Ethernet (PoE)

powerline carrier (PLC)

top-of-rack (ToR) switch

unified communications

Wi-Fi

1.11 References

XI11: Xi, H. “Bandwidth Needs in Core and Aggregation Nodes in the Optical Transport Network.” IEEE 802.3 Industry Connections Ethernet Bandwidth Assessment Meeting, November 8, 2011. http://www.ieee802.org/3/ad_hoc/bwa/public/nov11/index_1108.html

Chapter 2. Requirements and Technology

> Networks will make possible many straightforward and significant economies. There will be problems such as loss of control, a potential lack of responsiveness to changing needs, and priority conflicts; but many of these problems have already been solved to a considerable degree.
>
> —*What Can Be Automated?*
> The Computer Science and Engineering Research Study, National Science Foundation, 1980

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

- Present an overview of the major categories of packet traffic on the Internet and internets, including elastic, inelastic, and real-time traffic.
- Discuss the traffic demands placed on contemporary networks by big data, cloud computing, and mobile traffic.
- Explain the concept of quality of service.
- Explain the concept of quality of experience.
- Understand the essential elements of routing.
- Understand the effects of congestion and the types of techniques used for congestion control.
- Compare and contrast software-defined networking and network functions virtualization.

Chapter 1, "Elements of Modern Networking," provided a survey of the elements that make up the networking ecosystem, including network technologies, network architecture, services, and applications. In a concise fashion, this chapter provides motivation, technical background, and an overview of the key topics covered in this book.

2.1 Types of Network and Internet Traffic

Traffic on the Internet and enterprise networks can be divided into two broad categories: elastic and inelastic. A consideration of their differing requirements clarifies the need for an enhanced networking architecture.

Elastic Traffic

Elastic traffic is that which can adjust, over wide ranges, to changes in delay and throughput across an **internet** and still meet the needs of its applications. This is the traditional type of traffic supported on TCP/IP-based internets and is the type of traffic for which internets were

designed. Applications that generate such traffic typically use Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) as a transport protocol. In the case of UDP, the application will use as much capacity as is available up to the rate that the application generates data. In the case of TCP, the application will use as much capacity as is available up to the maximum rate that the end-to-end receiver can accept data. Also with TCP, traffic on individual connections adjusts to congestion by reducing the rate at which data are presented to the network.

Applications that can be classified as elastic include the common applications that operate over TCP or UDP, including file transfer (File Transfer Protocol / Secure FTP [FTP/SFTP]), electronic mail (Simple Mail Transport Protocol [SMTP]), remote login (Telnet, Secure Shell [SSH]), network management (Simple Network Management Protocol [SNMP]), and web access (Hypertext Transfer Protocol / HTTP Secure [HTTP/HTTPS]). However, there are differences among the requirements of these applications, including the following:

- E-mail is generally insensitive to changes in delay.
- When file transfer is done via user command rather than as an automated background task, the user expects the delay to be proportional to the file size and so is sensitive to changes in throughput.
- With network management, delay is generally not a serious concern. However, if failures in an internet are the cause of congestion, then the need for SNMP messages to get through with minimum delay increases with increased congestion.
- Interactive applications, such as remote logon and web access, are sensitive to delay.

It is important to realize that it is not per-packet delay that is the quantity of interest. Observation of real delays across the Internet suggest that wide variations in delay do not occur. Because of the congestion control mechanisms in TCP, when congestion develops, delays only increase modestly before the arrival rate from the various TCP connections slow down. Instead, the quality of service (QoS) perceived by the user relates to the total elapsed time to transfer an element of the current application. For an interactive Telnet-based application, the element may be a single keystroke or single line. For web access, the element is a web page, which could be as little as a few kilobytes or could be substantially larger for an image-rich page. For a scientific application, the element could be many megabytes of data.

For very small elements, the total elapsed time is dominated by the delay time across the Internet. However, for larger elements, the total elapsed time is dictated by the sliding-window performance of TCP and is therefore dominated by the throughput achieved over the TCP connection. Thus, for large transfers, the transfer time is proportional to the size of the file and the degree to which the source slows because of congestion.

It should be clear that even if you confine your attention to elastic traffic, some service prioritizing and controlling traffic could be of benefit. Without such a service, routers are dealing evenhandedly with arriving IP packets, with no concern for the type of application and whether a particular packet is part of a large transfer element or a small one. Under such circumstances, and if congestion develops, it is unlikely that resources will be allocated in such a way as to meet the needs of all applications fairly. When inelastic traffic is added to the mix, the results are even more unsatisfactory.

Inelastic Traffic

Inelastic traffic does not easily adapt, if at all, to changes in delay and throughput across an internet. Examples of inelastic traffic include multimedia transmission, such as voice and video, and high-volume interactive traffic, such as an interactive simulation application (for example, airline pilot simulation). The requirements for inelastic traffic may include the following:

- **Throughput:** A minimum throughput value may be required. Unlike most elastic traffic, which can continue to deliver data with perhaps degraded service, many inelastic applications absolutely require a given minimum throughput.
- **Delay**: Also called latency. An example of a delay-sensitive application is stock trading; someone who consistently receives later service will consistently act later, and with greater disadvantage.
- **Delay jitter:** The magnitude of delay variation, called **delay jitter**, or simply jitter, is a critical factor in real-time applications. Because of the variable delay imposed by an internet, the interarrival times between packets are not maintained at a fixed interval at the destination. To compensate for this, the incoming packets are buffered, delayed sufficiently to compensate for the jitter, and then released at a constant rate to the software that is expecting a steady real-time stream. The larger the allowable delay variation, the longer the real delay in delivering the data and the greater the size of the delay buffer required at receivers. Real-time interactive applications, such as teleconferencing, may require a reasonable upper bound on jitter.
- **Packet loss**: Real-time applications vary in the amount of packet loss, if any, that they can sustain.

Table 2.1 shows the loss, delay, and jitter characteristics of various classes of traffic, as specified in RFC 4594 (*Configuration Guidelines for DiffServ Service Classes*, August 2006). Table 2.2 gives examples of QoS requirements for various media-oriented applications [SZIG14].

Application Category	Service Class	Traffic Characteristics	Tolerance to Loss	Tolerance to Delay	Tolerance to Jitter
Control	Network control	Variable-size packets, mostly inelastic short messages, but traffic can also burst (BGP)	Low	Low	Yes
	OA&M	Variable-size packets, elastic and inelastic flows	Low	Medium	Yes
Media-Oriented	Telephony	Fixed-size small packets, constant emission rate, inelastic and low-rate flows	Very low	Very low	Very low
	Real-time interactive	RTP/UDP streams, inelastic, mostly variable rate	Low	Very low	Low
	Multimedia conferencing	Variable-size packets, constant transmit interval, rate adaptive, reacts to loss	Low-medium	Very low	Low
	Broadcast video	Constant and variable rate, inelastic, nonbursty flows	Very low	Medium	Low
	Multimedia Streaming	Variable-size packets, elastic with variable rate	Low-medium	Medium	Yes
Data	Low-latency data	Variable rate, bursty short-lived elastic flows	Low	Low-medium	Yes
	High-throughput data	Variable rate, bursty long-lived elastic flows	Low	Medium-high	Yes
	Low-priority data	Non-real-time and elastic	High	High	Yes
Best effort	Standard	A bit of everything	Not specified		

BGP = Border Gateway Protocol

OA&M = Operations, administration, and management

RTP = Real-time Transport Protocol

UDP = User Datagram Protocol

Table 2.1 Service Class Characteristics

Voice	One-way latency ≤ 150 ms
	One-way peak-to-peak jitter ≤ 30 ms
	Per-hop peak-to-peak jitter ≤ 10 ms
	Packet loss ≤ 1 percent
Broadcast video	Packet loss ≤ 0.1 percent
Real-time interactive video	One-way latency ≤ 200 ms
	One-way peak-to-peak jitter ≤ 50 ms
	Per-hop peak-to-peak jitter ≤ 10 ms
	Packet loss ≤ 0.1 percent
Multimedia conferencing	One-way latency ≤ 200 ms
	Packet loss ≤ 1 percent
Multimedia streaming	One-way latency ≤ 400 ms
	Packet loss ≤ 1 percent

Table 2.2 QoS Requirements by Application Class

These requirements are difficult to meet in an environment with variable queuing delays and congestion losses. Accordingly, inelastic traffic introduces two new requirements into the internet architecture. First, some means is needed to give preferential treatment to applications with more demanding requirements. Applications need to be able to state their requirements, either ahead of time in some sort of service request function, or on the fly, by means of fields in the IP packet header. The former approach provides more flexibility in stating requirements, and it enables the network to anticipate demands and deny new requests if the required resources are unavailable. This approach implies the use of some sort of resource reservation protocol.

An additional requirement in supporting inelastic traffic in an internet architecture is that elastic traffic must still be supported. Inelastic applications typically do not back off and reduce demand in the face of congestion, in contrast to TCP-based applications. Therefore, in times of congestion, inelastic traffic will continue to supply a high load, and elastic traffic will be crowded off the internet. A reservation protocol can help control this situation by denying service requests that would leave too few resources available to handle current elastic traffic.

Real-Time Traffic Characteristics

As mentioned, a common example of inelastic traffic is **real-time traffic**. With traditional elastic applications, such as file transfer, electronic mail, and client/server applications including the web, the performance metrics of interest are generally throughput and delay. There is also a concern with reliability, and mechanisms are used to make sure that no data are lost, corrupted, or misordered during transit. By contrast, **real-time** applications are concerned with timing issues as well as packet loss. In most cases, there is a requirement that data be delivered at a constant rate equal to the sending rate. In other cases, a deadline is associated with each block of data, such that the data are not usable after the deadline has expired.

Figure 2.1 illustrates a typical real-time environment. Here, a server is generating audio to be transmitted at 64 kbps. The digitized audio is transmitted in packets containing 160 octets of

data, so that one packet is issued every 20 ms. These packets are passed through an internet and delivered to a multimedia PC, which plays the audio in real time as it arrives. However, because of the variable delay imposed by the internet, the interarrival times between packets are not maintained at a fixed 20 ms at the destination. To compensate for this, the incoming packets are buffered, delayed slightly, and then released at a constant rate to the software that generates the audio. The buffer may be internal to the destination machine or in an external network device.

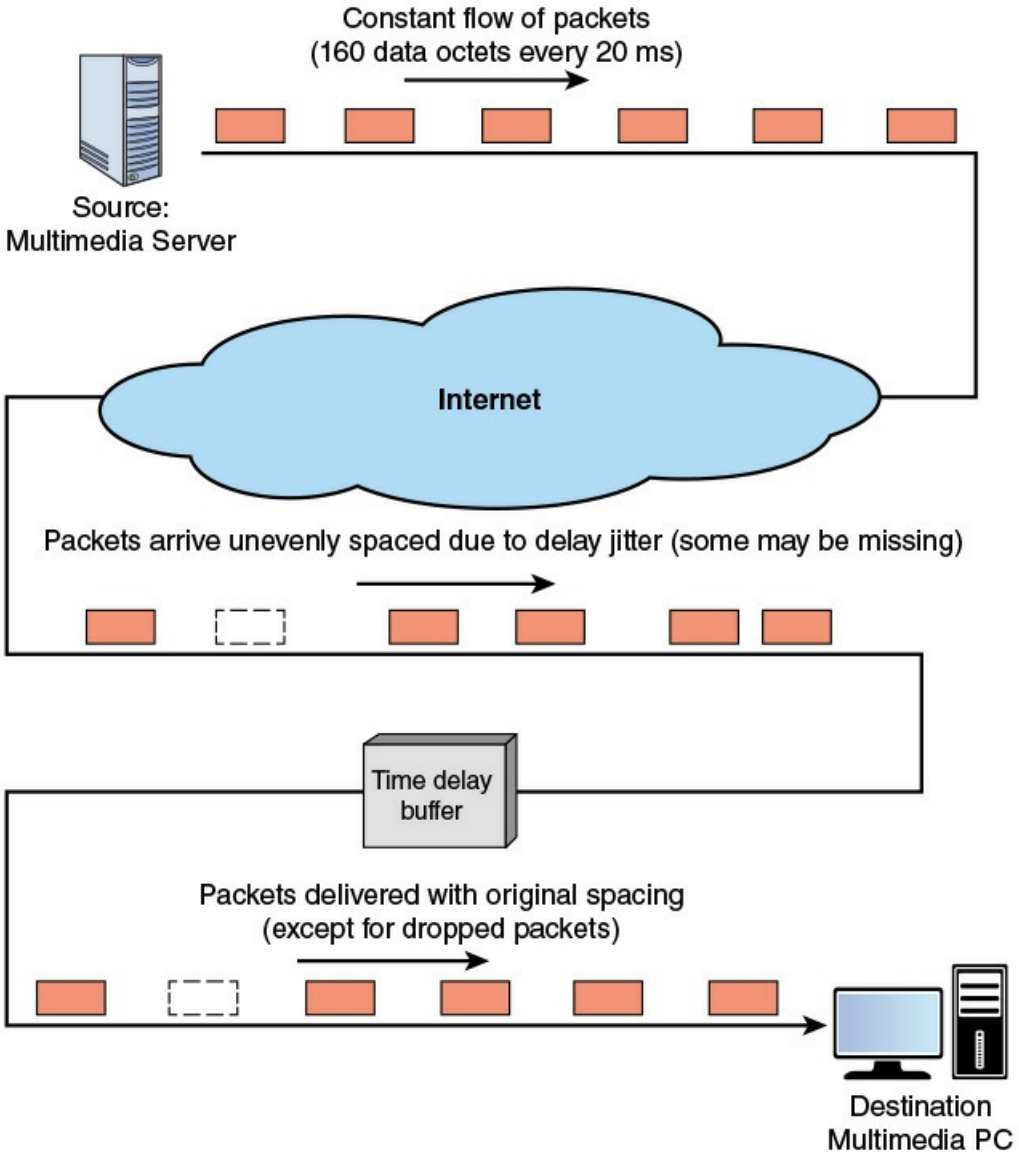


FIGURE 2.1 Real-Time Traffic

The compensation provided by the delay buffer is limited. For example, if the minimum end-to-end delay seen by any packet is 1 ms and the maximum is 6 ms, the delay jitter is 5 ms. As long as the time delay buffer delays incoming packets by at least 5 ms, the output of the buffer will include all incoming packets. However, if the buffer delayed packets by only 4 ms, any incoming packets that had experienced a relative delay of more than 4 ms (an absolute delay of more than 5 ms) would have to be discarded so as not to be played back out of order.

The description of real-time traffic so far implies a series of equal-size packets generated at a constant rate. This is not always the profile of the traffic. Figure 2.2 illustrates some of the common possibilities, as described in the list that follows.

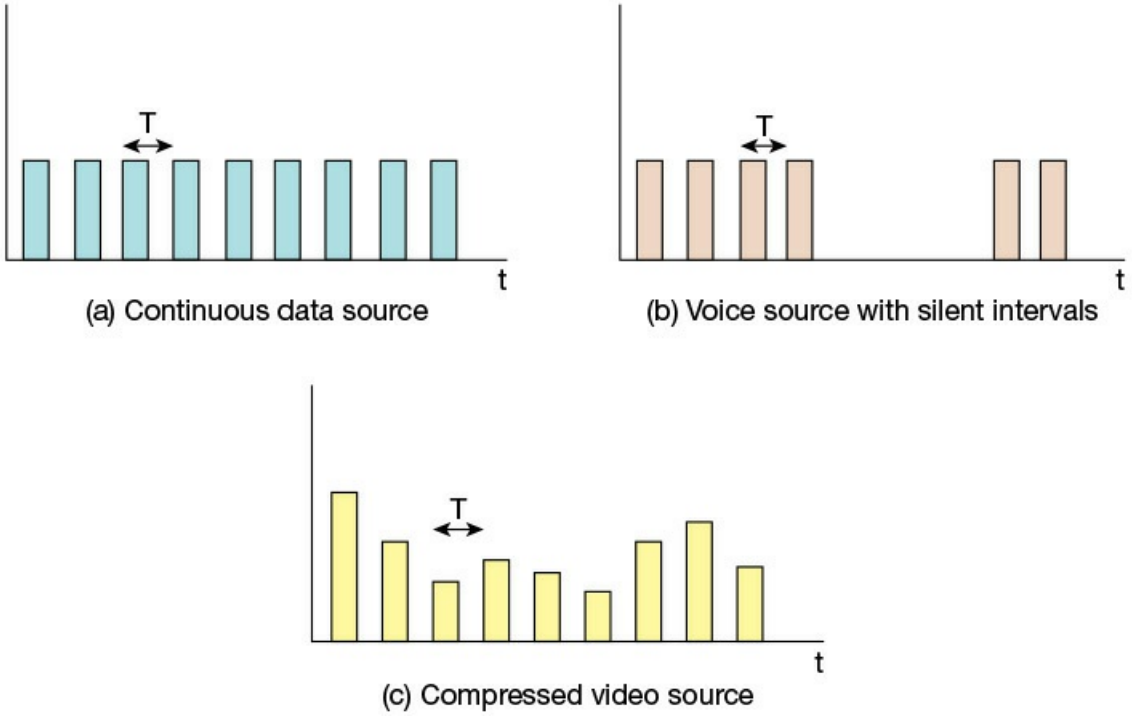


FIGURE 2.2 Real-Time Packet Transmission

- **Continuous data source:** Fixed-size packets are generated at fixed intervals. This characterizes applications that are constantly generating data, have few redundancies, and that are too important to compress in a lossy way. Examples are air traffic control radar and real-time simulations.
- **On/off source:** The source alternates between periods when fixed-size packets are generated at fixed intervals and periods of inactivity. A voice source, such as in telephony or audio conferencing, fits this profile.
- **Variable packet size**: The source generates variable-length packets at uniform intervals. An example is digitized video in which different frames may experience different compression ratios for the same output quality level.

2.2 Demand: Big Data, Cloud Computing, and Mobile Traffic

Having looked at the types of traffic presented to the Internet and other IP-based networks, consider the application areas that are generating the greatest stress on network resources and management. Three areas stand out: big data, cloud computing, and mobility. All of these areas suggest the need for using powerful tools such as software-defined networking (SDN) and network functions virtualization (NFV) for network operation and management, and for using comprehensive QoS and quality of experience (QoE) systems for effective delivery of services over IP-based networks.

Big Data

In simple terms, **big data** refers to everything that enables an organization to create, manipulate, and manage very large data sets (measured in terabytes, petabytes, exabytes, and so on) and the facilities in which these are stored. Distributed data centers, data warehouses, and cloud-based storage are common aspects of today's enterprise networks. Many factors have contributed to the merging of "big data" and business networks, including continuing declines in storage costs, the maturation of data mining and business intelligence (BI) tools, and government regulations and court cases that have caused organizations to stockpile large masses of structured and unstructured data, including documents, e-mail messages, voice-mail messages, text messages, and social media data. Other data sources being captured, transmitted, and stored include web logs, Internet documents, Internet search indexing, call detail records, scientific research data and results, military surveillance, medical records, video archives, and e-commerce transactions.

Data sets continue to grow with more and more being gathered by remote sensors, mobile devices, cameras, microphones, radio frequency identification (RFID) readers, and similar technologies. One study from a few years ago estimated that 2.5 exabytes (2.5×10^{18} bytes) of data are created each day, and 90 percent of the data in the world was created in the past two years [IBM11]. Those numbers are likely higher today.

Big Data Infrastructure Considerations

Traditional business data storage and management technologies include relational database management systems (RDBMS), network-attached storage (NAS), storage-area networks (SANs), data warehouses (DWs), and business intelligence (BI) analytics.

Traditional data warehouse and BI analytics systems tend to be highly centralized within an enterprise infrastructure. These often include a central data repository with a RDBMS, high-performance storage, and **analytics** software, such as online analytical processing (OLAP) tools for mining and visualizing data.

Increasingly, big data applications are becoming a source of competitive value for businesses, especially those that aspire to build data products and services to profit from the huge volumes of data that they capture and store. There is every indication that the exploitation of data will become increasingly important to enterprises in the years ahead as more and more businesses reap the benefits of big data applications.

Big Data Networking Example

To get some feel for the networking requirements for a typical big data system, consider the example ecosystem of Figure 2.3 (compared to Figure 1.1 from Chapter 1).

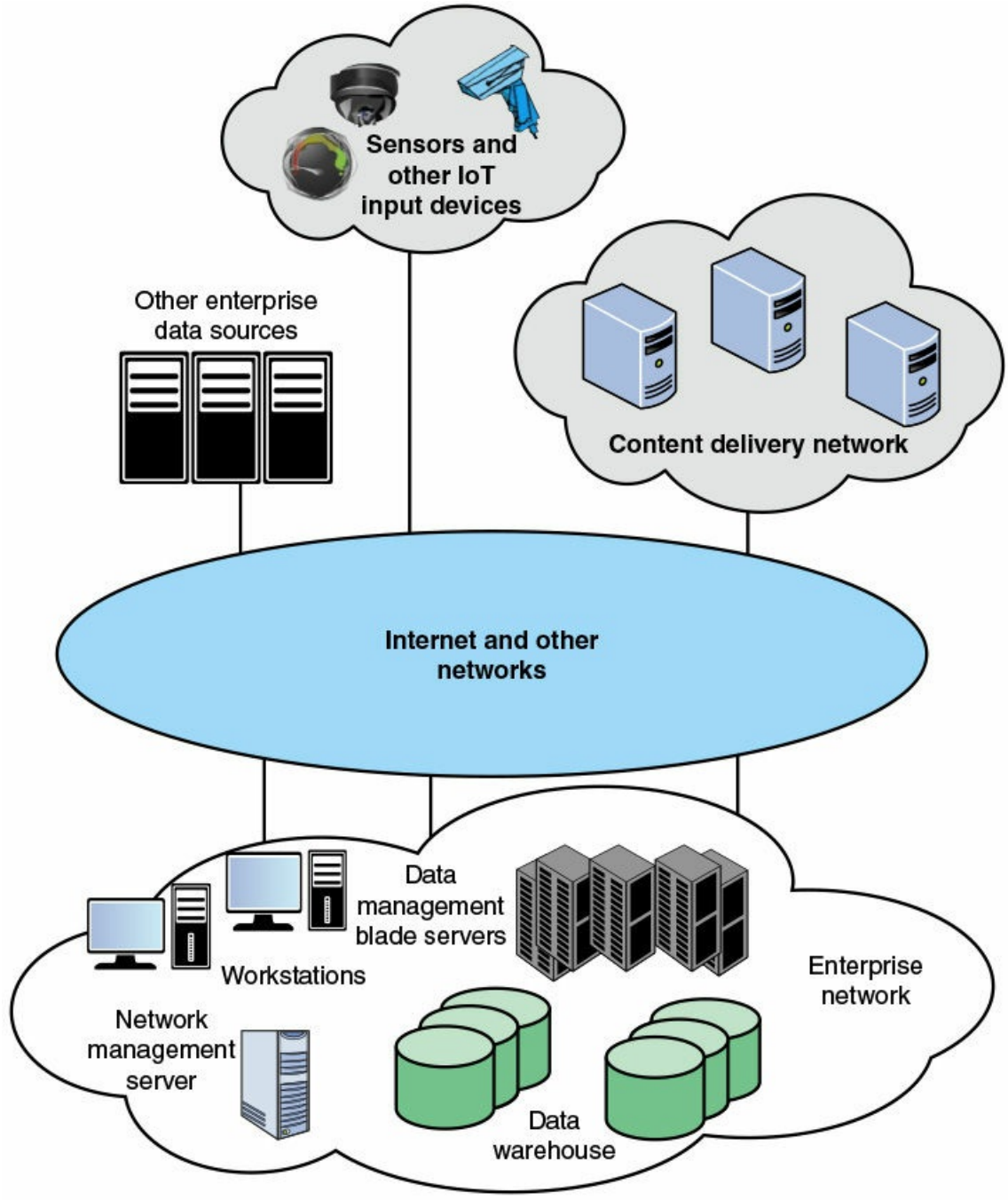


FIGURE 2.3 Big Data Networking Ecosystem

Key elements within the enterprise include the following:

- **Data warehouse:** The DW holds integrated data from multiple data sources, used for reporting and data analysis.
- **Data management servers:** Large banks of servers serve multiple functions with respect to big data. The servers run data analysis applications, such as data integration tools and analytics tools. Other applications integrate and structure data from enterprise activity, such as financial data, point-of-sale data, and e-commerce activity.

- **Workstations / data processing systems:** Other systems involved in the use of big data applications and in the generation of input to big data warehouses.
- **Network management server:** One or more servers responsible for network management, control, and monitoring.

Not shown in Figure 2.3 are other important network devices, including firewalls, intrusion detection/prevention systems (IDS/IPS), LAN switches, and routers.

The enterprise network can involve multiple sites distributed regionally, nationally, or globally. In addition, depending on the nature of the big data system, an enterprise can receive data from other enterprise servers, from dispersed sensors and other devices in an Internet of Things, in addition to multimedia content from content delivery networks.

The networking environment for big data is complex. The impact of big data on an enterprise's networking infrastructure is driven by the so-called three *V*'s:

- Volume (growing amounts of data)
- Velocity (increasing speed in storing and reading data)
- Variability (growing number of data types and sources)

Based on a Network World 2014 white paper, areas of concern include the following [NETW14]:

- **Network capacity:** Running big data analytics requires a lot of capacity on its own; the issue is magnified when big data and day-to-day application traffic are combined over an enterprise network.
- **Latency:** The real or near-real-time nature of big data demands a network architecture with consistent low latency to achieve optimal performance.
- **Storage capacity:** Massive amounts of highly scalable storage are required to address the insatiable appetite of big data, yet these resources must be flexible enough to handle many different data formats and traffic loads.
- **Processing:** Big data can add significant pressure on computational, memory, and storage systems, which, if not properly addressed, can negatively impact operational efficiency.
- **Secure data access:** Big data projects combine sensitive information from many sources like customer transactions, GPS coordinates, video streams, and so on, which must be protected from unauthorized access.

Cloud Computing

As with big data installations, cloud computing presents imposing challenges for effective and efficient flow of traffic through networks. It will be helpful in this regard to consider the cloud network model developed by ITU-T, and shown in Figure 2.4 [ITUT12]. This figure indicates the scope of network concerns for cloud network and service providers and for cloud service users.

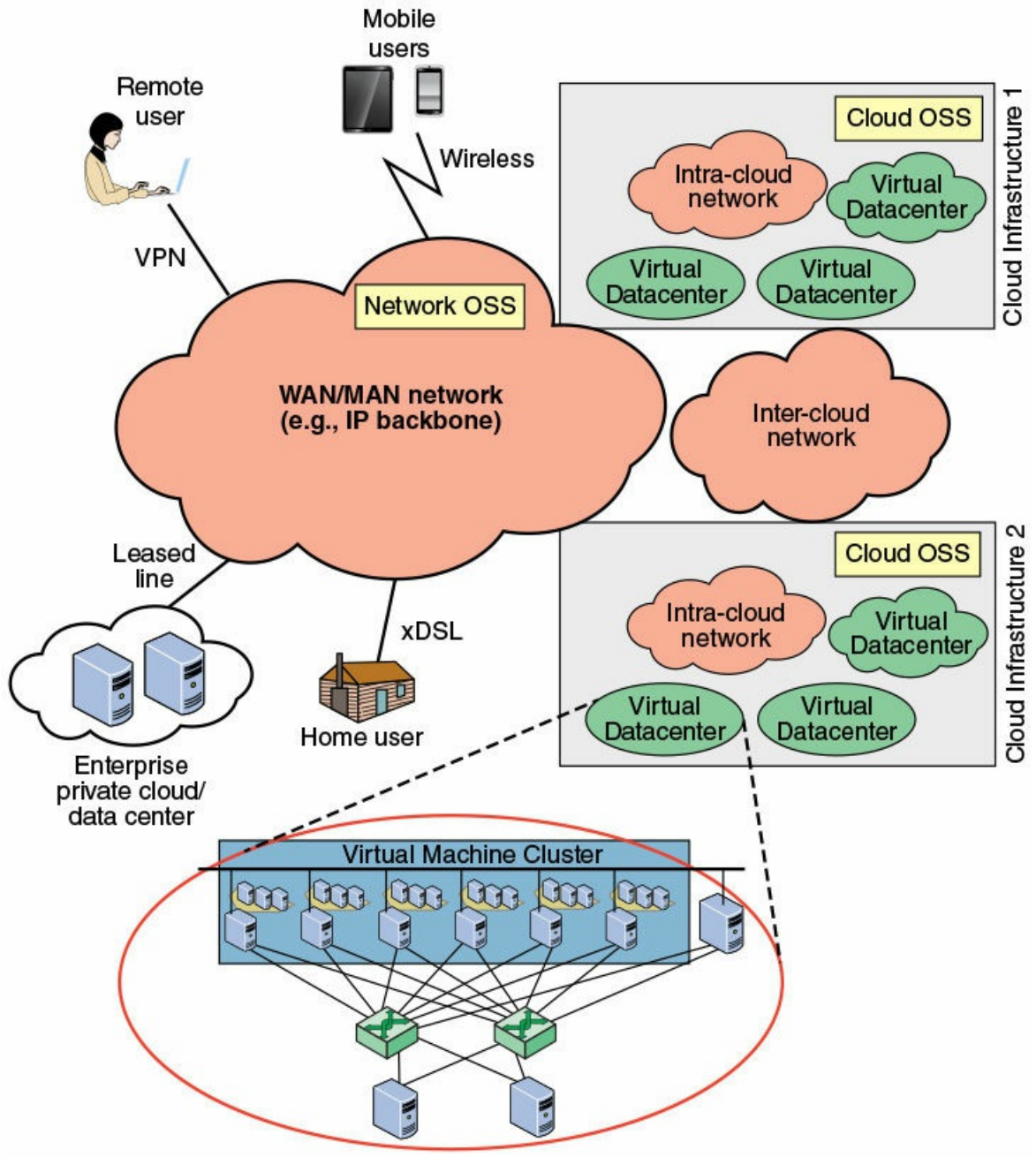


FIGURE 2.4 Cloud Network Model

A cloud service provider maintains one or more local or regional cloud infrastructures. An intracloud network connects the elements of the infrastructure, including database servers, storage arrays, and other servers (for example, firewalls, load balancers, application acceleration devices, and IDS/IPS). The intracloud network will likely include a number of LANs interconnected with IP routers. Within the infrastructure, database servers are organized as a cluster of **virtual machines**, providing virtualized, isolated computing environments for different users.

➔ *See Chapter 7, "Network Functions Virtualization: Concepts and Architecture"*

Intercloud networks interconnect cloud infrastructures together. These cloud infrastructures may be owned by the same cloud provider or by different ones. Finally, a core transport network is used by customers to access and consume cloud services deployed within the cloud provider's data center.

Also depicted in Figure 2.4 are two categories of operations support system (OSS):

- **Network OSS:** The traditional OSS is a system dedicated to providers of telecommunication services. The processes supported by a network OSS include service management and maintenance of the network inventory, configuration of particular network components, and fault management.
- **Cloud OSS:** OSS of cloud infrastructure is the system dedicated to providers of cloud computing services. Cloud OSS supports processes for the maintenance, monitoring, and configuration of cloud resources.

These three network components (intracloud, intercloud, core), together with the OSS components, are the foundation of cloud services composition and delivery. The ITU-T Focus Group on Cloud Computing Technical Report [ITUT12] lists the following functional requirements for this network capability:

- **Scalability:** Networks must be able to scale easily to meet the demands of moving from current cloud infrastructures of hundreds or a few thousand servers to networks of tens or even hundreds of thousands of servers. This scaling presents challenges in areas such as addressing, routing, and congestion control.
- **Performance:** Traffic in both big data installations and cloud provider networks is unpredictable and quite variable [KAND12]. There are sustained spikes between nearby servers within the same rack and intermittent heavy traffic with a single source server and multiple destination servers. Intracloud networks need to provide reliable high-speed direct (logical point-to-point) communications between servers with congestion-free links, and uniform capacity between any two arbitrary servers within the data center. The ITU-T report concludes that the current three-tier topology (access, aggregation, and core) used in data centers is not well adapted to provide these requirements. A more flexible and dynamic control of data flows, in addition to virtualization of network devices, provides a better foundation for providing the desired quality of service.
- **Agility and flexibility**: The cloud-based data center needs to be able to respond and manage the highly dynamic nature of cloud resource utilization. This includes the ability to adapt to virtual machine mobility and to provide fine-grained control of flows routing through the data center.

➔ *See Chapter 13, "Cloud Computing"*

We return to this discussion in Chapter 13, "Cloud Computing." For now, it suffices to point out that as the book unfolds, it should be clear that the combination of SDN and NFV are well suited to meeting the requirements in the preceding list.

Mobile Traffic

Technical innovations have contributed to the success of what were originally just mobile

phones. The prevalence of the latest devices, with multimegabit Internet access, mobile apps, high megapixel digital cameras, access to multiple types of wireless networks (for example, Wi-Fi, Bluetooth, 3G, 4G), and several onboard sensors, all add to this momentous achievement. Devices have become increasingly powerful while staying easy to carry. Battery life has increased (even though device energy usage has also expanded), and digital technology has improved reception and allowed better use of a finite spectrum. As with many types of digital equipment, the costs associated with mobile devices have been decreasing.

The first rush to wireless was for voice. Now, the attention is on data; some wireless devices are only rarely used for voice. Figure 2.5 shows the dramatic trend in world total mobile traffic in 2G, 3G, and 4G networks (not including DVB-H, Wi-Fi, and Mobile WiMAX) estimated by Ericsson [AKAM15]. Ericsson's presence in more than 180 countries and its customer base representing more than 1000 networks enable it to measure mobile voice and data volumes. The result is a representative base for calculating world total mobile traffic.

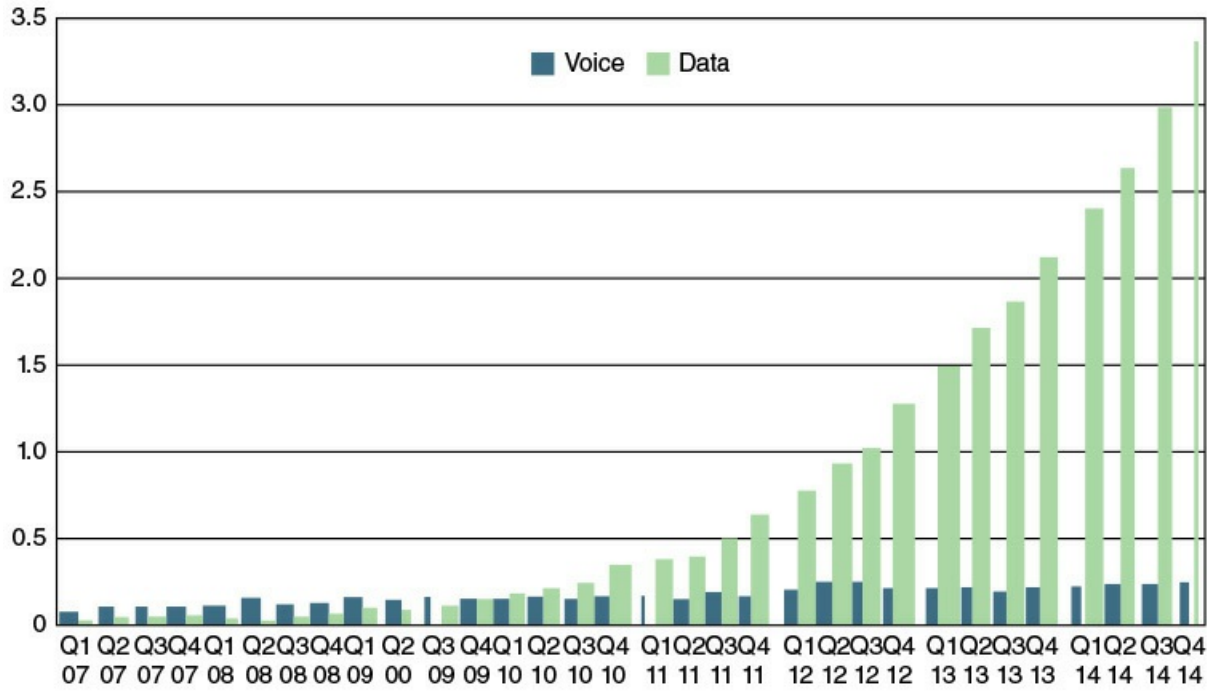


FIGURE 2.5 World Total Monthly Mobile Voice and Data Traffic (exabytes/month) [AKAM15]

A big part of the mobile market is the wireless Internet. Wireless users use the Internet differently than fixed users, but in many ways no less effectively. Wireless smartphones have limited displays and input capabilities compared with larger devices such as laptops or PCs, but mobile apps give quick access to intended information without using websites. Because wireless devices are location aware, information can be tailored to the geographic location of the user. Information finds users, instead of users searching for information. Tablet devices provide a happy medium between the larger screens and better input capabilities of PCs and the portability of smartphones.

Figure 2.6 shows a projection for mobile enterprise IP traffic [CISC14], where the term *enterprise* refers to businesses and governments. Cisco's methodology rests on a combination of

analyst projections, in-house estimates and forecasts, and direct data collection. As with mobile data traffic over cellular networks, mobile enterprise IP traffic is on a strong growth curve.

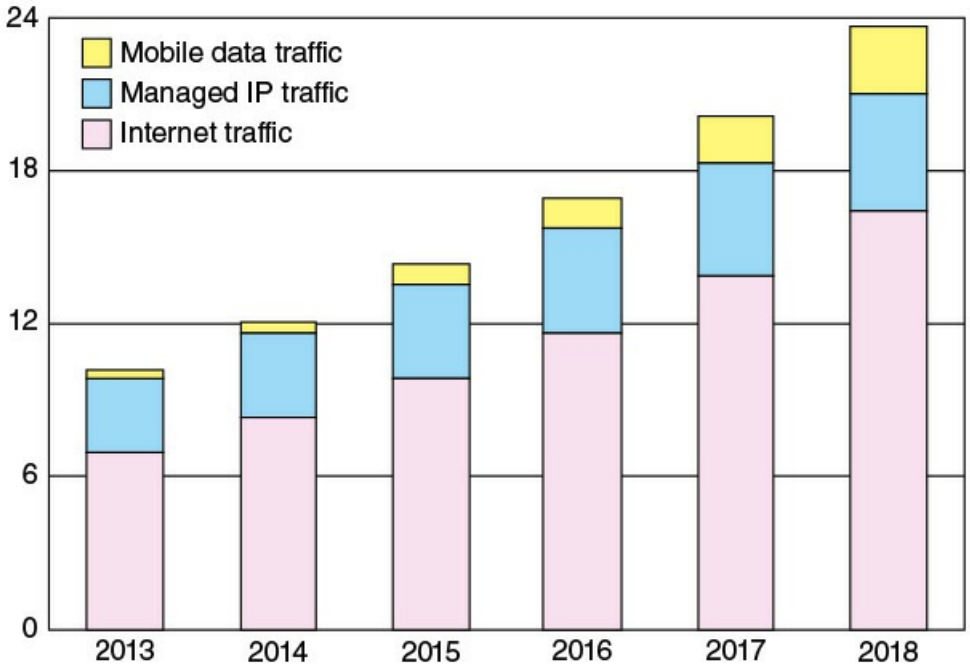


FIGURE 2.6 Forecast Monthly Enterprise IP Traffic (Exabytes/Month) [CISC14]

Figure 2.6 breaks the mobile traffic down into three categories:

- **Mobile data traffic:** All enterprise traffic that crosses a mobile access point
- **Managed IP traffic:** All enterprise traffic that is transported over IP but remains within the corporate WAN
- **Internet traffic:** All enterprise traffic that crosses the public Internet

Although mobile traffic is the smallest of the three categories of enterprise traffic, it is growing much more rapidly than the other two categories. Based on Cisco's projections, the compound annual growth rate over the period 2013 to 2018 for enterprise traffic is as follows:

Mobile Data Traffic	Managed IP Traffic	Internet Traffic	Total Traffic
55 percent	10 percent	18 percent	18 percent

Enterprise networks need to be flexible enough to handle the rapidly growing mobile data load. Such a load is characterized by dynamically changing physical access points into the network and a wide variety of elastic and inelastic traffic types. As you will learn, SDN and NFV are well suited to coping with this highly dynamic load.

2.3 Requirements: QoS and QoE

So far, this chapter has focused on the types of traffic the enterprise networks and the Internet carry and looked at three areas of demand that create significant challenges for providing effective and efficient network service to users. This section briefly introduces two concepts that provide a way of quantifying the network performance that the enterprise desires to achieve:

quality of service (QoS) and quality of experience (QoE). QoS and QoE enable the network manager to determine whether the network is meeting user needs and to diagnose problem areas that require adjustment to network management and network traffic control. QoS and QoE are treated in detail in Part IV of the book.

➔ *See Part IV, "Defining and Supporting User Needs"*

Quality of Service

You can define QoS as the measurable end-to-end performance properties of a network service, which can be guaranteed in advance by a service level agreement (SLA) between a user and a service provider, so as to satisfy specific customer application requirements. Commonly specified properties include the following:

- **Throughput**: A minimum or average throughput, in bytes per second or bits per second, for a given logical connection or traffic flow.
- **Delay:** The average or maximum delay. Also called latency.
- **Packet jitter:** Typically, the maximum allowable jitter.
- **Error rate:** Typically maximum error rate, in terms of fraction of bits delivered in error.
- **Packet loss:** Fraction of packets lost.
- **Priority:** A network may offer a given number of levels of priority. The assigned level for various traffic flows influences the way in which the different flows are handled by the network.
- **Availability:** Expressed as a percentage of time available.
- **Security:** Different levels or types of security may be defined.

QoS mechanisms ensure that business applications continue to receive the necessary performance guarantee even though they no longer run on dedicated hardware, such as when applications are transferred to a cloud. The QoS provided by an infrastructure is partially determined by its overall performance and efficiency. However, QoS is also the ability to prioritize specific workloads and allocate the needed resources to meet required service levels. It can offer a powerful way to allocate processor, memory, I/O, and network traffic resources among applications and virtual guests.

Quality of Experience

QoE is a subjective measure of performance as reported by the user. Unlike QoS, which can be precisely measured, QoE relies on human opinion. QoE is important particularly when we deal with multimedia applications and multimedia content delivery. QoS provides measurable, quantitative targets that guide the design and operation of a network and enable customer and provider to agree on what quantitative performance the network will deliver for give applications and traffic flows.

However, QoS processes by themselves are not sufficient in that they do not take into account

the user's perception of network performance and service quality. Although the maximum capacity may be fixed at a certain value by a media transmission system, this does not necessarily fix the quality of the multimedia content at, say, "high." This is because there are numerous ways the multimedia content could have been encoded, giving rise to differing perceived qualities. The ultimate measure of a network and the services it offers is how subscribers perceive the performance. QoE augments the traditional QoS by providing information regarding the delivered services from an end user point of view.

There is a wide range of factors and features that can be included in a requirement for QoE, which can, roughly, be classified into the following categories:

- **Perceptual:** This category encompasses the quality of the sensory aspects of the user experience. For video, examples include sharpness, brightness, contrast, flicker, and distortion. Audio examples include clarity and timbre.
- **Psychological:** This category deals with the user's feeling about the experience. Examples include ease of use, joy of use, usefulness, perceived quality, satisfaction, annoyance, and boredom.
- **Interactive:** This category deals with aspects of an experience related to the interaction between the user and the application or device, such as responsiveness, naturalness of interaction, communication efficiency, and accessibility.

For practical application, these features need to be converted to quantitative measures.

The management of QoE has become a crucial concept in the deployment of future successful applications, services, and products. The greatest challenges in providing QoE are developing effective methods for converting QoE features to quantitative measures and translating QoE measures to QoS measures. Whereas QoS can now easily be measured, monitored, and controlled at both the networking and application layers, and at both the end system and network sides, QoE is something that is still quite intricate to manage.

2.4 Routing

This section and the next briefly introduce two mechanisms that are fundamental to the operation of a network and its capability to transmit and deliver packet traffic: routing and congestion control. A detailed look is beyond the scope of the book. The purpose here is to indicate the basic concepts of routing and congestion control, because these are the basic tools needed to support network traffic and to provide QoS and QoE.

Characteristics

The primary function of an internet is to accept packets from a source station and deliver them to a destination station. To accomplish this, a path or route through the network must be determined; generally, more than one route is possible. Therefore, a routing function must be performed.

The selection of a route is generally based on some performance criterion. The simplest criterion is to choose the minimum-hop route (one that passes through the least number of nodes) through

the network. This is an easily measured criterion and should minimize the consumption of network resources. A generalization of the minimum-hop criterion is least-cost routing. In this case, a cost is associated with each link, and, for any pair of attached stations, the route through the network that accumulates the least cost is sought.

Figure 2.7 illustrates a network in which the two arrowed lines between a pair of nodes represent a link between these nodes, and the corresponding numbers represent the current link cost in each direction. Our concern, of course, is with an internet, in which each node is a router and the links between adjacent routers are networks or direct communications links. The shortest path (fewest hops) from node 1 to node 6 is 1-3-6 (Cost = 5 + 5 = 10), but the least-cost path is 1-4-5-6 (Cost = 1 + 1 + 2 = 4).

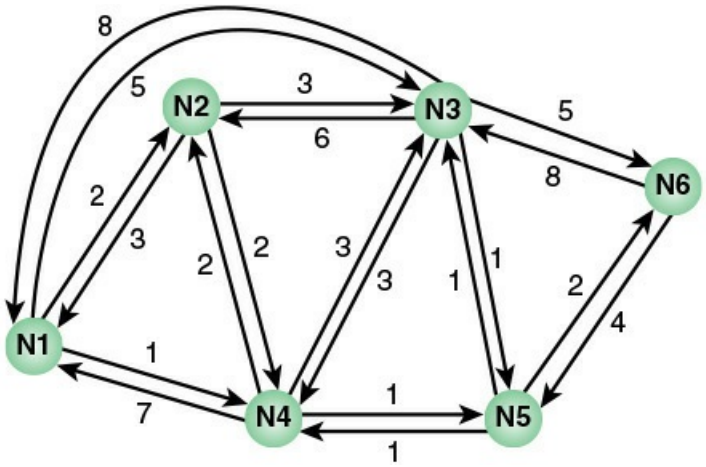


FIGURE 2.7 Network Architecture Example

Costs are assigned to links to support one or more design objectives. For example, the cost could be inversely related to the data rate (that is, the higher the data rate on a link, the lower the assigned cost of the link) or the current link delay. In the first case, the least-cost route should provide the highest throughput. In the second case, the least-cost route should minimize delay. Routing decisions can be based on other criteria as well. For example, a routing policy may dictate that certain types of traffic be restricted to certain routes for security concerns.

Packet Forwarding

The key function of any router is to accept incoming packets and forward them. For this purpose, a router maintains forwarding tables. Figure 2.8 shows a simplified example of how this might be implemented for the network, with its associated link costs, of Figure 2.7. A router's forwarding table shows, for each destination, the identity of the next node on the router. Each router may be responsible for discovering the appropriate routes. Alternatively, a network control center may be responsible for designing routes for all routers and maintaining a central forwarding table, providing each router with individual forwarding tables relevant only to that router.

CENTRAL FORWARDING TABLE

To Node \ From Node	1	2	3	4	5	6
1	–	1	5	2	4	5
2	2	–	5	2	4	5
3	4	3	–	5	3	5
4	4	4	5	–	4	5
5	4	4	5	5	–	5
6	4	4	5	5	6	–

Node 1 Table

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Table

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Table

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Table

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Table

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6 Table

Destination	Next Node
``	5
2	5
3	5
4	5
5	5

FIGURE 2.8 Packet Forwarding Tables (using Figure 2.7)

Note that it is not necessary to store the complete route for each possible pair of nodes. Rather, it is sufficient to know, for each pair of nodes, the identity of the first node on the route.

In the simple example of Figure 2.8, forwarding decisions are based solely on the identity of the destination system. Additional information is often used to determine the forwarding decision, such as the source address, packet flow identifier, or security level of the packet:

- **Failure:** When a node or link fails, it can no longer be used as part of a route.
- **Congestion:** When a particular portion of the network is heavily congested, it is desirable to route packets around rather than through the area of congestion.
- **Topology change**: The insertion of new links or nodes affects routing.

For adaptive routing to be possible, information about the state of the network must be exchanged among the nodes or between the nodes and a central controller.

Routing Protocols

The routers in an internet are responsible for receiving and forwarding packets through the interconnected set of networks. Each router makes routing decisions based on knowledge of the topology and traffic/delay conditions of the internet. Accordingly, a degree of dynamic cooperation is needed among the routers. In particular, the router must avoid portions of the network that have failed and should avoid portions of the network that are congested. To make such dynamic routing decisions, routers exchange routing information using a routing protocol. Information is needed about the status of the internet, in terms of which networks can be reached by which routes, and the delay characteristics of various routes.

There are essentially two categories of routing protocols, which are based on the concept of an **autonomous system (AS)**. We first define AS and then look at the two categories. An AS exhibits the following characteristics:

1. An AS is a set of routers and networks managed by a single organization.
2. An AS consists of a group of routers exchanging information via a common routing protocol.
3. Except in times of failure, an AS is connected (in a graph-theoretic sense); that is, there is a path between any pair of nodes.

A shared routing protocol, called here an **interior router protocol (IRP)**, passes routing information between routers within an AS. The protocol used within the AS does not need to be implemented outside of the system. This flexibility allows IRPs to be custom tailored to specific applications and requirements.

It may happen, however, that an internet will be constructed of more than one AS. For example, all the LANs at a site, such as an office complex or campus, could be linked by routers to form an AS. This system might be linked through a wide-area network to other autonomous systems. Figure 2.9 illustrates this situation.

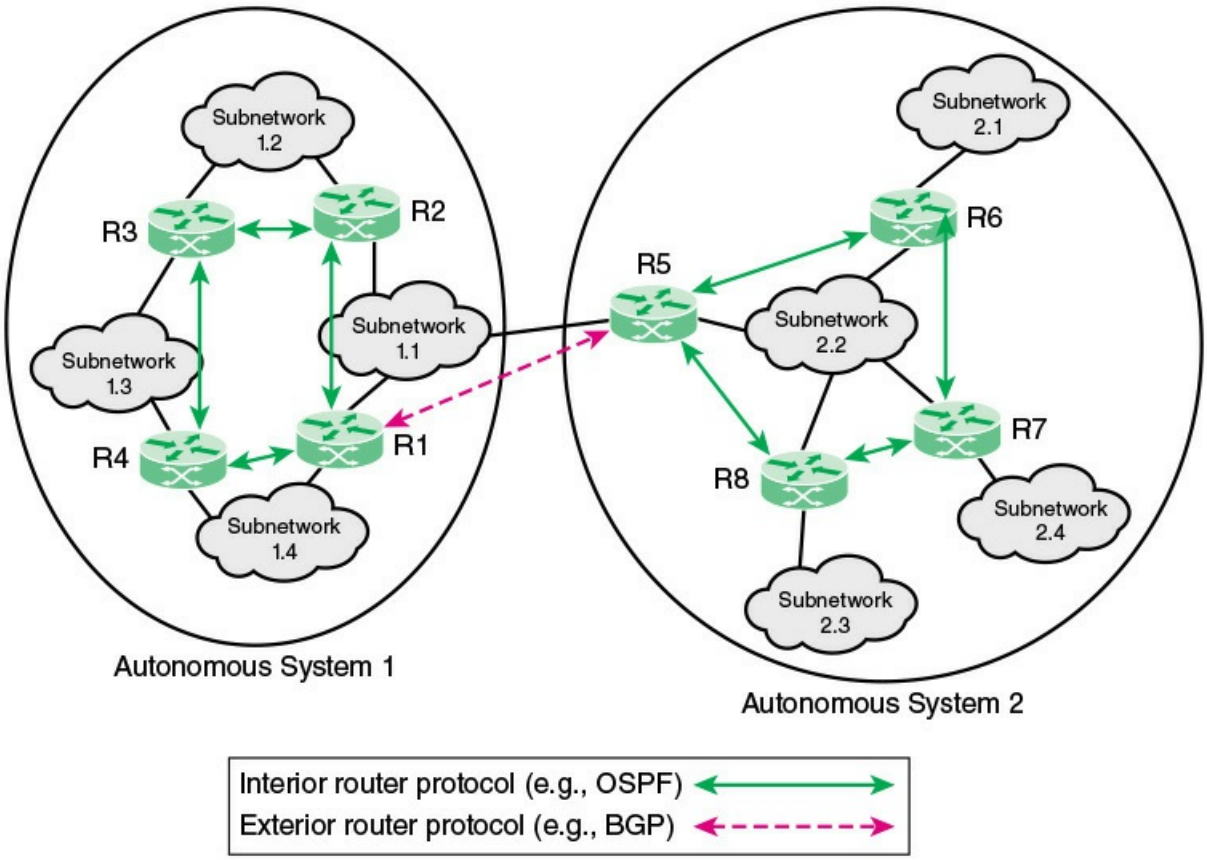


FIGURE 2.9 Use of Exterior and Interior Routing Protocols

In this case, the routing algorithms and information in routing tables used by routers in different autonomous systems may differ. Nevertheless, the routers in one AS need at least a minimal level of information concerning networks outside the system that can be reached. We refer to the protocol used to pass routing information between routers in different autonomous systems as an **exterior router protocol (ERP)**.

Note

In the literature, the terms *interior gateway protocol (IGP)* and *exterior gateway protocol (EGP)* are often used for what are referred to here as IRP and ERP. However, because the terms *IGP* and *EGP* also refer to specific protocols, we avoid their use when referring to the general concepts.

You can expect that an ERP will need to pass less information than an IRP for the following reason. If a packet is to be transferred from a host in one AS to a host in another AS, a router in the first system need only determine the target AS and devise a route to get into that target system. Once the packet enters the target AS, the routers within that system can cooperate to deliver the packet; the ERP is not concerned with, and does not know about, the details of the route followed within the target AS.

Elements of a Router

Figure 2.10 depicts the principal elements of a router, from the point of view of its routing function.

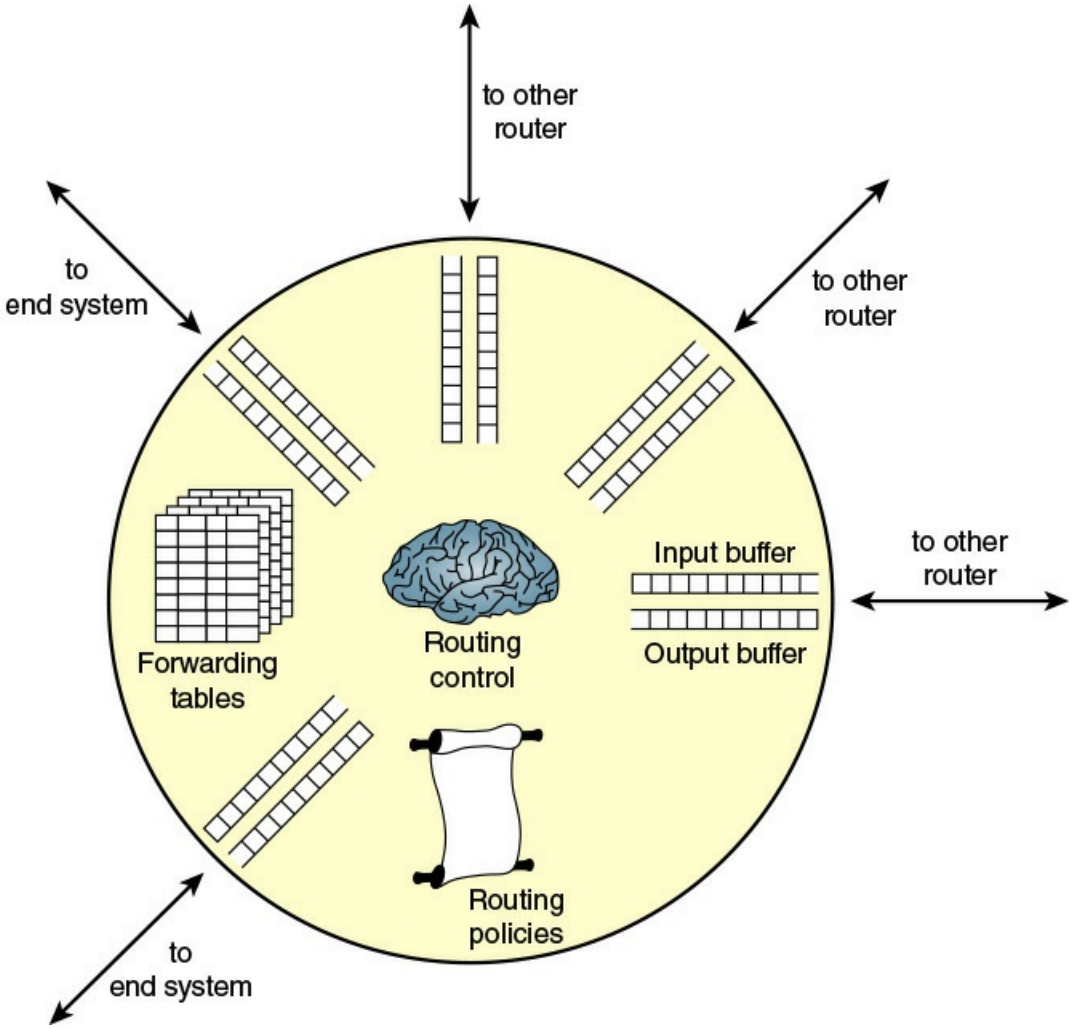


FIGURE 2.10 Elements of a Router

Any given router has a number of I/O ports attached to it: one or more to other routers, and zero or more to end systems. On each port, packets arrive and depart. You can consider that there are two buffers, or queues, at each port: one to accept arriving packets, and one to hold packets that are waiting to depart. In practice, there might be two fixed-size buffers associated with each port, or there might be a pool of memory available for all buffering activities. In the latter case, you can think of each port having two variable-size buffers associated with it, subject to the constraint that the sum of all buffer sizes is a constant.

In any case, as packets arrive, they are stored in the input buffer of the corresponding port. The router examines each incoming packet, makes a routing decision based on the forwarding tables, and then moves the packet to the appropriate output buffer. Packets queued for output are transmitted as rapidly as possible. Each output queue can be operated as a simple first-in, first-

out (FIFO) queue. More commonly, a more complex queuing discipline is used, to take into account the relative priority of the queued packets. A set of routing policies may also influence the construction of the forwarding tables and how various packets are to be treated. Policies may determine routing not just on the destination address but other factors, such as source address, packet size, and protocol of the payload.

The final element shown in Figure 2.10 is a routing control function. This function includes execution of routing protocols, adaptive maintenance of the routing tables, and supervising congestion control policies.

2.5 Congestion Control

If the traffic demand on an internet exceeds capacity, or if the internet does not manage the traffic efficiently, congestion will occur. This section provides a brief overview of the effects of congestion and a general introduction to approaches to congestion control.

Effects of Congestion

If packets arrive too fast for a router to process them (that is, make routing decisions) or faster than packets can be cleared from the outgoing buffers, eventually packets will arrive for which no memory is available. When such a saturation point is reached, one of two general strategies can be adopted. The first such strategy is to discard any incoming packet for which there is no available buffer space. The alternative is for the node that is experiencing these problems to exercise some sort of flow control over its neighbors so that the traffic flow remains manageable. But, as Figure 2.11 illustrates, each of a node's neighbors is also managing a number of queues. If node 6 restrains the flow of packets from node 5, this causes the output buffer in node 5 for the port to node 6 to fill up. Thus, congestion at one point in the network can quickly propagate throughout a region or the entire network. Although flow control is indeed a powerful tool, you need to use it in such a way as to manage the traffic on the entire network.

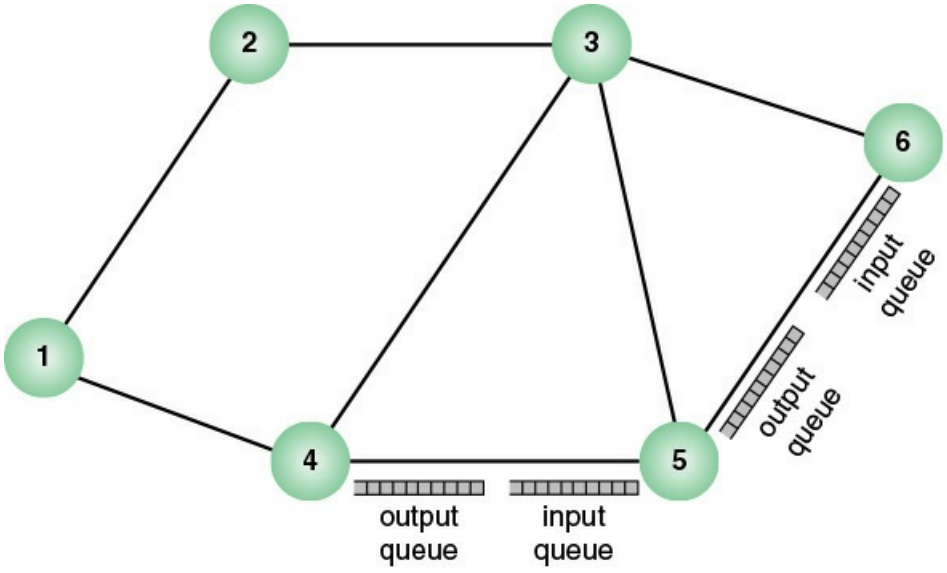


FIGURE 2.11 Interaction of Queues in a Data Network

Ideal Performance

Figure 2.12 suggests the ideal goal for network utilization.

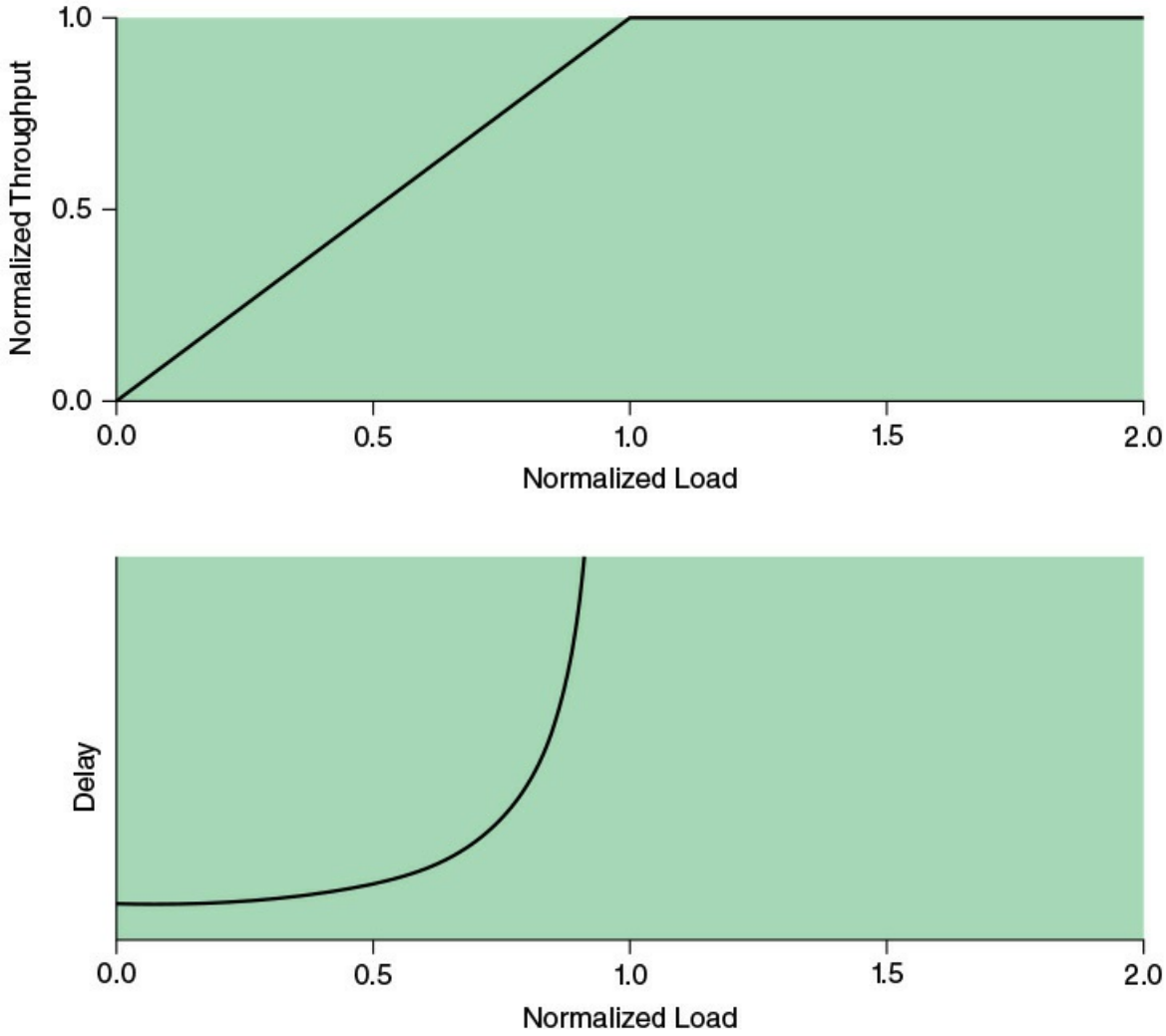


FIGURE 2.12 Ideal Network Utilization

The top graph plots the steady-state total throughput (number of packets delivered to destination end systems) through the network as a function of the offered load (number of packets transmitted by source end systems), both normalized to the maximum theoretical throughput of the network. In the ideal case, the throughput of the network increases to accommodate load up to an offered load equal to the full capacity of the network; then normalized throughput remains at 1.0 at higher input loads. Note, however, what happens to the end-to-end delay experienced by the average packet even with this assumption of ideal performance. At negligible load, there is some small constant amount of delay that consists of the propagation delay through the network from source to destination plus processing delay at each node. As the load on the network increases, queuing delays at each node are added to this fixed amount of delay. The reason for the increase in delay even when the total network capacity is not exceeded has to do with the

variability in load at each node. With multiple sources supplying data to the network, even if each source produced packets at fixed intervals, there will be fluctuation in the input rate at each individual network node. When a burst of packets arrives at a node, it will take some time to clear the backlog. As it is clearing the backlog, it is sending out a sustained burst of packets, thus imposing packet bursts on downstream node. And once a queue builds up at a node, even if packets only arrive at a rate the node can handle during a given time period, those packets have to wait their turn in the queue, and thus experience additional delay. This is a standard result of queuing theory: delays will grow with increasing load if the arrival rate is not constant.

When the load exceeds the network capacity, delays increase without bound. Here is a simple intuitive explanation of why delay must go to infinity. Suppose that each node in the network is equipped with buffers of infinite size and suppose that the input load exceeds network capacity. Under ideal conditions, the network will continue to sustain a normalized throughput of 1.0. Therefore, the rate of packets leaving the network is 1.0. Because the rate of packets entering the network is greater than 1.0, internal queue sizes grow. In the steady state, with input greater than output, these queue sizes grow without bound and therefore queuing delays grow without bound.

It is important to grasp the meaning of Figure 2.12 before looking at real-world conditions. This figure represents the ideal, but unattainable, goal of all traffic and congestion control schemes. No scheme can exceed the performance depicted in Figure 2.12.

Practical Performance

The ideal case reflected in Figure 2.12 assumes infinite buffers and no overhead related to congestion control. In practice, buffers are finite, leading to buffer overflow, and attempts to control congestion consume network capacity in the exchange of control signals.

Consider what happens in a network with finite buffers if no attempt is made to control congestion or to restrain input from end systems. The details, of course, differ depending on network architecture and on the statistics of the presented traffic; however, the graphs in Figure 2.13 depict the devastating outcome in general terms.

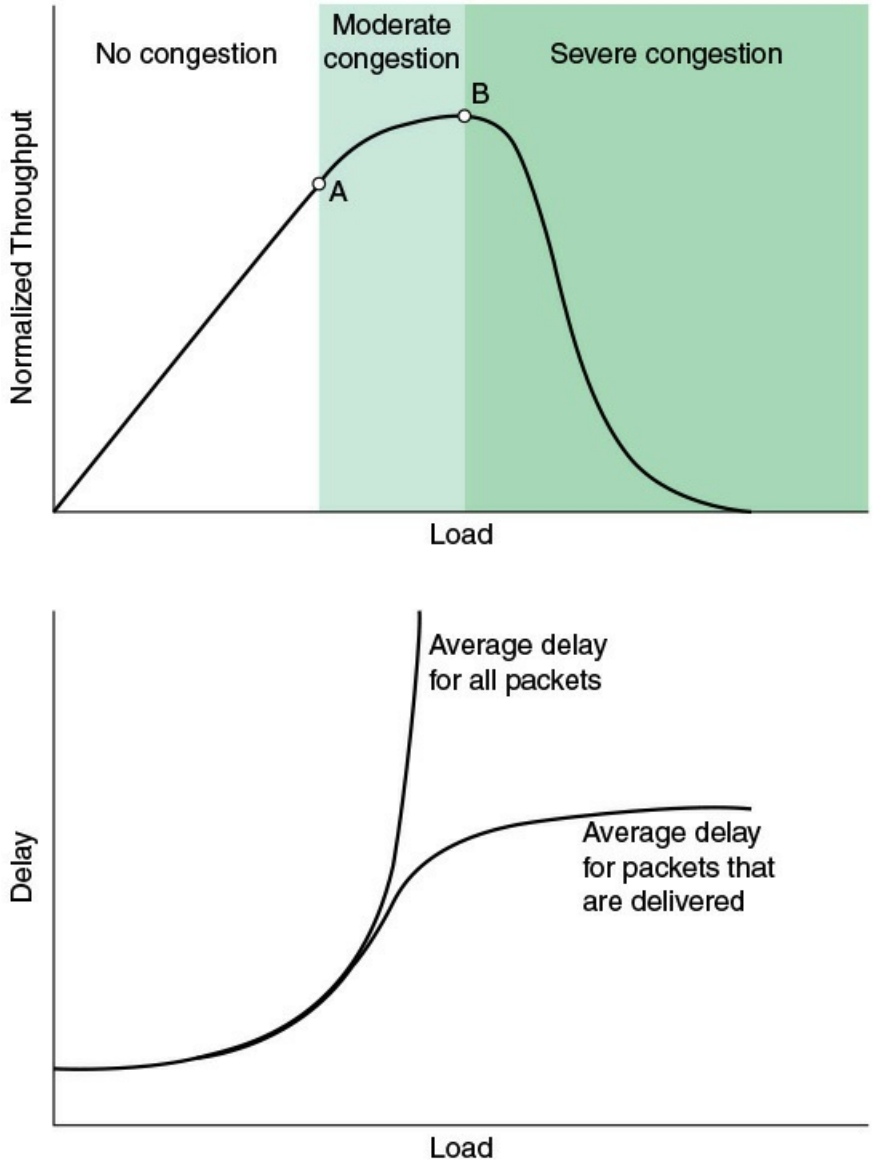


FIGURE 2.13 The Effects of Congestion

At light loads, throughput and hence network utilization increases as the offered load increases. As the load continues to increase, a point is reached (point A in the plot) beyond which the throughput of the network increases at a rate slower than the rate at which offered load is increased. This is because of network entry into a moderate congestion state. In this region, the network continues to cope with the load, although with increased delays. The departure of throughput from the ideal is accounted for by a number of factors. For one thing, the load is unlikely to be spread uniformly throughout the network. Therefore, while some nodes may experience moderate congestion, others may be experiencing severe congestion and may need to discard traffic. In addition, as the load increases, the network attempts to balance the load by routing packets through areas of lower congestion. For the routing function to work, an increased number of routing messages must be exchanged between nodes to alert each other to areas of congestion; this overhead reduces the capacity available for data packets.

As the load on the network continues to increase, the queue lengths of the various nodes continue to grow. Eventually, a point is reached (point B in the plot) beyond which throughput actually drops with increased offered load. The reason for this is that the buffers at each node are of finite size. When the buffers at a node become full, the node must discard packets. Therefore, the sources must retransmit the discarded packets in addition to new packets. This only exacerbates the situation: As more and more packets are retransmitted, the load on the system grows, and more buffers become saturated. While the system is trying desperately to clear the backlog, users are pumping old and new packets into the system. Even successfully delivered packets may be retransmitted because it takes too long, at a higher layer (for example, transport layer), to acknowledge them: The sender assumes the packet did not get through and retransmits. Under these circumstances, the effective capacity of the system declines to zero.

Congestion Control Techniques

Figure 2.14 provides a general depiction of important congestion control techniques. This section examines each of these.

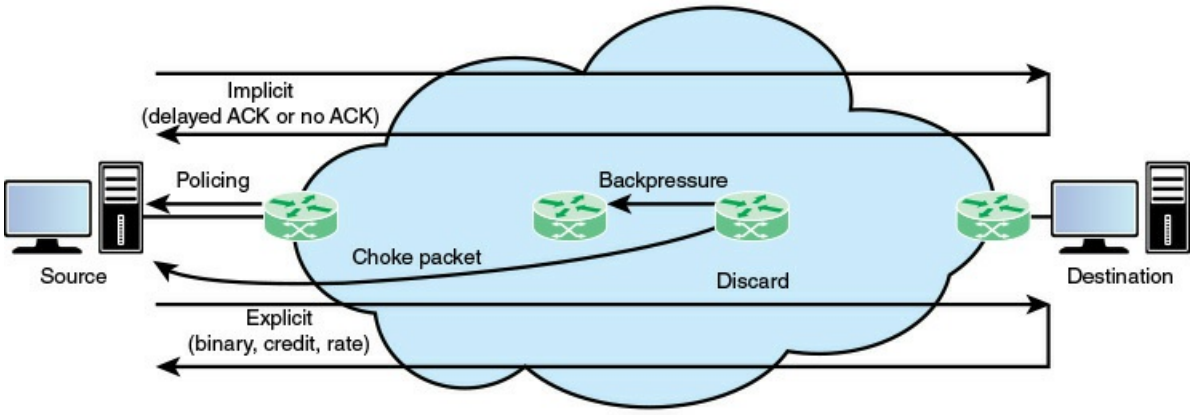


FIGURE 2.14 Mechanisms for Congestion Control

Backpressure

Backpressure can be exerted on the basis of links or logical connections (for example, virtual circuits). Referring again to Figure 2.11, if node 6 becomes congested (buffers fill up), node 6 can slow down or halt the flow of all packets from node 5 (or node 3, or both nodes 5 and 3). If this restriction persists, node 5 will need to slow down or halt traffic on its incoming links. This flow restriction propagates backward (against the flow of data traffic) to sources, which are restricted in the flow of new packets into the network.

Backpressure for all traffic on a particular link is automatically invoked by the flow control mechanisms of data link layer protocols. Backpressure can also be selectively applied to logical connections, so that the flow from one node to the next is only restricted or halted on some connections, generally the ones with the most traffic. In this case, the restriction propagates back along the connection to the source. Such a mechanism is used in Frame Relay and Asynchronous Transfer Mode (ATM) networks. However, the use of these networks has declined considerably in favor of Ethernet carrier networks and IP-based Multiprotocol Label Switching (MPLS)

networks.

Choke Packet

A choke packet is a control packet generated at a congested node and transmitted back to a source node to restrict traffic flow. Either a router or a destination end system may send this message to a source end system, requesting that it reduce the rate at which it is sending traffic to the internet destination. On receipt of a choke packet, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives choke packets. The choke packet can be used by a router or host that must discard IP datagrams because of a full buffer. In that case, the router or host will issue a choke packet for every packet that it discards. In addition, a system may anticipate congestion and issue choke packets when its buffers approach capacity. In that case, the packet referred to in the choke packet may well be delivered. Therefore, receipt of a choke packet does not imply delivery or nondelivery of the corresponding packet.

Implicit Congestion Signaling

When network congestion occurs, two things may happen:

1. The transmission delay for an individual packet from source to destination increases, so that it is noticeably longer than the fixed propagation delay, and
2. Packets are discarded.

If a source can detect increased delays and packet discards, it has implicit evidence of network congestion. If all sources can detect congestion and, in response, reduce flow on the basis of congestion, the network congestion will be relieved. Therefore, congestion control on the basis of implicit signaling is the responsibility of end systems and does not require action on the part of network nodes.

Implicit signaling is an effective congestion control technique in connectionless, or datagram, networks, such as IP-based internets. In such cases, there are no logical connections through the internet on which flow can be regulated. However, between the two end systems, logical connections can be established at the TCP level. TCP includes mechanisms for acknowledging receipt of TCP segments and for regulating the flow of data between source and destination on a TCP connection.

Explicit Congestion Signaling

It is desirable to use as much of the available capacity in a network as possible but still react to congestion in a controlled and fair manner. This is the purpose of explicit congestion avoidance techniques. In general terms, for explicit congestion avoidance, the network alerts end systems to growing congestion within the network and the end systems take steps to reduce the offered load to the network.

Explicit congestion signaling approaches can work in one of two directions:

- **Backward:** Notifies the source that congestion avoidance procedures should be initiated

where applicable for traffic in the opposite direction of the received notification. It indicates that the packets that the user transmits on this logical connection may encounter congested resources. Backward information is transmitted either by altering bits in a header of a data packet headed for the source to be controlled or by transmitting separate control packets to the source.

- **Forward**: Notifies the user that congestion avoidance procedures should be initiated where applicable for traffic in the same direction as the received notification. It indicates that this packet, on this logical connection, has encountered congested resources. Again, this information may be transmitted either as altered bits in data packets or in separate control packets. In some schemes, when a forward signal is received by an end system, it echoes the signal back along the logical connection to the source. In other schemes, the end system is expected to exercise flow control upon the source end system at a higher layer (for example, TCP).

You can divide explicit congestion signaling approaches into three general categories:

- **Binary:** A bit is set in a data packet as it is forwarded by the congested node. When a source receives a binary indication of congestion on a logical connection, it may reduce its traffic flow.
- **Credit based:** These schemes are based on providing an explicit credit to a source over a logical connection. The credit indicates how many octets or how many packets the source may transmit. When the credit is exhausted, the source must await additional credit before sending additional data. Credit-based schemes are common for end-to-end flow control, in which a destination system uses credit to prevent the source from overflowing the destination buffers, but credit-based schemes have also been considered for congestion control. Credit-based schemes are defined in Frame Relay and ATM networks.
- **Rate based**: These schemes are based on providing an explicit data rate limit to the source over a logical connection. The source may transmit data at a rate up to the set limit. To control congestion, any node along the path of the connection can reduce the data rate limit in a control message to the source.

2.6 SDN and NFV

With the ever-increasing volume and variety of network traffic, generated by such high-demand sources as big data, cloud computing, and mobile traffic, it becomes increasingly difficult to meet stringent QoS and QoE requirements. Networks need to be more adaptable and scalable. To provide adaptability and scalability, two key technologies that are rapidly being deployed by a variety of network service and application providers are software-defined networking (SDN) and network functions virtualization (NFV). Because these two topics occupy a large portion of this book, only a brief introduction is appropriate here.

Software-Defined Networking

SDN has reached a tipping point at which it is replacing the traditional networking model. Software-defined networks provide an enhanced level of flexibility and customizability to meet

the needs of newer networking and IT trends such as cloud, mobility, social networking, and video.

SDN Functionality

The two elements involved in forwarding packets through routers are a control function, which decides the route the traffic takes and the relative priority of traffic, and a data function, which forwards data based on control-function policy. Prior to SDN, these functions were performed in an integrated fashion at each network device (router, bridge, packet switch, and so on). Control in such a traditional network is exercised by means of a routing and control network protocol that is implemented in each network node. This approach is relatively inflexible and requires all the network nodes to implement the same protocols. With SDN, a central controller performs all complex functionality, including routing, naming, policy declaration, and security checks (see Figure 2.15).

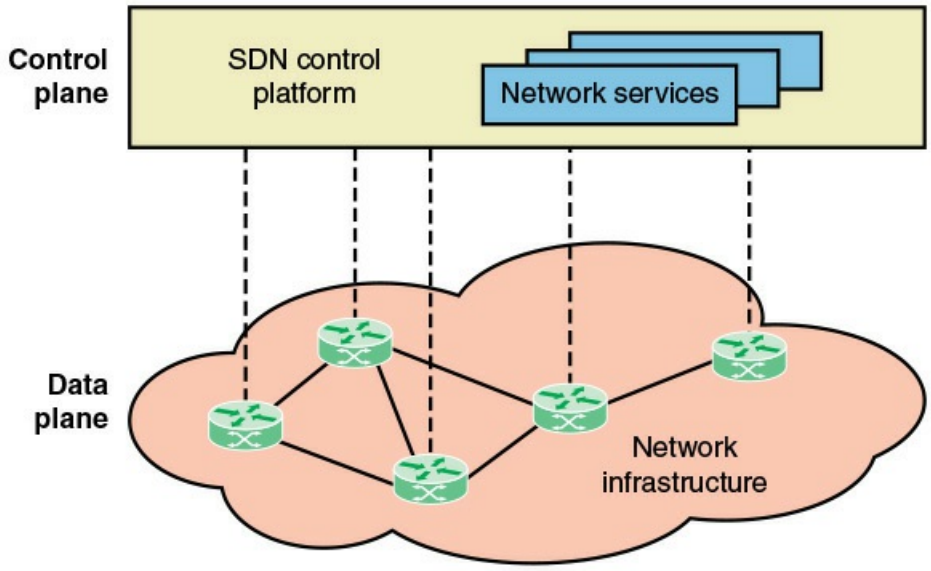


FIGURE 2.15 Software-Defined Networking

➔ *See Part II, "Software-Defined Networks"*

This constitutes the SDN **control plane**, and consists of one or more SDN controllers. The SDN controller defines the data flows that occur in the SDN data plane. Each flow through the network is configured by the controller, which verifies that the communication is permissible by the network policy. If the controller allows a flow requested by an end system, it computes a route for the flow to take, and adds an entry for that flow in each of the switches along the path. With all complex function subsumed by the controller, switches simply manage flow tables whose entries can only be populated by the controller. The switches constitute the **data plane**. Communication between the controller and the switches uses a standardized protocol.

Key Drivers

One driving factor for SDN is the increasingly widespread use of server virtualization. In

essence, server virtualization masks server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. This makes it possible to partition a single machine into multiple, independent servers, conserving hardware resources. It also makes it possible to quickly migrate a server from one machine to another for load balancing or for dynamic switchover in the case of machine failure. Server virtualization has become a central element in dealing with big data applications and in implementing cloud computing infrastructures. But it creates problems with traditional network architectures. One problem is configuring virtual LANs. Network managers need to make sure the VLAN used by the virtual machine (VM) is assigned to the same switch port as the physical server running the VM. But with the VM being movable, it is necessary to reconfigure the VLAN every time that a virtual server is moved. In general terms, to match the flexibility of server virtualization, the network manager needs to be able to dynamically add, drop, and change network resources and profiles. This is difficult to do with conventional network switches, in which the control logic for each switch is collocated with the switching logic.

Another effect of server virtualization is that traffic flows differ substantially from the traditional client/server model. Typically, there is a considerable amount of traffic among virtual servers, for such purposes as maintaining consistent images of database and invoking security functions such as access control. These server-to-server flows change in location and intensity over time, demanding a flexible approach to managing network resources.

Another factor leading to the need for rapid response in allocating network resources is the increasing use by employees of mobile devices, such as smartphones, tablets, and notebooks to access enterprise resources. These devices can add fast-changing and unpredictable large loads on the network, and can rapidly change their network attachment point. Network managers must be able to respond to rapidly changing resource, QoS, and security requirements for mobile devices.

Existing network infrastructures can respond to changing requirements for the management of traffic flows, providing differentiated QoS levels and security levels for individual flows, but the process can be very time-consuming if the enterprise network is large or involves network devices from multiple vendors. The network manager must configure each vendor’s equipment separately, and adjust performance and security parameters on a per-session, per-application basis. In a large enterprise, every time a new VM is brought up, it can take hours or even days for network managers to do the necessary reconfiguration.

Network Functions Virtualization

The discussion of SDN mentioned that a key driving factor in the deployment of SDN is the need to provide flexible network response to the widespread use of virtualized servers. VM technology over the Internet or an enterprise network has, until recently, been used for application-level server functions such as database servers, cloud servers, web servers, e-mail servers, and so on. This same technology, however, can equally be applied to network devices, such as routers, LAN switches, firewalls, and IDS/IPS servers (see Figure 2.16).

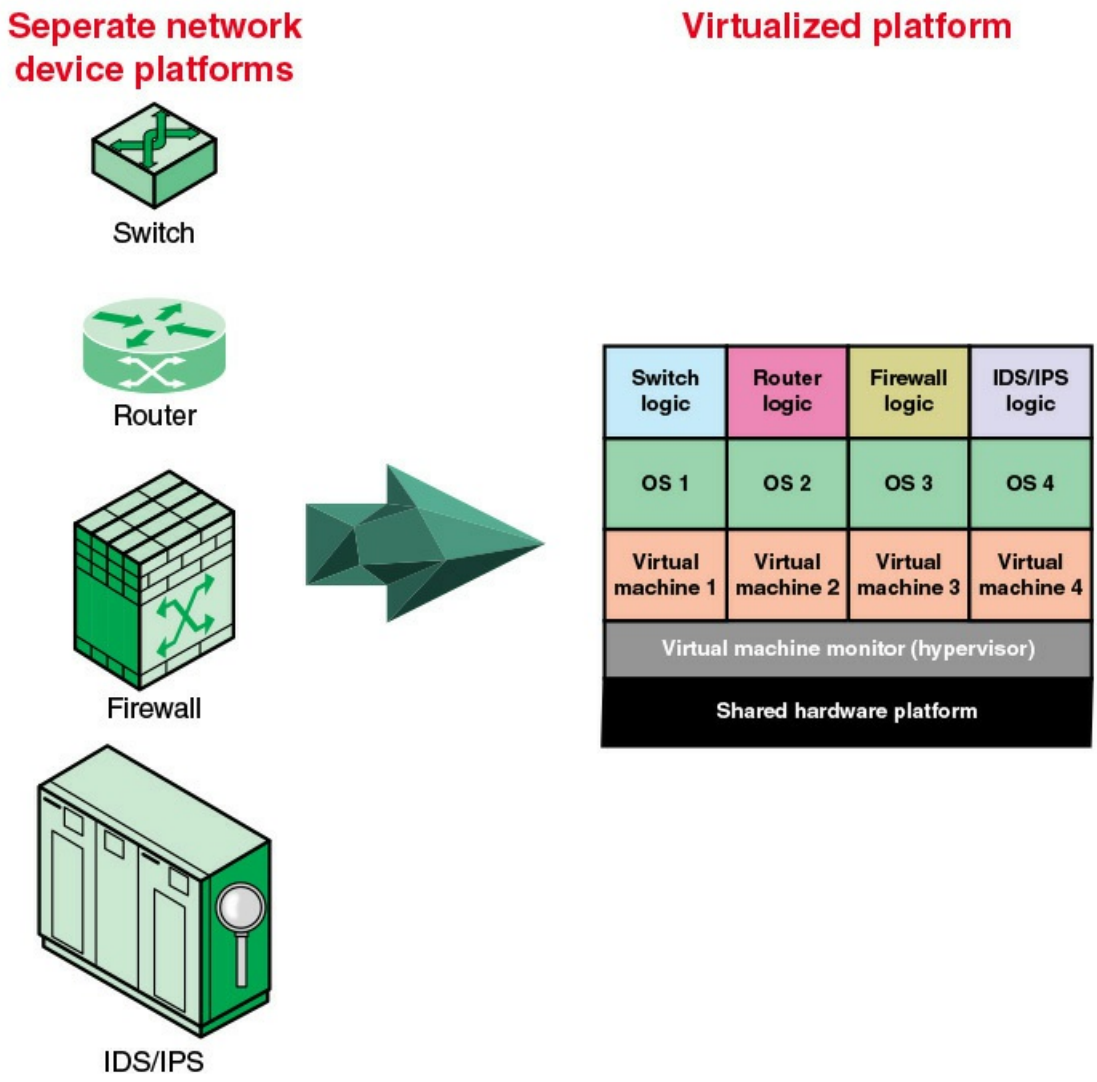


FIGURE 2.16 Network Functions Virtualization

➔ *See Part III, "Virtualization"*

Network functions virtualization (NFV) decouples network functions, such as routing, firewalling, intrusion detection, and Network Address Translation from proprietary hardware platforms and implements these functions in software. It utilizes standard virtualization technologies that run on high-performance hardware to virtualize network functions. It is applicable to any data plane processing or control plane function in both wired and wireless network infrastructures.

NFV has a number of features in common with SDN. They share the following objectives:

- Move functionality to software
- Use commodity hardware platforms instead of proprietary platforms
- Use standardized or open application program interfaces (APIs)
- Support more efficient evolution, deployment, and repositioning of network functions

NFV and SDN are independent but complementary schemes. SDN decouples the data and control planes of network traffic control, making the control and routing of network traffic more flexible and efficient. NFV decouples network functions from specific hardware platforms via virtualization to make the provision of these functions more efficient and flexible. Virtualization can be applied to the data plane functions of the routers and other network functions, including SDN controller functions. So, either can be used alone, but the two can be combined to reap greater benefits.

2.7 Modern Networking Elements

This chapter ends with a rough depiction of how the major elements of modern networking treated in this book fit together (see Figure 2.17). The discussion that follows works through this figure from the bottom up.

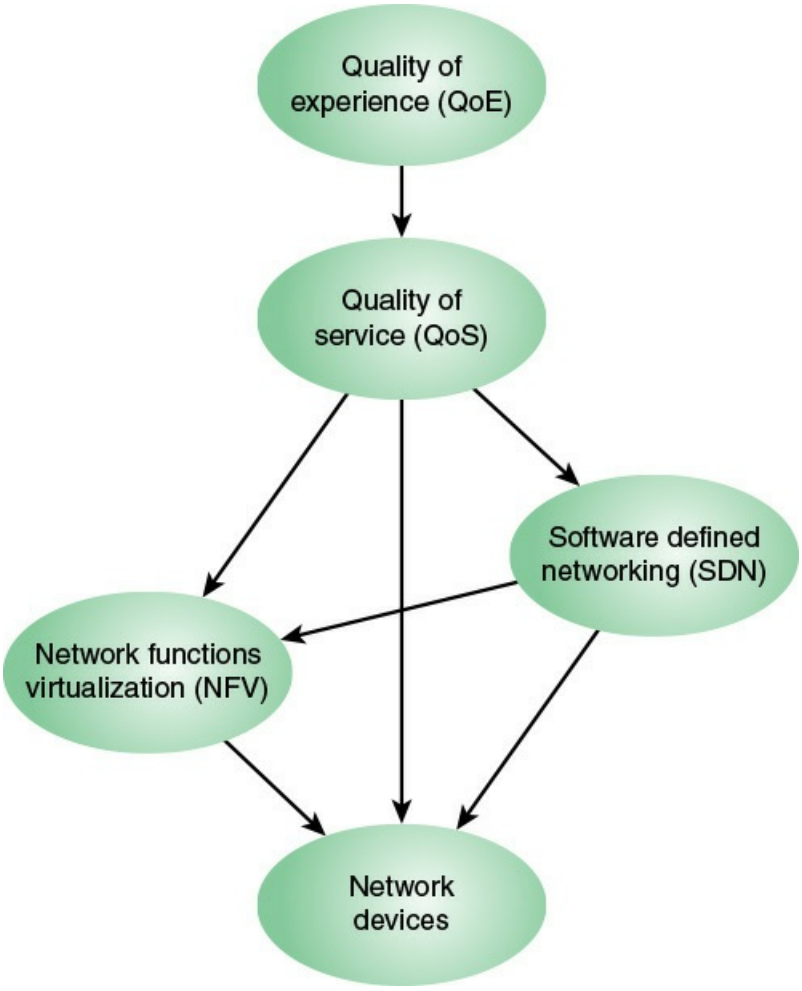


FIGURE 2.17 Modern Networking Schema

Ultimately, the concern of a network service provider is about the set of network devices (such as routers) and the control and management of the functions they perform (such as packet forwarding). If NFV is used, these network functions are implemented in software and executed on VMs. If instead the network functions are implemented on dedicated machines and SDN is

used, the control functions are implemented on central SDN controllers, which interact with the network devices.

However, SDN and NFV are not mutually exclusive. If both SDN and NFV are implemented for a network, the following relationships hold:

- Network data plane functionality is implemented on VMs.
- The control plane functionality may be implemented on a dedicated SDN platform or on an SDN VM.

In either case, the SDN controller interacts with the data plane functions running on VMs.

QoS measures are commonly used to specify the service required by various network customers or users and to dictate the traffic management policies used on the network. The common case, until recently, is that QoS was implemented on network that used neither NFV nor SDN. In this case, routing and traffic control policies must be configured directly on network devices using a variety of automated and manual techniques. If NFV but not SDN is implemented, the QoS settings are communicated to the VMs. With SDN, regardless of whether NFV is used, it is the SDN controller that is responsible for enforcing QoS parameters for the various network users.

If QoE considerations come into play, these are used to adjust QoS parameters to satisfy the users' QoE requirements.

2.8 Key Terms

After completing this chapter, you should be able to define the following terms.

analytics

autonomous system

big data

cloud computing

congestion

delay jitter

elastic traffic

exterior router protocol (ERP)

inelastic traffic

interior router protocol (IRP)

internet

Internet

network functions virtualization (NFV)

operations support system (OSS)

packet forwarding

quality of experience (QoE)

quality of service (QoS)

real-time traffic

router

routing

routing protocol

software-defined networking (SDN)

virtual machine (VM)

2.9 References

AKAM15: Akamai Technologies. *Akamai's State of the Internet*. Akamai Report, Q4|2014. 2015.

CISC14: Cisco Systems. *Cisco Visual Networking Index: Forecast and Methodology*, 2013–2018. White Paper, 2014.

IBM11: IBM Study, "Every Day We Create 2.5 Quintillion Bytes of Data." Storage Newsletter, October 21, 2011. http://www.storagenewsletter.com/rubriques/market-reportsresearch/ibm-cmo-study/

ITUT12: ITU-T. Focus Group on Cloud Computing Technical Report Part 3: Requirements and Framework Architecture of Cloud Infrastructure. FG Cloud TR, February 2012.

KAND12: Kandula, A., Sengupta, S., and Patel, P. "The Nature of Data Center Traffic: Measurements and Analysis." ACM SIGCOMM Internet Measurement Conference, November 2009.

NETW14: Network World. *Survival Tips for Big Data's Impact on Network Performance*. White paper. April 2014.

SZIG14: Szigeti, T., Hattingh, C., Barton, R., and Briley, K. *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*. Englewood Cliffs, NJ: Pearson. 2014.

Part II: Software-Defined Networks

One man had a vision of railways that would link all the mainline railroad termini. His name was Charles Pearson and, though born the son of an upholsterer, he became Solicitor to the city of London. There had previously been a plan for gaslit subway streets through which horse-drawn traffic could pass. This was rejected on the grounds that such sinister tunnels would become lurking places for thieves. Twenty years before his system was built, Pearson envisaged a line running through "a spacious archway," well-lit and well-ventilated.

His was a scheme for trains in a drain.

—*King Solomon's Carpet*, Barbara Vine (Ruth Rendell)

The heart of modern networking is software-defined networking (SDN). Part II is devoted to a broad and thorough presentation of SDN concepts, technology, and applications. Chapter 3 begins the discussion by laying out what the SDN approach is and why it is needed, and provides an overview of the SDN architecture. This chapter also looks at the organizations that are issuing specifications and standards for SDN. Chapter 4 is a detailed look at the SDN data plane, including the key components, how they interact, and how they are managed. Much of the chapter is devoted to OpenFlow, a vital data plane technology as well as an interface to the control plane. The chapter explains why OpenFlow is needed and then proceeds to provide a detailed technical explanation. Chapter 5 is devoted to the SDN control plane. It includes a discussion of OpenDaylight, an important open source implementation of the control plane. Chapter 6 covers the SDN application plane. In addition to examining the general SDN application plane architecture, the chapter discusses six major application areas that can be supported by SDN and provides a number of examples of SDN applications.

Chapter 3. SDN: Background and Motivation

> The requirements for a future all-digital-data distributed network which provides common user service for a wide range of users having different requirements is considered. The use of a standard format message block permits building relatively simple switching mechanisms using an adaptive store-and-forward routing policy to handle all forms of digital data including "real-time" voice. This network rapidly responds to changes in network status.
>
> —On Distributed Communications: Introduction to Distributed Communications Networks, Rand Report RM-3420-PR, Paul Baran, August 1964

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

- Make a presentation justifying the position that traditional network architectures are inadequate for modern networking needs.
- List and explain the key requirements for an SDN architecture.
- Present an overview of an SDN architecture, to include explaining the significance of northbound and southbound APIs.
- Summarize the work being done on SDN and NFV standardization by various organizations.

This chapter begins the discussion of software-defined networks (SDNs) by providing some background and motivation for the SDN approach.

3.1 Evolving Network Requirements

A number of trends are driving network providers and users to reevaluate traditional approaches to network architecture. These trends can be grouped under the categories of demand, supply, and traffic patterns.

Demand Is Increasing

As was described in Chapter 2, "Requirements and Technology," a number of trends are increasing the load on enterprise networks, the Internet, and other internets. Of particular note are the following:

- **Cloud computing:** There has been a dramatic shift by enterprises to both public and private cloud services.
- **Big data:** The processing of huge data sets requires massive parallel processing on thousands of servers, all of which require a degree of interconnection to each other. Therefore, there is a large and constantly growing demand for network capacity within the

data canter.

- **Mobile traffic:** Employees are increasingly accessing enterprise network resources via mobile personal devices, such as smartphones, tablets, and notebooks. These devices support sophisticated apps that can consume and generate image and video traffic, placing new burdens on the enterprise network.
- **The Internet of Things (IoT):** Most "things" in the IoT generate modest traffic, although there are exceptions, such as surveillance video cameras. But the sheer number of such devices for some enterprises results in a significant load on the enterprise network.

Supply Is Increasing

As the demand on networks is rising, so is the capacity of network technologies to absorb rising loads. In terms of transmission technology, Chapter 1, "Elements of Modern Networking," established that the key enterprise wired and wireless network technologies, Ethernet and Wi-Fi respectively, are well into the gigabits per second (Gbps) range. Similarly, 4G and 5G cellular networks provide greater capacity for mobile devices from remote employees who access the enterprise network via cellular networks rather than Wi-Fi.

The increase in the capacity of the network transmission technologies has been matched by an increase in the performance of network devices, such as LAN switches, routers, firewalls, intrusion detection system/intrusion prevention systems (IDS/IPS), and network monitoring and management systems. Year by year, these devices have larger, faster memories, enabling greater buffer capacity and faster buffer access, as well as faster processor speeds.

Traffic Patterns Are More Complex

If it were simply a matter of supply and demand, it would appear that today's networks should be able to cope with today's data traffic. But as traffic patterns have changed and become more complex, traditional enterprise network architectures are increasingly ill suited to the demand.

Until recently, and still common today, the typical enterprise network architecture consisted of a local or campus-wide tree structure of Ethernet switches with routers connecting large Ethernet LANs and connecting to the Internet and WAN facilities. This architecture is well suited to the client/server computing model that was at one time dominant in the enterprise environment. With this model, interaction, and therefore traffic, was mostly between one client and one server. In such an environment, networks could be laid out and configured with relatively static client and server locations and relatively predictable traffic volumes between clients and servers.

A number of developments have resulted in far more dynamic and complex traffic patterns within the enterprise data center, local and regional enterprise networks, and carrier networks. These include the following:

- Client/server applications typically access multiple databases and servers that must communicate with each other, generating "horizontal" traffic between servers as well as "vertical" traffic between servers and clients.
- Network convergence of voice, data, and video traffic creates unpredictable traffic

patterns, often of large multimedia data transfers.

- Unified communications (UC) strategies involve heavy use of applications that trigger access to multiple servers.
- The heavy use of mobile devices, including personal bring your own device (BYOD) policies, results in user access to corporate content and applications from any device anywhere any time. As illustrated previously in Figure 2.6 in Chapter 2, this mobile traffic is becoming an increasingly significant fraction of enterprise network traffic.
- The widespread use of public clouds has shifted a significant amount of what previously had been local traffic onto WANs for many enterprises, resulting in increased and often very unpredictable loads on enterprise routers.
- The now-common practice of application and database server virtualization has significantly increased the number of hosts requiring high-volume network access and results in every-changing physical location of server resources.

Traditional Network Architectures are Inadequate

Even with the greater capacity of transmission schemes and the greater performance of network devices, traditional network architectures are increasingly inadequate in the face of the growing complexity, variability, and high volume of the imposed load. In addition, as quality of service (QoS) and quality of experience (QoE) requirements imposed on the network are expanded as a result of the variety of applications, the traffic load must be handled in an increasingly sophisticated and agile fashion.

The traditional internetworking approach is based on the **TCP/IP protocol architecture**. Three noteworthy characteristics of this approach are as follows:

- Two-level end system addressing

  The protocol architecture built around the TCP and IP protocols, consisting of five layers: physical, data link, network/Internet (usually IP), transport (usually TCP or UDP), and application.

- Routing based on destination
- Distributed, autonomous control

Let’s look at each of these characteristics in turn.

The traditional architecture relies heavily on the network interface identity. At the physical layer of the TCP/IP model, devices attached to networks are identified by hardware-based identifiers, such as Ethernet MAC addresses. At the internetworking level, including both the Internet and private internets, the architecture is a network of networks. Each attached device has a physical layer identifier recognized within its immediate network and a logical network identifier, its IP address, which provides global visibility.

The design of TCP/IP uses this addressing scheme to support the networking of autonomous networks, with distributed control. This architecture provides a high level of resilience and scales well in terms of adding new networks. Using IP and distributed routing protocols, routes can be discovered and used throughout an internet. Using transport-level protocols such as TCP,

distributed and decentralized algorithms can be implemented to respond to congestion.

Traditionally, routing was based on each packet's destination address. In this **datagram** approach, successive packets between a source and destination may follow different routes through the internet, as routers constantly seek to find the minimum-delay path for each individual **packet**. More recently, to satisfy QoS requirements, packets are often treated in terms of **flows** of packets. Packets associated with a given flow have defined QoS characteristics, which affect the routing for the entire flow.

> A packet that is treated independently of other packets for packet switching. A datagram carries information sufficient for routing from the source to the destination without the necessity of establishing a logical connection between the endpoints.
>
> A unit of data sent across a network. A packet is a group of bits that includes data plus protocol control information. The term generally applies to protocol data units at the network layer.
>
> A sequence of packets between a source and destination that are recognized by the network as related and are treated in a uniform fashion.
>
> A method of transmitting messages through a communications network, in which long messages are subdivided into short packets. Each packet is passed from source to destination through intermediate nodes. At each node, the entire message is received, stored briefly, and then forwarded to the next node.

However, this distributed, autonomous approach developed when networks were predominantly static and end systems predominantly of fixed location. Based on these characteristics, the Open Networking Foundation (ONF) cites four general limitations of traditional network architectures [ONF12]:

- **Static, complex architecture:** To respond for demands such as differing levels of QoS, high and fluctuating traffic volumes, and security requirements, networking technology has grown more complex and difficult to manage. This has resulted in a number of independently defined protocols each of which addresses a portion of networking requirements. An example of the difficulty this presents is when devices are added or moved. The network management staff must use device-level management tools to make changes to configuration parameters in multiple switches, routers, firewalls, web authentication portals, and so on. The updates include changes to access control lists (ACLs), virtual LAN settings, QoS settings in numerous devices, and other protocol-related adjustments. Another example is the adjustment of QoS parameters to meet changing user requirements and traffic patterns. Manual procedures must be used to configure each vendor's equipment on a per-application and even per-session basis.
- **Inconsistent policies:** To implement a network-wide security policy, staff may have to make configuration changes to thousands of devices and mechanisms. In a large network, when a new virtual machine is activated, it can take hours or even days to reconfigure ACLs across the entire network.
- **Inability to scale:** Demands on networks are growing rapidly, both in volume and variety. Adding more switches and transmission capacity, involving multiple vendor equipment, is difficult because of the complex, static nature of the network. One strategy enterprises

have used is to oversubscribe network links based on predicted traffic patterns. But with the increased use of virtualization and the increasing variety of multimedia applications, traffic patterns are unpredictable.

- **Vendor dependence:** Given the nature of today's traffic demands on networks, enterprises and carriers need to deploy new capabilities and services rapidly in response to changing business needs and user demands. A lack of open interfaces for network functions leaves the enterprises limited by the relatively slow product cycles of vendor equipment.

3.2 The SDN Approach

This section provides an overview of SDN and shows how it is designed to meet evolving network requirements.

Requirements

Based on the narrative of Section 3.1, we are now in a position to detail the principal requirements for a modern networking approach. The Open Data Center Alliance (ODCA) provides a useful, concise list of requirements, which include the following [ODCA14]:

- **Adaptability:** Networks must adjust and respond dynamically, based on application needs, business policy, and network conditions.
- **Automation:** Policy changes must be automatically propagated so that manual work and errors can be reduced.
- **Maintainability**. Introduction of new features and capabilities (software upgrades, patches) must be seamless with minimal disruption of operations.
- **Model management:** Network management software must allow management of the network at a model level, rather than implementing conceptual changes by reconfiguring individual network elements.
- **Mobility:** Control functionality must accommodate mobility, including mobile user devices and virtual servers.
- **Integrated security:** Network applications must integrate seamless security as a core service instead of as an add-on solution.
- **On-demand scaling:** Implementations must have the ability to scale up or scale down the network and its services to support on-demand requests.

SDN Architecture

An analogy can be drawn between the way in which computing evolved from closed, vertically integrated, proprietary systems into an open approach to computing and the evolution coming with SDN (see Figure 3.1). In the early decades of computing, vendors such as IBM and DEC provided a fully integrated product, with a proprietary processor hardware, unique assembly

language, unique operating system (OS), and the bulk if not all of the application software. In this environment, customers, especially large customers, tended to be locked in to one vendor, dependent primarily on the applications offered by that vendor. Migration to another vendor's hardware platform resulted in major upheaval at the application level.

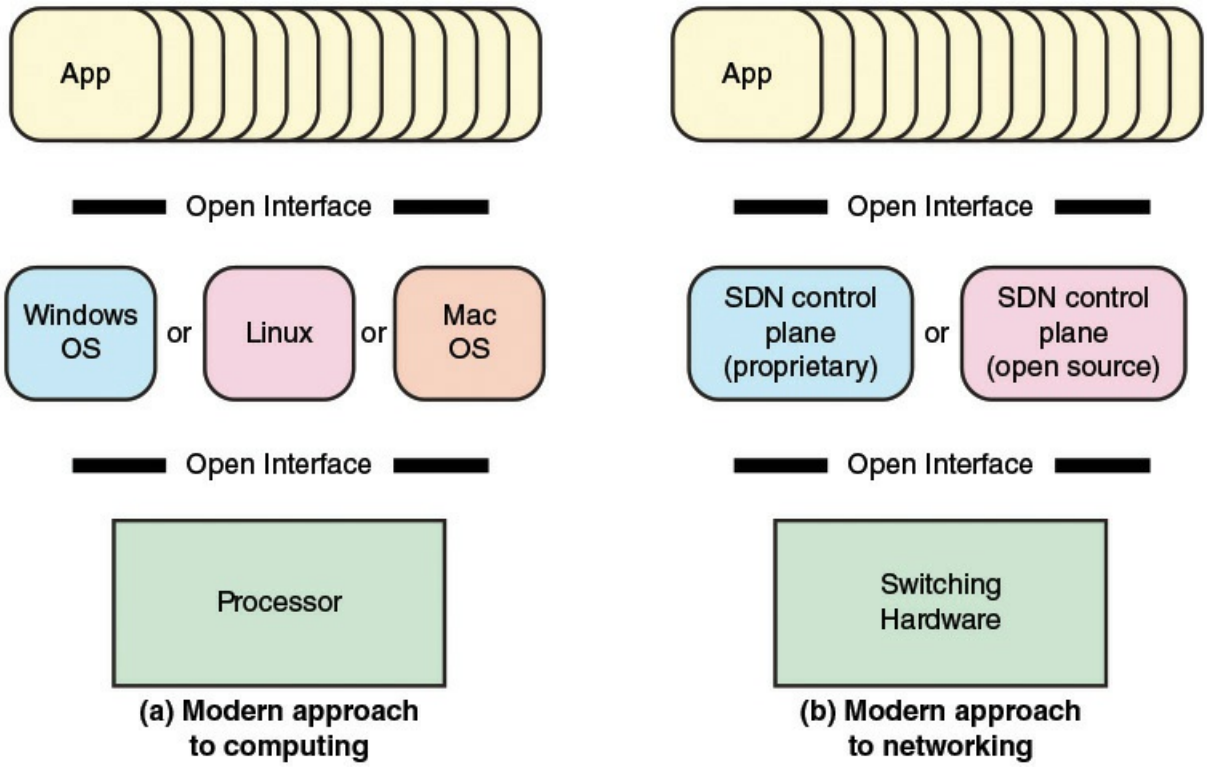


FIGURE 3.1 The Modern Approach to Computing and Networking

Today, the computing environment is characterized by extreme openness and great customer flexibility. The bulk of computing hardware consists of x86 and x86-compatible processors for standalone systems and ARM processors for embedded systems. This makes it easy to port operating systems implemented in C, C++, Java, and the like. Even proprietary hardware architectures, such as IBM's zEnterprise line, provide standardized compilers and programming environments and so can easily run open sources operating systems such as Linux. Therefore, applications written for Linux or other open operating systems can easily be moved from one vendor platform to another. Even proprietary systems such as Windows and Mac OS provide programming environments to make porting of applications an easy matter. It also enables the development of virtual machines that can be moved from one server to another across hardware platforms and operating systems.

The networking environment today faces some of the same limitations faced in the pre-open era of computing. Here the issue is not developing applications that can run on multiple platforms. Rather, the difficulty is the lack of integration between applications and network infrastructure. As demonstrated in the preceding section, traditional network architectures are inadequate to meet the demands of the growing volume and variety of traffic.

The central concept behind SDN is to enable developers and network managers to have the same type of control over network equipment that they have had over x86 servers. As discussed in

Section 2.6 in Chapter 2, the SDN approach splits the switching function between a data plane and a control plane that are on separate devices (see Figure 3.2). The data plane is simply responsible for forwarding packets, whereas the control plane provides the "intelligence" in designing routes, setting priority and routing policy parameters to meet QoS and QoE requirements and to cope with the shifting traffic patterns. Open interfaces are defined so that the switching hardware presents a uniform interface regardless of the details of internal implementation. Similarly, open interfaces are defined to enable networking applications to communicate with the SDN controllers.

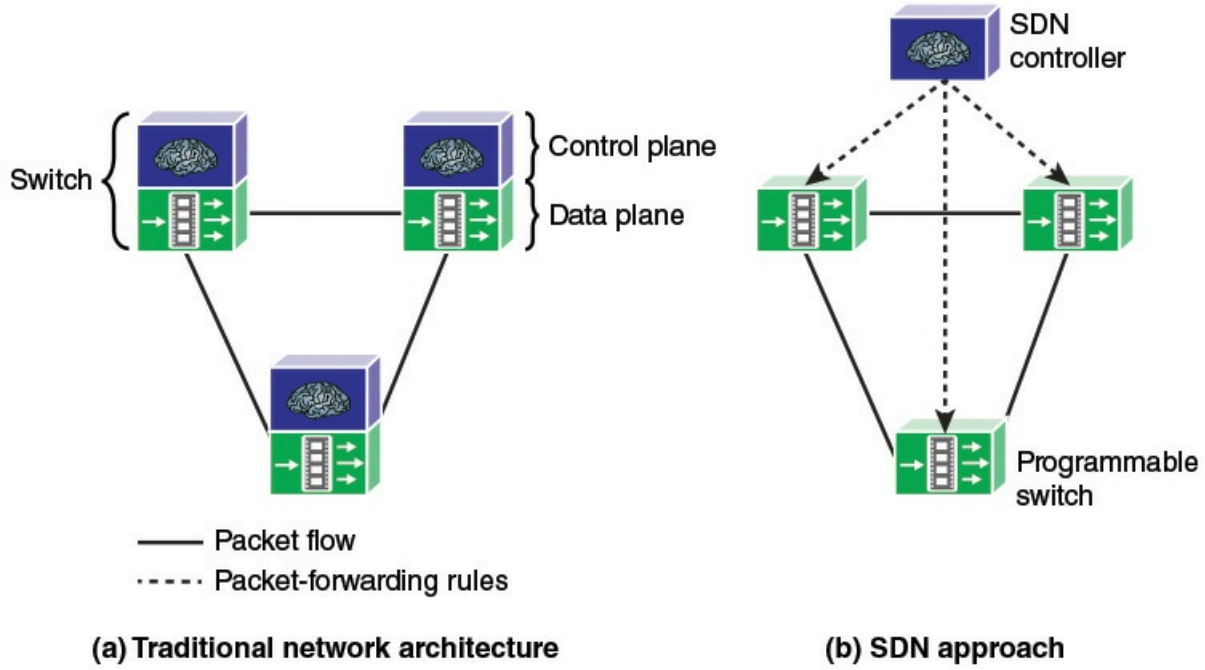


FIGURE 3.2 Control and Data Planes

← *See Figure 2.15, Software Defined Networking*

Figure 3.3 elaborates on the structure shown in Figure 2.15, showing more detail of the SDN approach. The data plane consists of physical switches and virtual switches. In both cases, the switches are responsible for forwarding packets. The internal implementation of buffers, priority parameters, and other data structures related to forwarding can be vendor dependent. However, each switch must implement a model, or abstraction, of packet forwarding that is uniform and open to the SDN controllers. This model is defined in terms of an open **application programming interface (API)** between the control plane and the data plane (southbound API). The most prominent example of such an open API is OpenFlow, discussed in Chapter 4, "SDN Data Plane and OpenFlow." As Chapter 4 explains, the OpenFlow specification defines both a protocol between the control and data planes and an API by which the control plane can invoke the OpenFlow protocol.

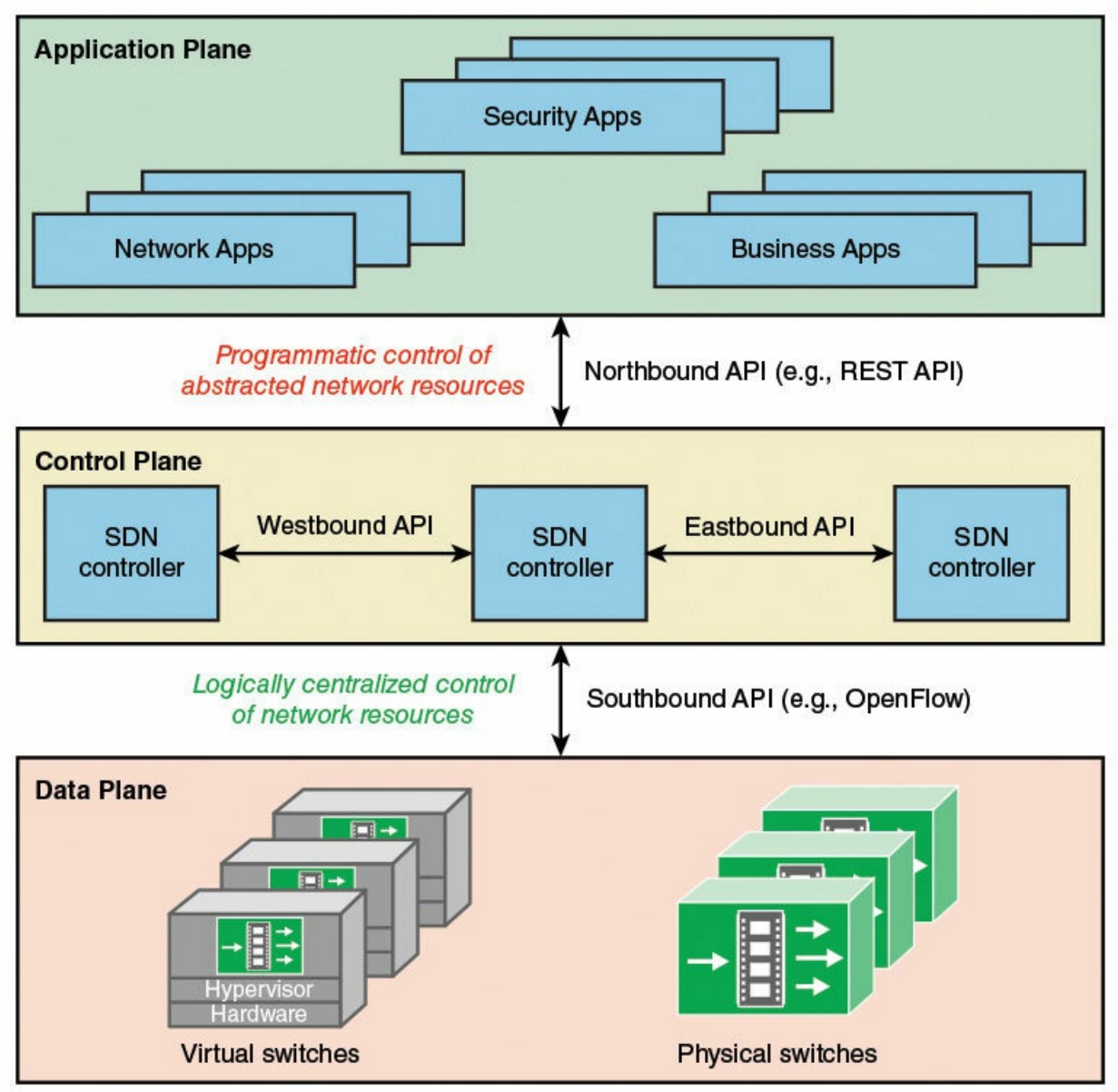


FIGURE 3.3 Software-Defined Architecture

A language and message format used by an application program to communicate with the operating system or some other control program such as a database management system (DBMS) or communications protocol. APIs are implemented by writing function calls in the program, which provide the linkage to the required subroutine for execution. An open or standardized API can ensure the portability of the application code and the vendor independence of the called service.

SDN controllers can be implemented directly on a server or on a virtual server. OpenFlow or some other open API is used to control the switches in the data plane. In addition, controllers use information about capacity and demand obtained from the networking equipment through which the traffic flows. SDN controllers also expose northbound APIs, which allow developers and network managers to deploy a wide range of off-the-shelf and custom-built network applications,

many of which were not feasible before the advent of SDN. As yet there is no standardized northbound API nor a consensus on an open northbound API. A number of vendors offer a REpresentational State Transfer (REST)-based API to provide a programmable interface to their SDN controller.

➔ *See Chapter 5, "SDN Control Plane"*

Also envisioned but not yet defined are horizontal APIs (east/westbound), which would enable communication and cooperation among groups or federations of controllers to synchronize state for high availability.

At the application plane are a variety of applications that interact with SDN controllers. SDN applications are programs that may use an abstract view of the network for their decision-making goals. These applications convey their network requirements and desired network behavior to the SDN controller via a northbound API. Examples of applications are energy-efficient networking, security monitoring, access control, and network management.

Characteristics of Software-Defined Networking

Putting it all together, the key characteristics of SDN are as follows:

- The control plane is separated from the data plane. Data plane devices become simple packet-forwarding devices (refer back to Figure 3.2).
- The control plane is implemented in a centralized controller or set of coordinated centralized controllers. The SDN controller has a centralized view of the network or networks under its control. The controller is portable software that can run on commodity servers and is capable of programming the forwarding devices based on a centralized view of the network.
- Open interfaces are defined between the devices in the control plane (controllers) and those in the data plane.
- The network is programmable by applications running on top of the SDN controllers. The SDN controllers present an abstract view of network resources to the applications.

3.3 SDN- and NFV-Related Standards

Unlike some technology areas, such as Wi-Fi, there is no single standards body responsible for developing open **standards** for SDN and NFV. Rather, there is a large and evolving collection of standards-developing organizations (SDOs), industrial consortia, and open development initiatives involved in creating standards and guidelines for SDN and NFV. Table 3.1 lists the main SDOs and other organizations involved in the effort and the main outcomes so far produced. This section covers some of the most prominent efforts.

Organization	Mission	SDN- and NFV-Related Effort
Open Networking Foundation (ONF)	An industry consortium dedicated to the promotion and adoption of SDN through open standard development.	OpenFlow
Internet Engineering Task Force (IETF)	The Internet's technical standards body. Produces RFCs and Internet standards.	Interface to routing systems (I2RS) Service function chaining
European Telecommunications Standards Institute (ETSI)	An EU-sponsored standards organization that produces globally applicable standards for information and communications technologies.	NFV architecture
OpenDaylight	A collaborative project under the auspices of the Linux Foundation.	OpenDaylight
International Telecommunication Union—Telecommunication Standardization Sector (ITU-T)	United Nations agency that produces Recommendations with a view to standardizing telecommunications on a worldwide basis.	SDN functional requirements and architecture
Internet Research Task Force (IRTF) Software Defined Networking Research Group (SDNRG)	Research group within IRTF. Produces SDN-related RFCs.	SDN architecture

Broadband Forum (BBF)	Industry consortium developing broadband packet networking specifications.	Requirements and framework for SDN in telecommunications broadband networks
Metro Ethernet Forum (MEF)	Industry consortium that promotes the use of Ethernet for metropolitan and wide-area applications.	Defining APIs for service orchestration over SDN and NFV
IEEE 802	An IEEE committee responsible for developing standards for LANs.	Standardize SDN capabilities on access networks.
Optical Internetworking Forum (OIF)	Industry consortium promoting development and deployment of interoperable networking solutions and services for optical networking products.	Requirements on transport networks in SDN architectures
Open Data Center Alliance (ODCA)	Consortium of leading IT organizations developing interoperable solutions and services for cloud computing.	SDN usage model
Alliance for Telecommunications Industry Solutions (ATIS)	A standards organization that develops standards for the unified communications (UC) industry.	Operational opportunities and challenges of SDN/NFV programmable infrastructure
Open Platform for NFV (OPNFV)	An open source project focused on accelerating the evolution of NFV.	NFV infrastructure

TABLE 3.1 SDN and NFV Open Standards Activities

Documents that provide requirements, specifications, guidelines, or characteristics that can be used consistently to ensure that materials, products, processes, and services are fit for their purpose. Standards are established by consensus among those participating in a standards-making organization and are approved by a generally recognized body.

A standard that is: developed on the basis of an open decision-making procedure available to all interested parties, is available for implementation to all on a royalty-free basis, and is intended to promote interoperability among products from multiple vendors.

Standards-Developing Organizations

The Internet Society, ITU-T, and ETSI are all making key contributions to the standardization of SDN and NFV.

Internet Society

A number of **standards-developing organizations (SDOs)** are looking at various aspects of SDN. Perhaps the most active are two groups within the Internet Society (ISOC): IETF and

IRTF. ISOC is the coordinating committee for Internet design, engineering, and management. Areas covered include the operation of the Internet itself and the standardization of protocols used by end systems on the Internet for interoperability. Various organizations under the ISOC are responsible for the actual work of standards development and publication.

> An official national, regional, or international standards body that develops standards and coordinates the standard activities of a specific country, region or the world. Some SDOs facilitate the development of standards through support of technical committee activities, and some may be directly involved in standards development.

The Internet Engineering Task Force (IETF) has working groups developing SDN-related specifications in the following areas:

- **Interface to routing systems (I2RS):** Develop capabilities to interact with routers and routing protocols to apply routing policies.
- **Service function chaining**: Develop an architecture and capabilities for controllers to direct subsets of traffic across the network in such a way that each virtual service platform sees only the traffic it must work with.

The Internet Research Task Force (IRTF) has published *Software-Defined Networking (SDN): Layers and Architecture Terminology* (RFC 7426, January 2015). The document provides a concise reference that reflects current approaches regarding the SDN layer architecture. The **Request For Comments (RFC)** also provides a useful discussion of the southbound API (Figure 3.3) and describes some specific APIs, such as for I2RS.

> A document in the archival series that is the official channel for publications of the Internet Society, including IETF and IRTF publications. An RFC may be informational, best practice, draft standard, or an official Internet standard.

IRTF also sponsors the Software Defined Networking Research Group (SDNRG). This group investigates SDN from various perspectives with the goal of identifying the approaches that can be defined, deployed, and used in the near term and identifying future research challenges.

ITU-T

The International Telecommunication Union—Telecommunication Standardization Sector (ITU-T) is a UN agency that issues standards, called recommendations, in the telecommunications area. So far, their only published contribution to SDN is Recommendation Y.3300 (*Framework of Software-Defined Networking*, June 2014). The document addresses definitions, objectives, high-level capabilities, requirements, and high-level architecture of SDN. It provides a valuable framework for standards development.

ITU-T has established a Joint Coordination Activity on Software-Defined Networking (JCA-SDN) and begun work on developing SDN-related standards.

Four ITU-T study groups (SGs) are involved in SDN-related activities:

- **SG 13 (Future networks, including cloud computing, mobile, and next-generation networks):** This is the lead study group of SDN in ITU-T and developed Y.3300. This group is studying SDN and virtualization aspects for next-generation networks (NGNs).

- **SG 11 (Signaling requirements, protocols, and test specifications):** This group is studying the framework for SDN signaling and how to apply SDN technologies for IPv6.
- **SG 15 (Transport, access, and home):** This group looks at optical transport networks, access networks, and home networks. The group is investigating transport aspects of SDN, aligned with the Open Network Foundation's SDN architecture.
- **SG 16 (Multimedia):** This group is evaluating OpenFlow as a protocol to control multimedia packet flows, and is studying virtual content delivery networks.

European Telecommunications Standards Institute

ETSI is recognized by the European Union as a European Standards Organization. However, this not-for-profit SDO has member organizations worldwide and its standards have international impact.

ETSI has taken the lead role in defining standards for NFV. ETSI's Network Functions Virtualisation (NFV) Industry Specification Group (ISG) began work in January 2013 and produced a first set of specifications in January 2015. The 11 specifications include an NFV's architecture, infrastructure, service quality metrics, management and orchestration, resiliency requirements, and security guidance.

Industry Consortia

Consortia for open standards began to appear in the late 1980s. There was a growing feeling within private-sector multinational companies that the SDOs acted too slowly to provide useful standards in the fast-paced world of technology. Recently, a number of consortia have become involved in the development of SDN and NFV standards. We mention here three of the most significant efforts.

➔ *See Chapter 4, "SDN Data Plane and OpenFlow"*

By far the most important **consortium** involved in SDN standardization is the Open Networking Foundation (ONF). ONF is an industry consortium dedicated to the promotion and adoption of SDN through open standards development. Its most important contribution to date is the OpenFlow protocol and API. The OpenFlow protocol is the first standard interface specifically designed for SDN and is already being deployed in a variety of networks and networking products, both hardware based and software based. The standard enables networks to evolve by giving logically centralized control software the power to modify the behavior of network devices through a well-defined "forwarding instruction set." Chapter 4 is devoted to this protocol.

> A group of independent organizations joined by common interests. In the area of standards development, a consortium typically consists of individual corporations and trade groups concerned with a specific area of technology.

The Open Data Center Alliance (ODCA) is a consortium of leading global IT organizations dedicated to accelerating adoption of interoperable solutions and services for cloud computing. Through the development of usage models for SDN and NFV, ODCA is defining requirements

for SDN and NFV cloud deployment.

The Alliance for Telecommunications Industry Solutions (ATIS) is a membership organization that provides the tools necessary for the industry to identify standards, guidelines, and operating procedures that make the interoperability of existing and emerging telecommunications products and services possible. Although ATIS is accredited by ANSI, it is best viewed as a consortium rather than an SDO. So far, ATIS has issued a document that identifies operational issues and opportunities associated with increasing programmability of the infrastructure using SDN and NFV.

Open Development Initiatives

There are a number of other organizations that are not specifically created by industry members and are not official bodies such as SDOs. Generally, these organizations are user created and driven and have a particular focus, always with the goal of developing open standards or open source software. A number of such groups have become active in SDN and NFV standardization. This section lists three of the most significant efforts.

OpenDaylight

OpenDaylight is an open source software activity under the auspices of the Linux foundation. Its member companies provide resources to develop an SDN controller for a wide range of applications. Although the core membership consists of companies, individual developers and users can also participate, so OpenDaylight is more in the nature of an open development initiative than a consortium. ODL also supports network programmability via southbound protocols, a bunch of programmable network services, a collection of northbound APIs, and a set of applications.

➔ *See Section 5.3, "Open-Daylight"*

OpenDaylight is composed of about 30 projects, and releases their outputs in simultaneous manner. After its first release, Hydrogen, in February 2014, it successfully delivered the second one, Helium, at the end of September 2014.

Open Platform for NFV

Open Platform for NFV is an open source project dedicated to acceleration the adoption of standardized NFV elements. OPNFV will establish a carrier-grade, integrated, open source reference platform that industry peers will build together to advance the evolution of NFV and to ensure consistency, performance, and interoperability among multiple open source components. Because multiple open source NFV building blocks already exist, OPNFV will work with upstream projects to coordinate continuous integration and testing while filling development gaps.

➔ *See Section 7.4, "NFV Benefits and Requirements"*

OpenStack

OpenStack is an open source software project that aims to produce an open source cloud operating system. It provides multitenant Infrastructure as a Service (IaaS), and aims to meets the needs of public and private clouds regardless of size, by being simple to implement and massively scalable. SDN technology is expected to contribute to its networking part, and to make the cloud operating system more efficient, flexible, and reliable.

OpenStack is composed of a number of projects. One of them, Neutron, is dedicated for networking. It provides Network as a Service (NaaS) to other OpenStack services. Almost all SDN controllers have provided plug-ins for Neutron, and through them services on OpenStack and other OpenStack services can build rich networking topologies and can configure advanced network policies in the cloud.

3.4 Key Terms

After completing this chapter, you should be able to define the following terms.

application programming interface (API)

consortium

datagram

flow

IEEE 802

northbound API

open standard

packet switching

REpresentational State Transfer (REST)

Request For Comments (RFC)

service function chaining

southbound API

standard

standards-developing organization (SDO)

TCP/IP protocol architecture

3.5 References

ODCA14: Open Data Center Alliance. *Open Data Center Alliance Master Usage Model: Software-Defined Networking Rev. 2.0*. White Paper. 2014.

ONF12: Open Networking Foundation. *Software-Defined Networking: The New Norm for*

Networks. ONF White Paper, April 13, 2012.

Chapter 4. SDN Data Plane and OpenFlow

> "I tell you," went on Syme with passion, "that every time a train comes in I feel that it has broken past batteries of besiegers, and that man has won a battle against chaos. You say contemptuously that when one has left Sloane Square one must come to Victoria. I say that one might do a thousand things instead, and that whenever I really come there I have the sense of hairbreadth escape. And when I hear the guard shout out the word 'Victoria', it is not an unmeaning word. It is to me the cry of a herald announcing conquest. It is to me indeed 'Victoria'; it is the victory of Adam."
>
> —*The Man Who Was Thursday*, G. K. Chesterton

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

- Present an overview of the functions of the SDN data plane.
- Understand the concept of an OpenFlow logical network device.
- Describe and explain the OpenFlow flow table entry structure.
- Summarize the operation of the OpenFlow pipeline.
- Explain the operation of the group table.
- Understand the basic elements of the OpenFlow protocol.

Section 4.1 of this chapter begins the detailed study of software-defined networking (SDN) with a discussion of the data plane (Figure 4.1). The remainder of the chapter is devoted to OpenFlow, the most widely used implementation of the SDN data plane. OpenFlow is both a specification of the logical structure of data plane functionality and a protocol between SDN controllers and network devices. Sections 4.2 and 4.3, respectively, examine the OpenFlow logical network device and the OpenFlow protocol in more detail.

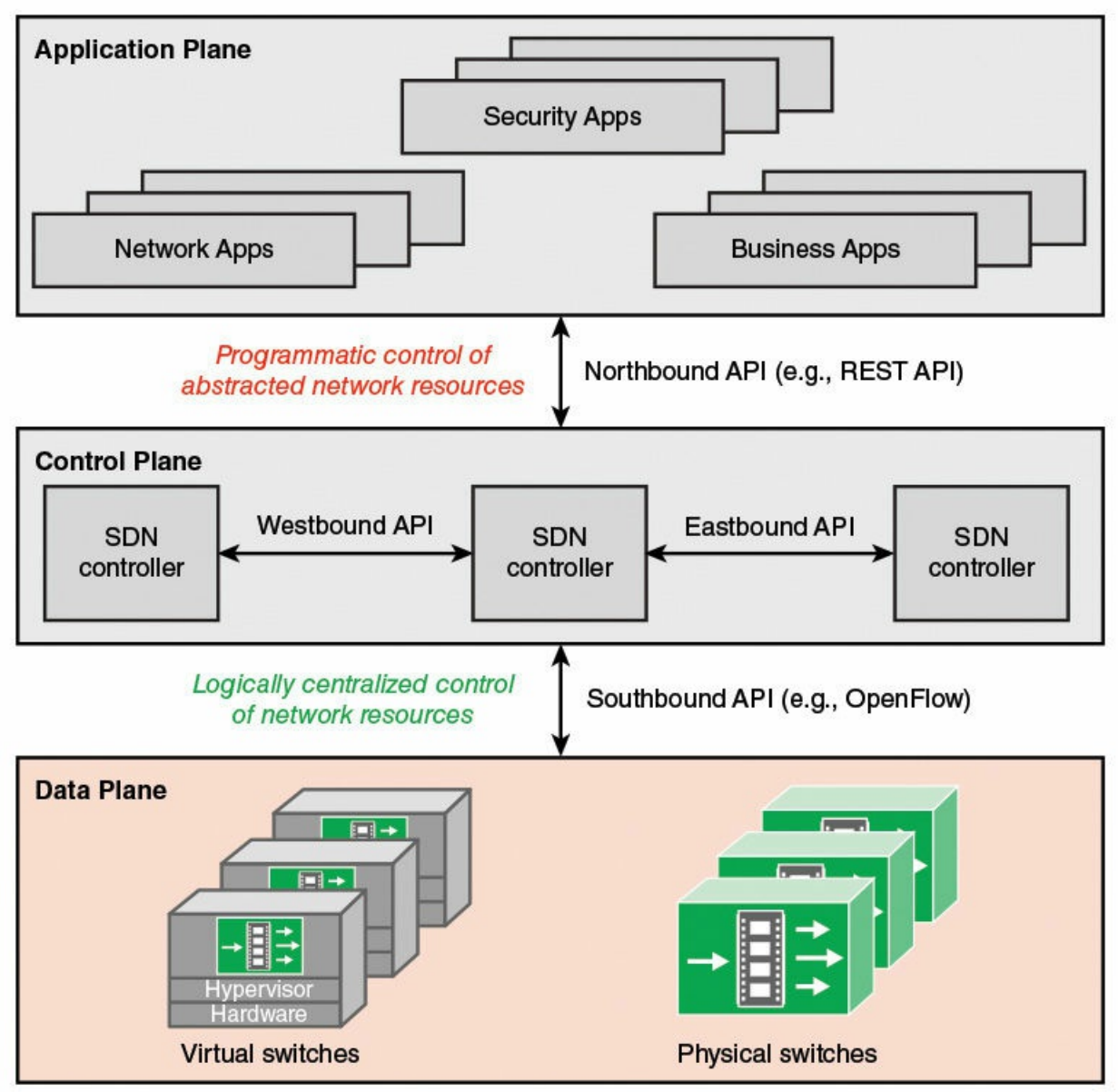


FIGURE 4.1 SDN Architecture

4.1 SDN Data Plane

The SDN data plane, referred to as the resource layer in ITU-T Y.3300 and also often referred to as the infrastructure layer, is where network forwarding devices perform the transport and processing of data according to decisions made by the SDN control plane. The important characteristic of the network devices in an SDN network is that these devices perform a simple forwarding function, without embedded software to make autonomous decisions.

Data Plane Functions

Figure 4.2 illustrates the functions performed by the data plane network devices (also called data plane network elements or switches). The principal functions of the network device are the following:

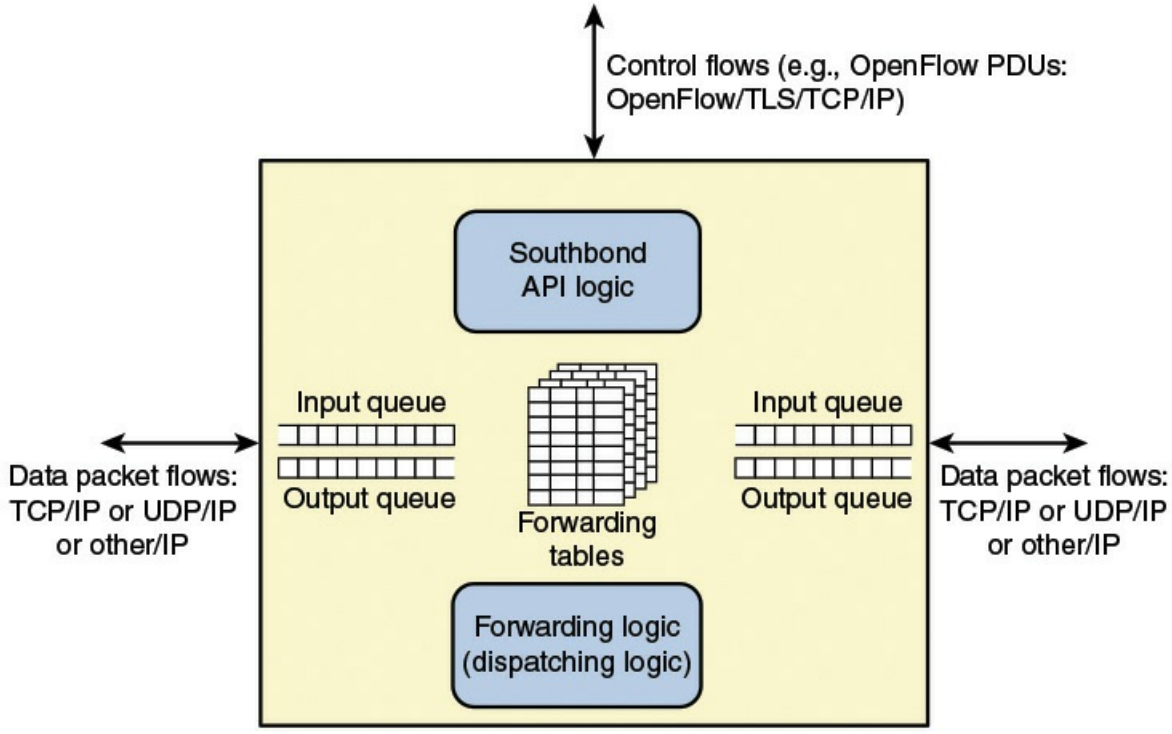


FIGURE 4.2 Data Plane Network Device

- **Control support function:** Interacts with the SDN control layer to support programmability via resource-control interfaces. The switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol.
- **Data forwarding function:** Accepts incoming data flows from other network devices and end systems and forwards them along the data forwarding paths that have been computed and established according to the rules defined by the SDN applications.

These forwarding rules used by the network device are embodied in forwarding tables that indicate for given categories of packets what the next hop in the route should be. In addition to simple forwarding of a packet, the network device can alter the packet header before forwarding, or discard the packet. As shown, arriving packets may be placed in an input queue, awaiting processing by the network device, and forwarded packets are generally placed in an output queue, awaiting transmission.

The network device in Figure 4.2 is shown with three I/O ports: one providing control communication with an SDN controller, and two for the input and output of data packets. This is a simple example. The network device may have multiple ports to communicate with multiple SDN controllers, and may have more than two I/O ports for packet flows into and out of the device.

Data Plane Protocols

Figure 4.2 suggests the protocols supported by the network device. Data packet flows consist of streams of IP packets. It may be necessary for the forwarding table to define entries based on fields in upper-level protocol headers, such as TCP, UDP, or some other transport or application protocol. The network device examines the IP header and possibly other headers in each packet and makes a forwarding decision.

The other important flow of traffic is via the southbound application programming interface (API), consisting of OpenFlow protocol data units (PDUs) or some similar southbound API protocol traffic.

4.2 OpenFlow Logical Network Device

To turn the concept of SDN into practical implementation, two requirements must be met:

- There must be a common logical architecture in all switches, routers, and other network devices to be managed by an SDN controller. This logical architecture may be implemented in different ways on different vendor equipment and in different types of network devices, as long as the SDN controller sees a uniform logical switch functionality.
- A standard, secure protocol is needed between the SDN controller and the network device.

These requirements are addressed by OpenFlow, which is both a protocol between SDN controllers and network devices and a specification of the logical structure of the network switch functionality. OpenFlow is defined in the *OpenFlow Switch Specification*, published by the Open Networking Foundation (ONF).

Open Network Foundation OpenFlow Definition

This section covers the logical switch architecture defined by OpenFlow. Our discussion is based on the OpenFlow specification current at the time of this writing: Version 1.5.1, March 26, 2015.

Figure 4.3 indicates the main elements of an OpenFlow environment, consisting of SDN controllers that include OpenFlow software, OpenFlow switches, and end systems.

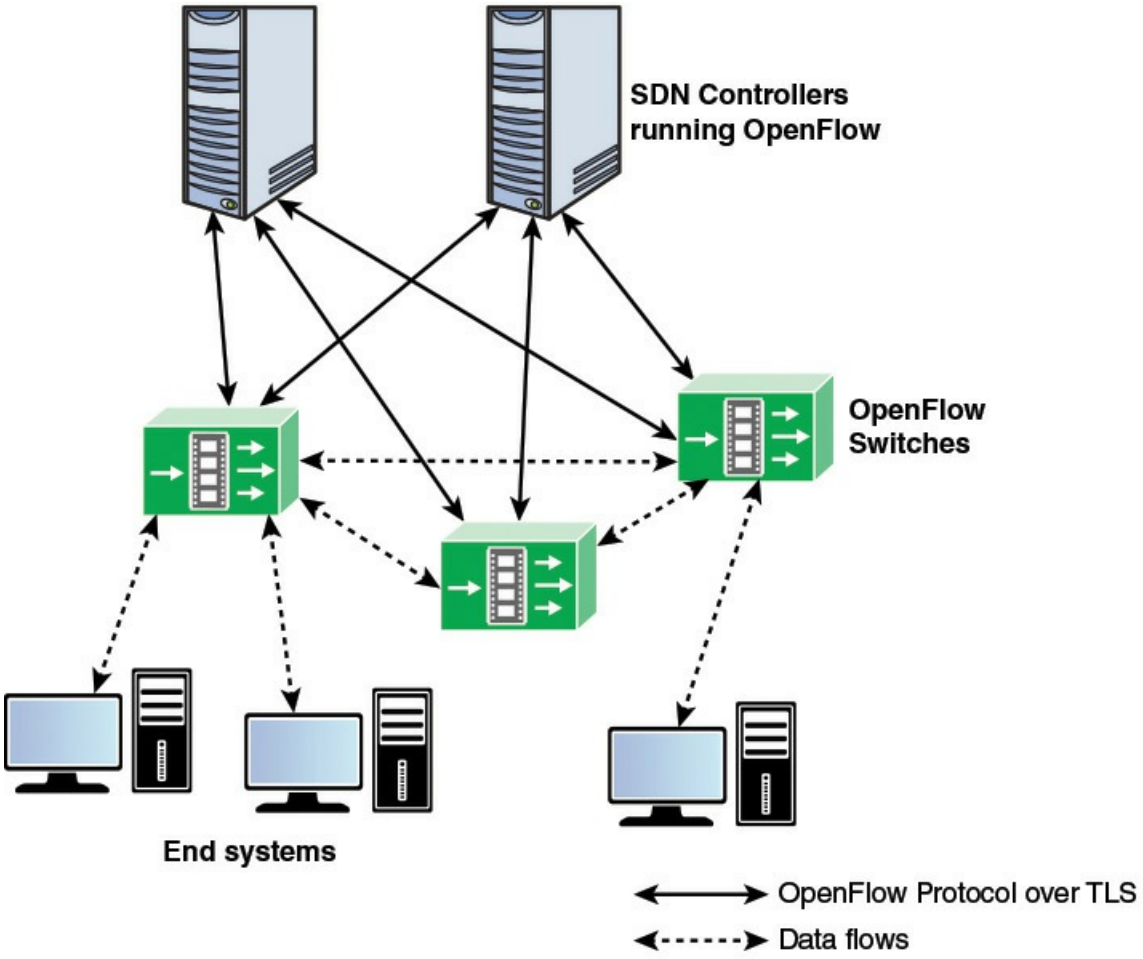


FIGURE 4.3 OpenFlow Switch Context

Figure 4.4 displays the main components of an OpenFlow switch. An SDN controller communicates with OpenFlow-compatible switches using the OpenFlow protocol running over Transport Layer Security (TLS). Each switch connects to other **OpenFlow switches** and, possibly, to end-user devices that are the sources and destinations of packet flows. On the switch side, the interface is known as an **OpenFlow channel**. These connections are via OpenFlow ports. An **OpenFlow port** also connects the switch to the SDN controller. OpenFlow defines three types of ports:

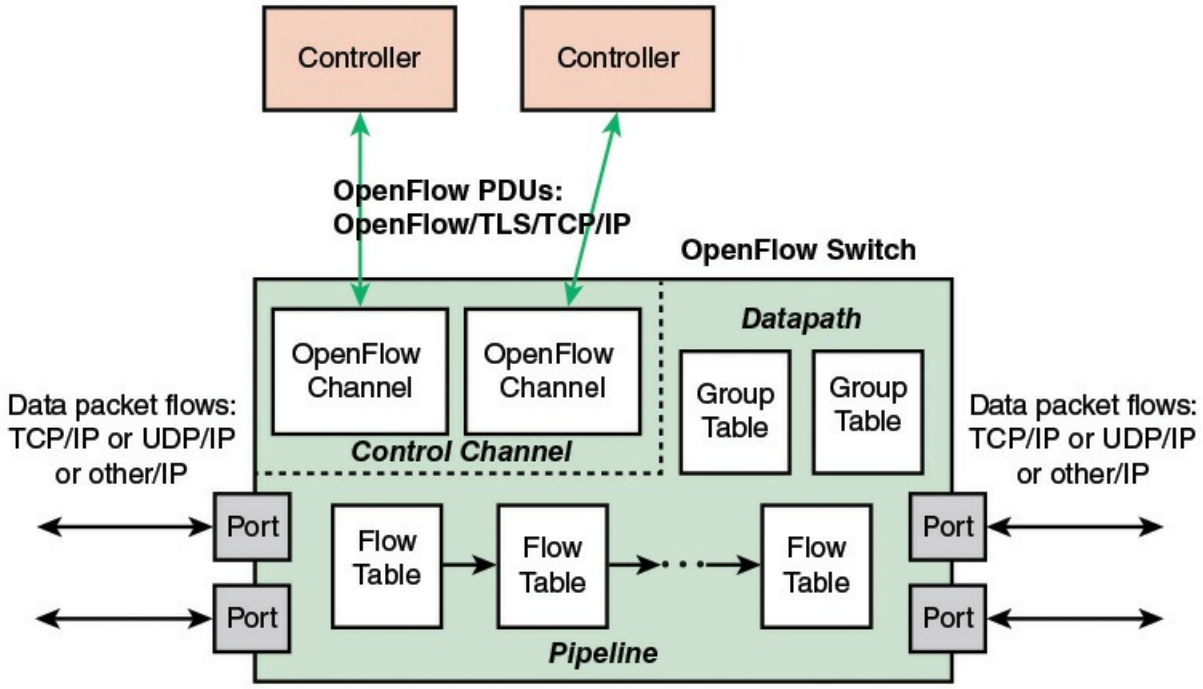


FIGURE 4.4 OpenFlow Switch

- **Physical port:** Corresponds to a hardware interface of the switch. For example, on an Ethernet switch, physical ports map one to one to the Ethernet interfaces.
- **Logical port:** Does not correspond directly to a hardware interface of the switch. Logical ports are higher-level abstractions that may be defined in the switch using non-OpenFlow methods (for example, link aggregation groups, tunnels, loopback interfaces). Logical ports may include packet encapsulation and may map to various physical ports. The processing done by the logical port is implementation dependent and must be transparent to OpenFlow processing, and those ports must interact with OpenFlow processing like OpenFlow physical ports.
- **Reserved port:** Defined by the OpenFlow specification. It specifies generic forwarding actions such as sending to and receiving from the controller, flooding, or forwarding using non-OpenFlow methods, such as "normal" switch processing.

Within each switch, a series of tables is used to manage the flows of packets through the switch.

The OpenFlow specification defines three types of tables in the logical switch architecture. A **flow table** matches incoming packets to a particular flow and specifies what functions are to be performed on the packets. There may be multiple flow tables that operate in a pipeline fashion, as explained subsequently. A flow table may direct a flow to a **group table**, which may trigger a variety of actions that affect one or more flows. A **meter table** can trigger a variety of performance-related actions on a flow. Meter tables are discussed in Chapter 10. Using the OpenFlow switch protocol, the controller can add, update, and delete flow entries in tables, both reactively (in response to packets) and proactively.

➔ *See Chapter 10, "Quality of Service"*

Before proceeding, it is helpful to define what is meant by the term *flow*. Curiously, this term is

not defined in the OpenFlow specification, nor is there an attempt to define it in virtually all of the literature on OpenFlow. In general terms, a flow is a sequence of packets traversing a network that share a set of header field values. For example, a flow could consist of all packets with the same source and destination IP addresses or all packets with the same virtual LAN (VLAN) identifier. The sections that follow provide a more specific definition of this concept.

Flow Table Structure

The basic building block of the logical switch architecture is the **flow table**. Each packet that enters a switch passes through one of more flow tables. Each flow table consists of a number of rows, called **entries**, consisting of seven components (see part a of Figure 4.5), as defined in the list that follows.

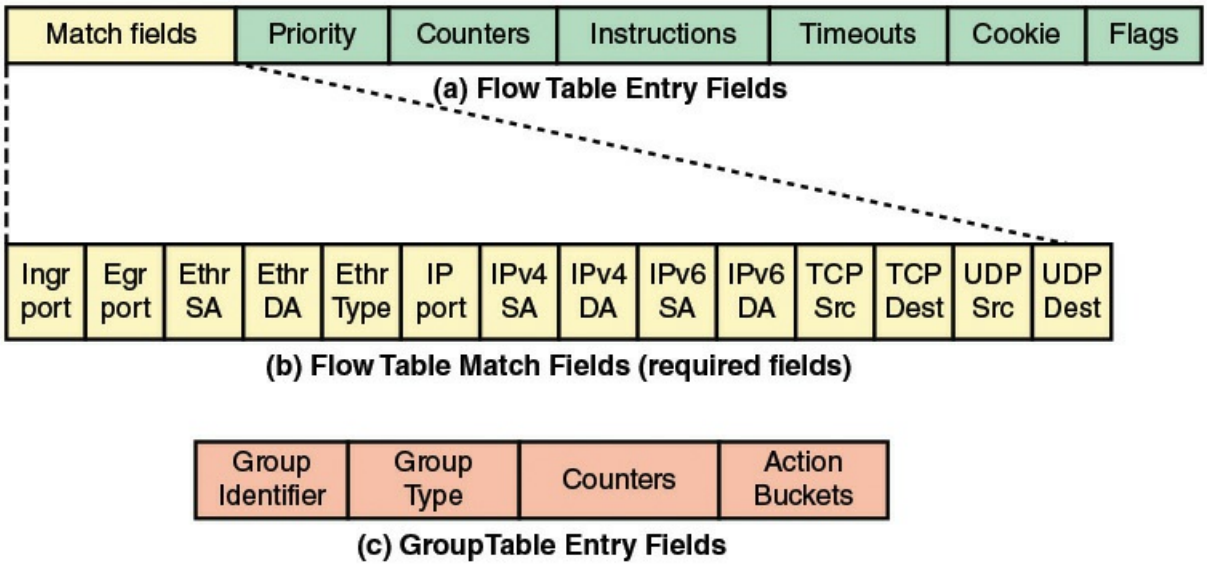


FIGURE 4.5 OpenFlow Table Entry Formats

- **Match fields:** Used to select packets that match the values in the fields.
- **Priority:** Relative priority of table entries. This is a 16-bit field with 0 corresponding to the lowest priority. In principle, there could be 2^{16} = 64k priority levels.
- **Counters:** Updated for matching packets. The OpenFlow specification defines a variety of counters. Table 4.1 lists the counters that must be supported by an OpenFlow switch.

Counter	Usage	Bit Length
Reference count (active entries)	Per flow table	32
Duration (seconds)	Per flow entry	32
Received packets	Per port	64
Transmitted packets	Per port	64
Duration (seconds)	Per port	32
Transmit packets	Per queue	64
Duration (seconds)	Per queue	32
Duration (seconds)	Per group	32
Duration (seconds)	Per meter	32

TABLE 4.1 Required OpenFlow Counters

- **Instructions:** Instructions to be performed if a match occurs.
- **Timeouts:** Maximum amount of idle time before a flow is expired by the switch. Each flow entry has an idle_timeout and a hard_timeout associated with it. A nonzero hard_timeout field causes the flow entry to be removed after the given number of seconds, regardless of how many packets it has matched. A nonzero idle_timeout field causes the flow entry to be removed when it has matched no packets in the given number of seconds.
- **Cookie:** 64-bit opaque data value chosen by the controller. May be used by the controller to filter flow statistics, flow modification and flow deletion; not used when processing packets.
- **Flags**: Flags alter the way flow entries are managed; for example, the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.

Match Fields Component

The match fields component of a table entry consists of the following required fields (see part b of Figure 4.5):

- **Ingress port:** The identifier of the port on this switch on which the packet arrived. This may be a physical port or a switch-defined virtual port. Required in ingress tables.
- **Egress port:** The identifier of the egress port from action set. Required in egress tables.
- **Ethernet source and destination addresses:** Each entry can be an exact address, a bitmasked value for which only some of the address bits are checked, or a wildcard value (match any value).
- **Ethernet type field:** Indicates type of the Ethernet packet payload.
- **IP:** Version 4 or 6.
- **IPv4 or IPv6 source address, and destination address:** Each entry can be an exact address, a bitmasked value, a subnet mask value, or a wildcard value.
- **TCP source and destination ports:** Exact match or wildcard value.

- **UDP source and destination ports:** Exact match or wildcard value.

The preceding match fields must be supported by any OpenFlow-compliant switch. The following fields may be optionally supported.

- **Physical port:** Used to designate underlying physical port when packet is received on a logical port.
- **Metadata:** Additional information that can be passed from one table to another during the processing of a packet. Its use is discussed subsequently.
- **VLAN ID and VLAN user priority:** Fields in the IEEE 802.1Q virtual LAN header. SDN support for VLANs is discussed in Chapter 8, "NFV Functionality."
- **IPv4 or IPv6 DS and ECN:** Differentiated Services and Explicit Congestion Notification fields.
- **SCTP source and destination ports:** Exact match or wildcard value for Stream Transmission Control Protocol.
- **ICMP type and code fields:** Exact match or wildcard value.
- **ARP opcode:** Exact match in Ethernet Type field.
- **Source and target IPv4 addresses in ARP payload:** Can be an exact address, a bitmasked value, a subnet mask value, or a wildcard value.
- **IPv6 flow label:** Exact match or wildcard.
- **ICMPv6 type and code fields:** Exact match or wildcard value.
- **IPv6 neighbor discovery target address:** In an IPv6 Neighbor Discovery message.
- **IPv6 neighbor discovery source and target addresses:** Link-layer address options in an IPv6 Neighbor Discovery message.
- **MPLS label value, traffic class, and BoS:** Fields in the top label of an MPLS label stack.
- **Provider bridge traffic ISID:** Service instance identifier.
- **Tunnel ID:** Metadata associated with a logical port.
- **TCP flags:** Flag bits in the TCP header. May be used to detect start and end of TCP connections.
- **IPv6 extension**: Extension header.

Thus, OpenFlow can be used with network traffic involving a variety of protocols and network services. Note that at the MAC/link layer, only Ethernet is supported. Therefore, OpenFlow as currently defined cannot control Layer 2 traffic over wireless networks.

Each of the fields in the match fields component either has a specific value or a wildcard value, which matches any value in the corresponding packet header field. A flow table may include a table-miss flow entry, which wildcards all match fields (every field is a match regardless of value) and has the lowest priority.

We can now offer a more precise definition of the term *flow*. From the point of view of an individual switch, a flow is a sequence of packets that matches a specific entry in a flow table.

The definition is packet oriented, in the sense that it is a function of the values of header fields of the packets that constitute the flow, and not a function of the path they follow through the network. A combination of flow entries on multiple switches defines a flow that is bound to a specific path.

Instructions Component

The instructions component of a table entry consists of a set of instructions that are executed if the packet matches the entry. Before describing the types of instructions, we need to define the terms *action* and *action set*. Actions describe packet forwarding, packet modification, and group table processing operations. The OpenFlow specification includes the following actions:

- **Output:** Forward packet to specified port. The port could be an output port to another switch or the port to the controller. In the latter case, the packet is encapsulated in a message to the controller.
- **Set-Queue:** Sets the queue ID for a packet. When the packet is forwarded to a port using the output action, the queue ID determines which queue attached to this port is used for scheduling and forwarding the packet. Forwarding behavior is dictated by the configuration of the queue and is used to provide basic QoS support. SDN support for QoS is discussed in Chapter 10.
- **Group:** Process packet through specified group.
- **Push-Tag/Pop-Tag:** Push or pop a tag field for a VLAN or Multiprotocol Label Switching (MPLS) packet.
- **Set-Field:** The various Set-Field actions are identified by their field type and modify the values of respective header fields in the packet.
- **Change-TTL:** The various Change-TTL actions modify the values of the IPv4 TTL (time to live), IPv6 hop limit, or MPLS TTL in the packet.
- **Drop**: There is no explicit action to represent drops. Instead, packets whose action sets have no output action should be dropped.

An action set is a list of actions associated with a packet that are accumulated while the packet is processed by each table and that are executed when the packet exits the processing pipeline.

The types of instructions can be grouped into four categories:

- **Direct packet through pipeline:** The Goto-Table instruction directs the packet to a table farther along in the pipeline. The Meter instruction directs the packet to a specified meter.
- **Perform action on packet:** Actions may be performed on the packet when it is matched to a table entry. The Apply-Actions instruction applies the specified actions immediately, without any change to the action set associated with this packet. This instruction may be used to modify the packet between two tables in the pipeline.
- **Update action set:** The Write-Actions instruction merges specified actions into the current action set for this packet. The Clear-Actions instruction clears all the actions in the action set.

- **Update metadata**: A metadata value can be associated with a packet. It is used to carry information from one table to the next. The Write-Metadata instruction updates an existing metadata value or creates a new value.

Flow Table Pipeline

A switch includes one or more flow tables. If there is more than one flow table, they are organized as a pipeline, with the tables labeled with increasing numbers starting with zero. The use of multiple tables in a pipeline, rather than a single flow table, provides the SDN controller with considerable flexibility.

The OpenFlow specification defines two stages of processing:

- **Ingress processing:** Ingress processing always happens, beginning with Table 0, and uses the identity of the input port. Table 0 may be the only table, in which case the ingress processing is simplified to the processing performed on that single table, and there is no egress processing.
- **Egress processing:** Egress processing is the processing that happens after the determination of the output port. It happens in the context of the output port. This stage is optional. If it occurs, it may involve one or more tables. The separation of the two stages is indicated by the numerical identifier of the first egress table. All tables with a number lower than the first egress table must be used as ingress tables, and no table with a number higher than or equal to the first egress table can be used as an ingress table.

Pipeline processing always starts with ingress processing at the first flow table; the packet must be first matched against flow entries of flow Table 0. Other ingress flow tables may be used depending on the outcome of the match in the first table. If the outcome of ingress processing is to forward the packet to an output port, the OpenFlow switch may perform egress processing in the context of that output port.

When a packet is presented to a table for matching, the input consists of the packet, the identity of the ingress port, the associated metadata value, and the associated action set. For Table 0, the metadata value is blank and the action set is null. At each table, processing proceeds as follows (see Figure 4.6):

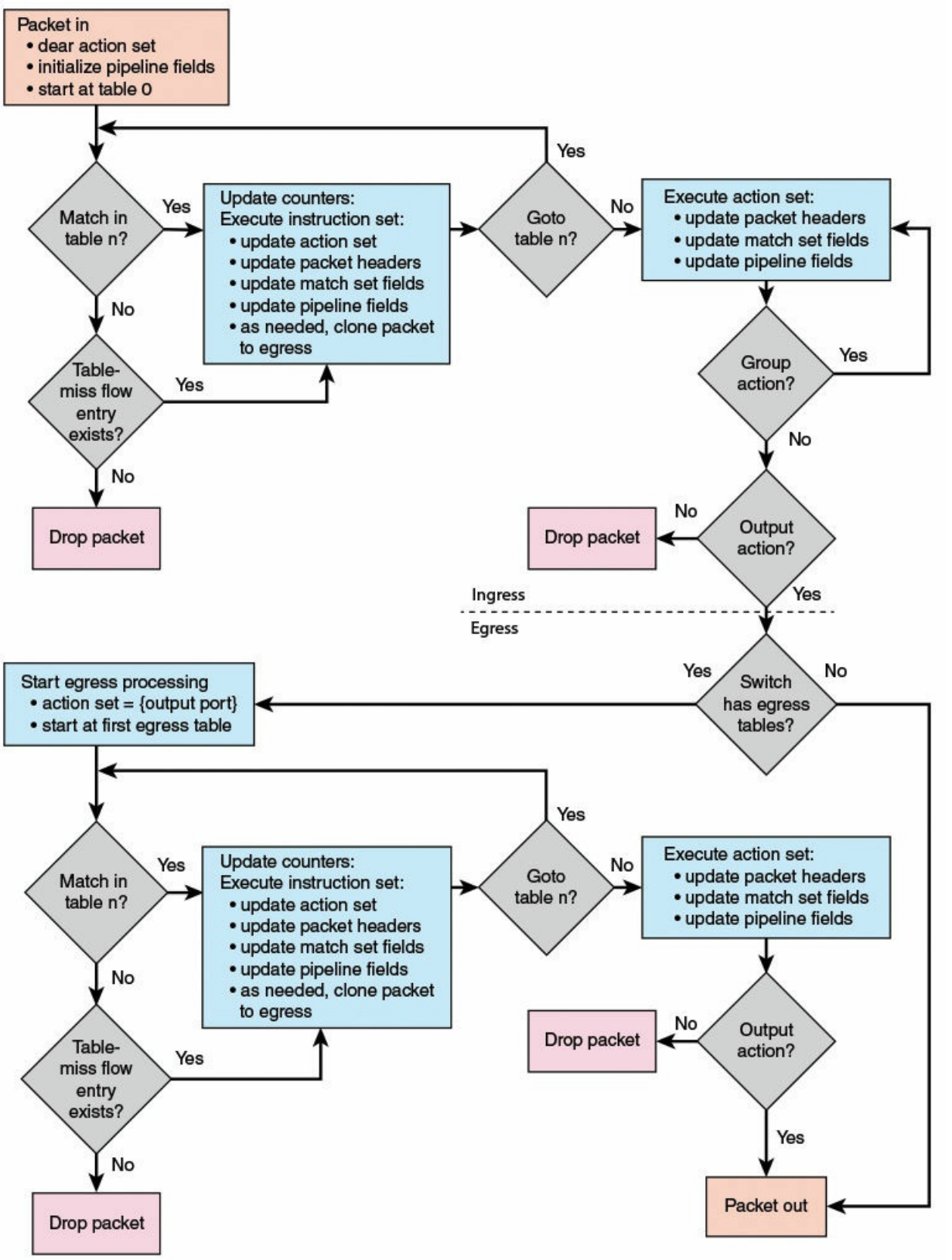


FIGURE 4.6 Simplified Flowchart Detailing Packet Flow Through an OpenFlow Switch

1. If there is a match on one or more entries, other than the table-miss entry, the match is defined to be with the highest-priority matching entry. As mentioned in the preceding discussion, the priority is a component of a table entry and is set via OpenFlow; the priority is determined by the user or application invoking OpenFlow. The following steps may then be performed:

a. Update any counters associated with this entry.

b. Execute any instructions associated with this entry. This may include updating the action set, updating the metadata value, and performing actions.

c. The packet is then forwarded to a flow table further down the pipeline, to the group table, to the meter table, or directed to an output port.

2. If there is a match only on a table-miss entry, the table entry may contain instructions, as with any other entry. In practice, the table-miss entry specifies one of three actions:

a. Send packet to controller. This will enable the controller to define a new flow for this and similar packets, or decide to drop the packet.

b. Direct packet to another flow table farther down the pipeline.

c. Drop the packet.

3. If there is no match on any entry and there is no table-miss entry, the packet is dropped.

For the final table in the pipeline, forwarding to another flow table is not an option. If and when a packet is finally directed to an output port, the accumulated action set is executed and then the packet is queued for output. Figure 4.7 illustrates the overall ingress pipeline process.

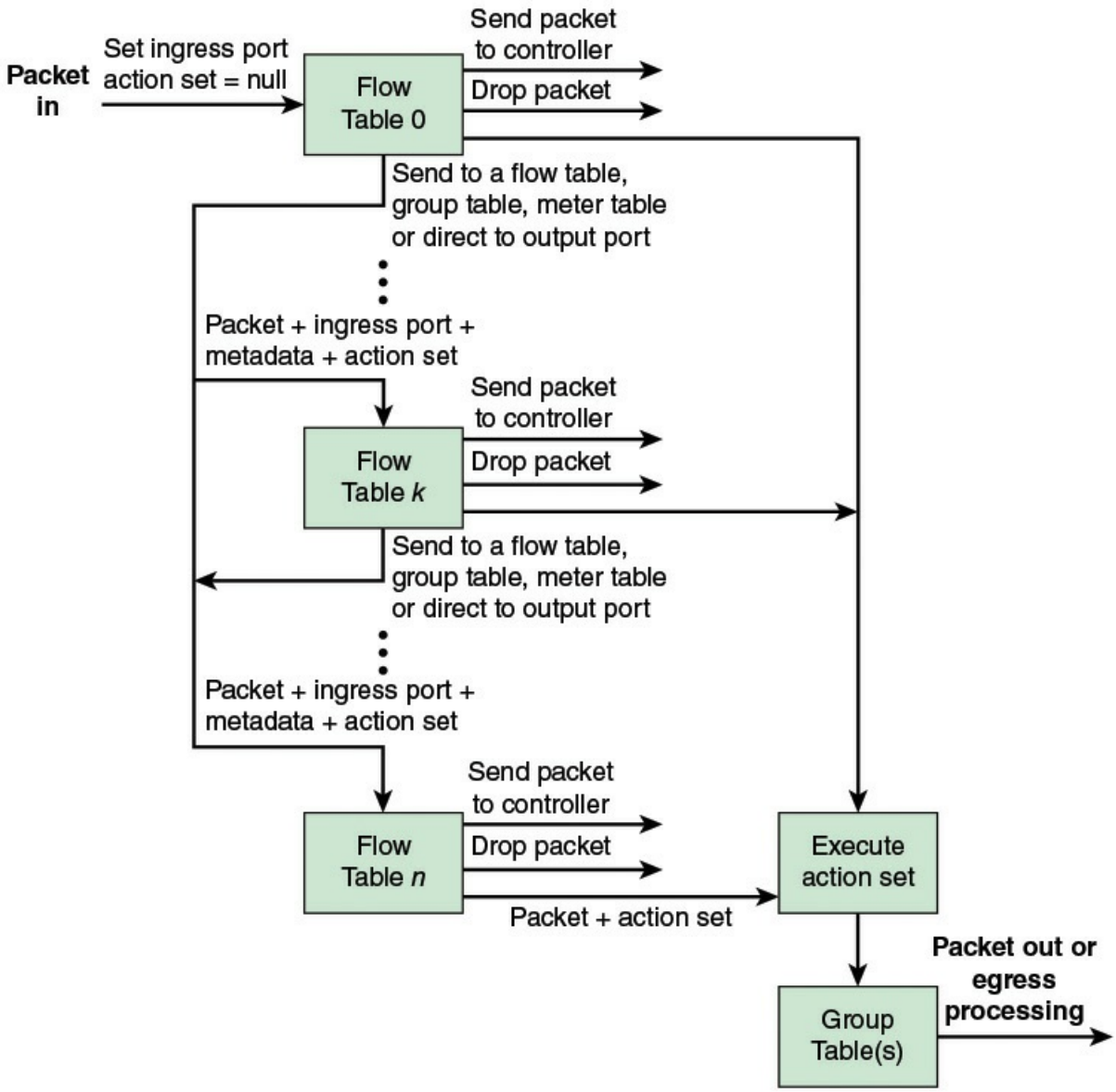


FIGURE 4.7 Packet Flow Through an OpenFlow Switch: Ingress Processing

If egress processing is associated with a particular output port, then after a packet is directed to an output port at the completion of the ingress processing, the packet is directed to the first flow table of the egress pipeline. Egress pipeline processing proceeds in the same fashion as for ingress processing, except that there is no group table processing at the end of the egress pipeline. Egress processing is shown in Figure 4.8.

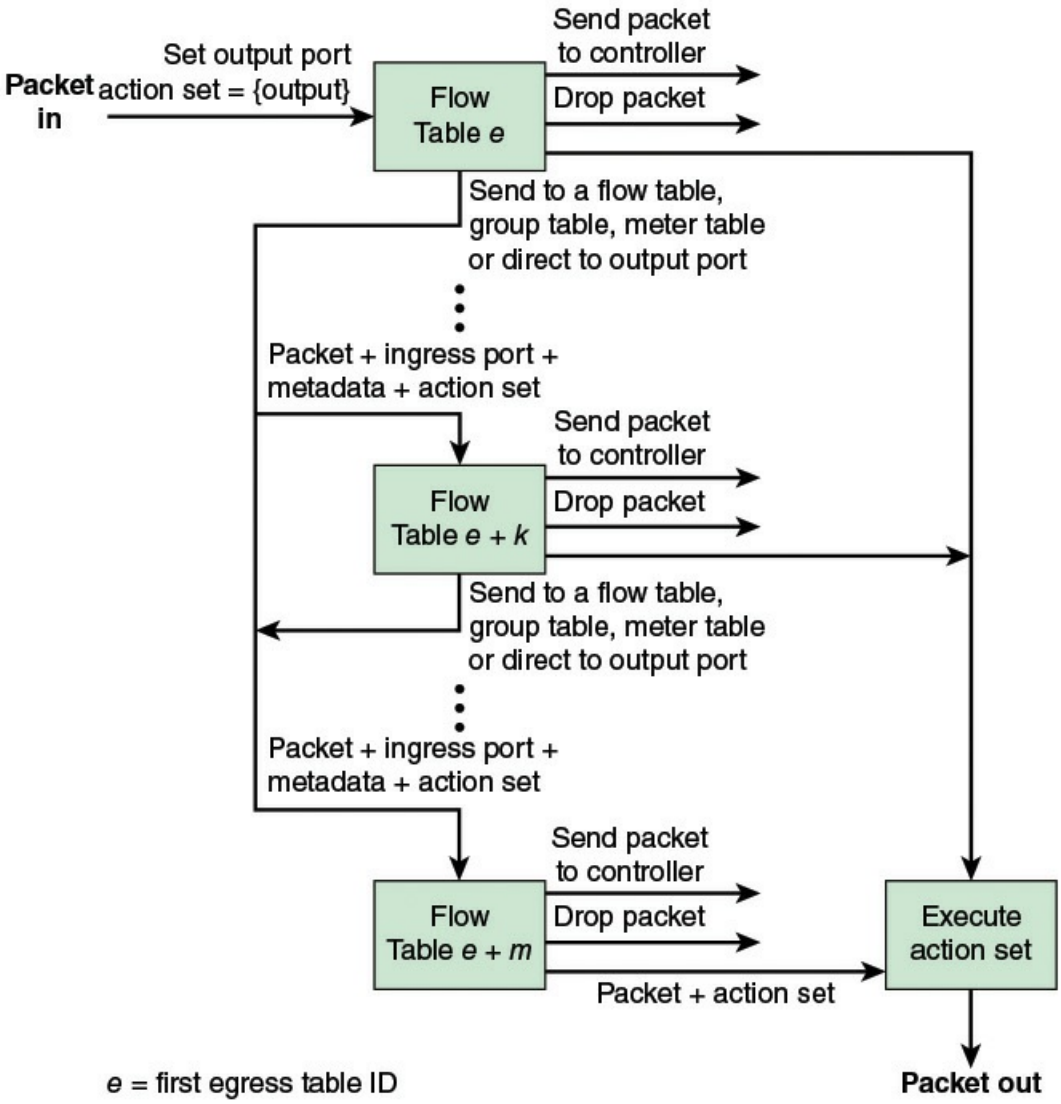


FIGURE 4.8 Packet Flow Through OpenFlow Switch: Egress Processing

The Use of Multiple Tables

The use of multiple tables enables the nesting of flows, or put another way, the breaking down of a single flow into a number of parallel subflows. Figure 4.9 illustrates this property. In this example, an entry in Table 0 defines a flow consisting of packets traversing the network from a specific source IP address to a specific destination IP address. Once a least-cost route between these two endpoints is established, it might make sense for all traffic between these two endpoints to follow that route, and the next hop on that route from this switch can be entered in Table 0. In Table 1, separate entries for this flow can be defined for different transport layer protocols, such as TCP and UDP. For these subflows, the same output port might be retained so that the subflows all follow the same route. However, TCP includes elaborate congestion control mechanisms not normally found with UDP, so it might be reasonable to handle the TCP and UDP subflows differently in terms of quality of service (QoS)-related parameters. Any of the

Table 1 entries could immediately route its respective subflow to the output port, but some or all of the entries may invoke Table 2, further dividing each subflow. The figure shows that the TCP subflow could be divided on the basis of the protocol running on top of TCP, such as Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP). Similarly, the UDP flow could be subdivided based on protocols running on UDP, such as Simple Network Management Protocol (SNMP). The figure also indicates other subflows at Table 1 and 2, which may be used for other purposes.

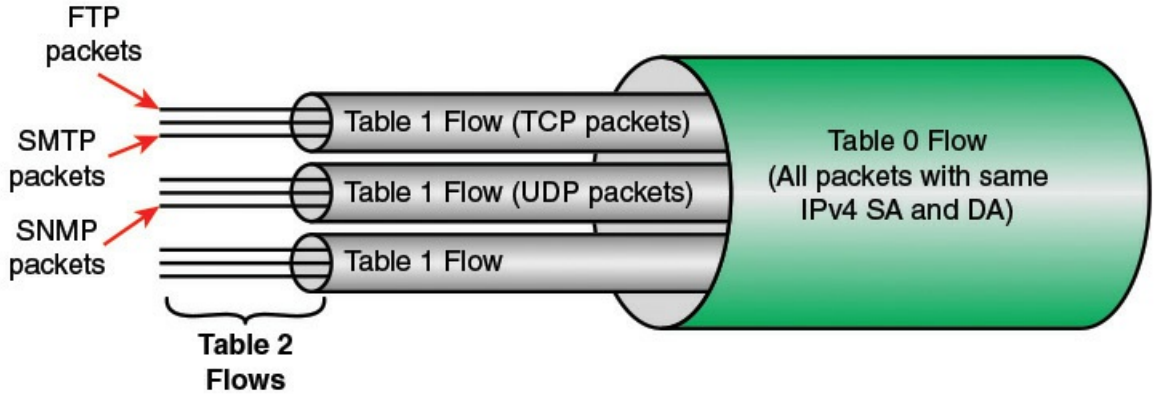


FIGURE 4.9 Example of Nested Flows

For this example, it would be possible to define each of these fine-grained subflows in Table 0. The use of multiple tables simplifies the processing in both the SDN controller and the OpenFlow switch. Actions such as next hop that apply to the aggregate flow can be defined once by the controller and examined and performed once by the switch. The addition of new subflows at any level involves less setup. Therefore, the use of pipelined, multiple tables increases the efficiency of network operations, provides granular control, and enables the network to respond to real-time changes at the application, user, and session levels.

Group Table

In the course of pipeline processing, a flow table may direct a flow of packets to the group table rather than another flow table. The group table and group actions enable OpenFlow to represent a set of ports as a single entity for forwarding packets. Different types of groups are provided to represent different forwarding abstractions, such as multicasting and broadcasting.

Each group table consists of a number of rows, called group entries, consisting of four components (refer back to part c of Figure 4.5):

- **Group identifier:** A 32-bit unsigned integer uniquely identifying the group. A **group** is defined as an entry in the group table.
- **Group type:** To determine group semantics, as explained subsequently.
- **Counters:** Updated when packets are processed by a group.
- **Action buckets:** An ordered list of action buckets, where each action bucket contains a set of actions to execute and associated parameters.

Each group includes a set of one or more action buckets. Each bucket contains a list of actions.

Unlike the action set associated with a flow table entry, which is a list of actions that accumulate while the packet is processed by each flow table, the action list in a bucket is executed when a packet reaches a bucket. The action list is executed in sequence and generally ends with the Output action, which forwards the packet to a specified port. The action list may also end with the Group action, which sends the packet to another group. This enables the chaining of groups for more complex processing.

A group is designated as one of the types depicted in Figure 4.10: all, select, fast failover, and indirect.

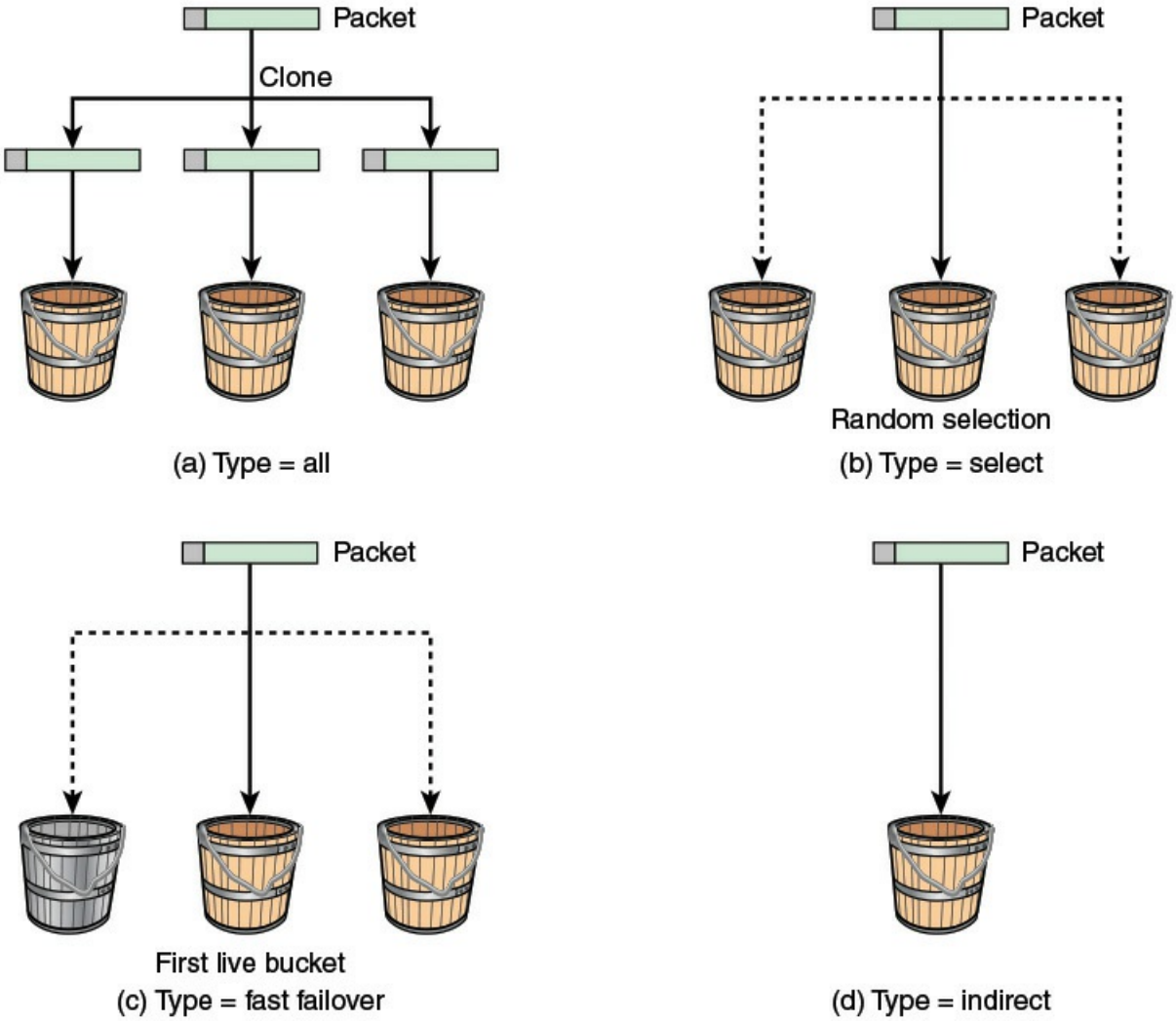


FIGURE 4.10 Group Types

The **all type** executes all the buckets in the group. Thus, each arriving packet is effectively cloned. Typically, each bucket will designate a different output port, so that the incoming packet is then transmitted on multiple output ports. This group is used for multicast or broadcast forwarding.

The **select type** executes one bucket in the group, based on a switch-computed selection algorithm (for example, hash on some user-configured tuple or simple round-robin). The selection algorithm should implement equal load sharing or, optionally, load sharing based on

bucket weights assigned by the SDN controller.

The **fast failover type** executes the first live bucket. Port liveness is managed by code outside of the scope of OpenFlow and may have to do with routing algorithms or congestion control mechanisms. The buckets are evaluated in order, and the first live bucket is selected. This group type enables the switch to change forwarding without requiring a round trip to the controller.

The three just-mentioned types all work with a single packet flow. The **indirect type** allows multiple packet flows (that is, multiple flow table entries) to point to a common group identifier. This type provides for more efficient management by the controller in certain situations. For example, suppose that there are 100 flow entries that have the same match value in the IPv4 destination address match field, but differ in some other match field, but all of them forward the packet to port X by including the action Output X on the action list. We can instead replace this action with the action Group GID, where GID is the ID of an indirect group entry that forwards the packet to port X. If the SDN controller needs to change from port X to port Y, it is not necessary to update all 100 flow table entries. All that is required is to update the group entry.

4.3 OpenFlow Protocol

The OpenFlow protocol describes message exchanges that take place between an OpenFlow controller and an OpenFlow switch. Typically, the protocol is implemented on top of TLS, providing a secure OpenFlow channel.

The OpenFlow protocol enables the controller to perform add, update, and delete actions to the flow entries in the flow tables. It supports three types of messages (see Table 4.2):

Message	Description
Controller to Switch	
Features	Request the capabilities of a switch. Switch responds with a features reply that specifies its capabilities.
Configuration	Set and query configuration parameters. Switch responds with parameter settings.
Modify-State	Add, delete, and modify flow/group entries and set switch port properties.
Read-State	Collect information from switch, such as current configuration, statistics, and capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role-Request	Set or query role of the OpenFlow channel. Useful when switch connects to multiple controllers.
Asynchronous-Configuration	Set filter on asynchronous messages or query that filter. Useful when switch connects to multiple controllers.
Asynchronous	
Packet-in	Transfer packet to controller.
Flow-Removed	Inform the controller about the removal of a flow entry from a flow table.
Port-Status	Inform the controller of a change on a port.
Role-Status	Inform controller of a change of its role for this switch from master controller to slave controller.
Controller-Status	Inform the controller when the status of an OpenFlow channel changes. This can assist failover processing if controllers lose the ability to communicate among themselves.
Flow-monitor	Inform the controller of a change in a flow table. Allows a controller to monitor in real time the changes to any subsets of the flow table done by other controllers.
Hello	Exchanged between the switch and controller upon connection startup.
Echo	Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply.
Error	Used by the switch or the controller to notify problems to the other side of the connection.
Experimenter	For additional functionality.

TABLE 4.2 OpenFlow Messages

- **Controller to switch:** These messages are initiated by the controller and, in some cases, require a response from the switch. This class of messages enables the controller to manage the logical state of the switch, including its configuration and details of flow and group table entries. Also included in this class is the Packet-out message. This message is

sent by the controller to a switch when that switch sends a packet to the controller and the controller decides not to drop the packet but to direct it to a switch output port.

- **Asynchronous:** These types of messages are sent without solicitation from the controller. This class includes various status messages to the controller. Also included is the Packet-in message, which may be used by the switch to send a packet to the controller when there is no flow table match.
- **Symmetric**: These messages are sent without solicitation from either the controller or the switch. They are simple yet helpful. Hello messages are typically sent back and forth between the controller and switch when the connection is first established. Echo request and reply messages can be used by either the switch or controller to measure the latency or bandwidth of a controller-switch connection or just verify that the device is up and running. The Experimenter message is used to stage features to be built in to future versions of OpenFlow.

In general terms, the OpenFlow protocol provides the SDN controller with three types of information to be used in managing the network:

- **Event-based messages:** Sent by the switch to the controller when a link or port change occurs.
- **Flow statistics:** Generated by the switch based on traffic flow. This information enables the controller to monitor traffic, reconfigure the network as needed, and adjust flow parameters to meet QoS requirements.
- **Encapsulated packets:** Sent by the switch to the controller either because there is an explicit action to send this packet in a flow table entry or because the switch needs information for establishing a new flow.

The OpenFlow protocol enables the controller to manage the logical structure of a switch, without regard to the details of how the switch implements the OpenFlow logical architecture.

4.4 Key Terms

After completing this chapter, you should be able to define the following terms.

action bucket

action list

action set

egress table

flow

flow table

group table

ingress table

match fields

OpenFlow action

OpenFlow instruction

OpenFlow message

OpenFlow port

OpenFlow switch

SDN data plane

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.

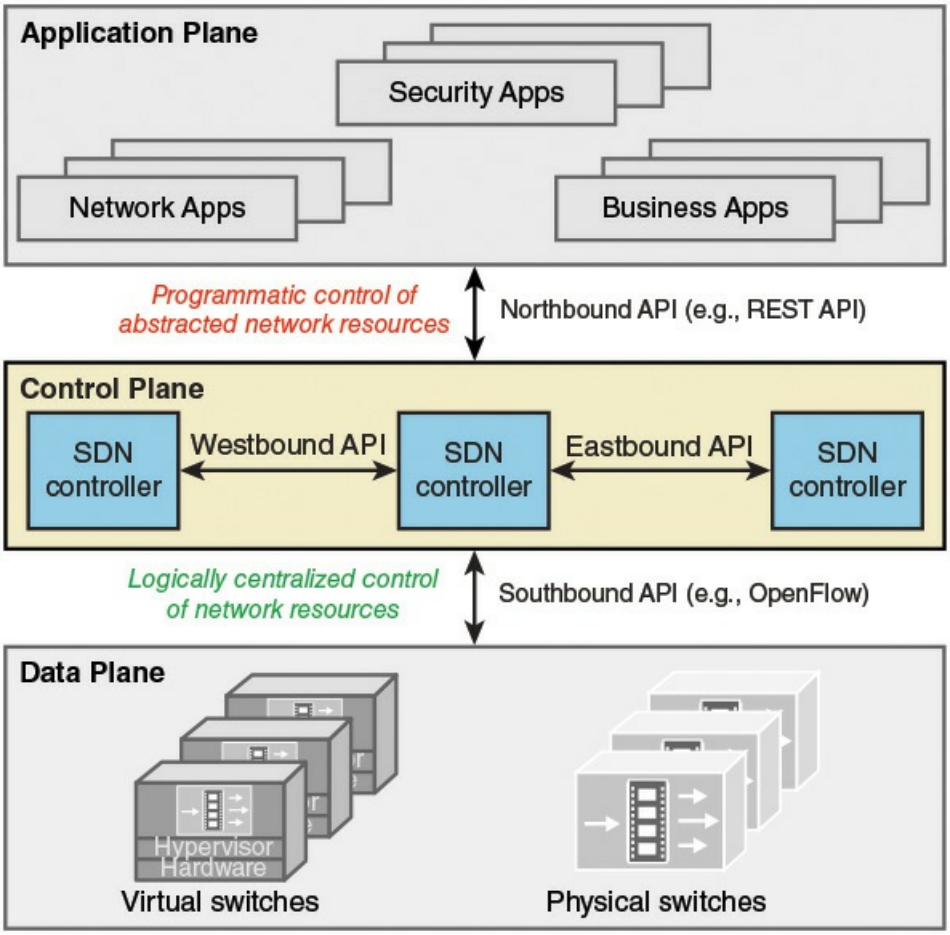


FIGURE 5.1 SDN Architecture

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:

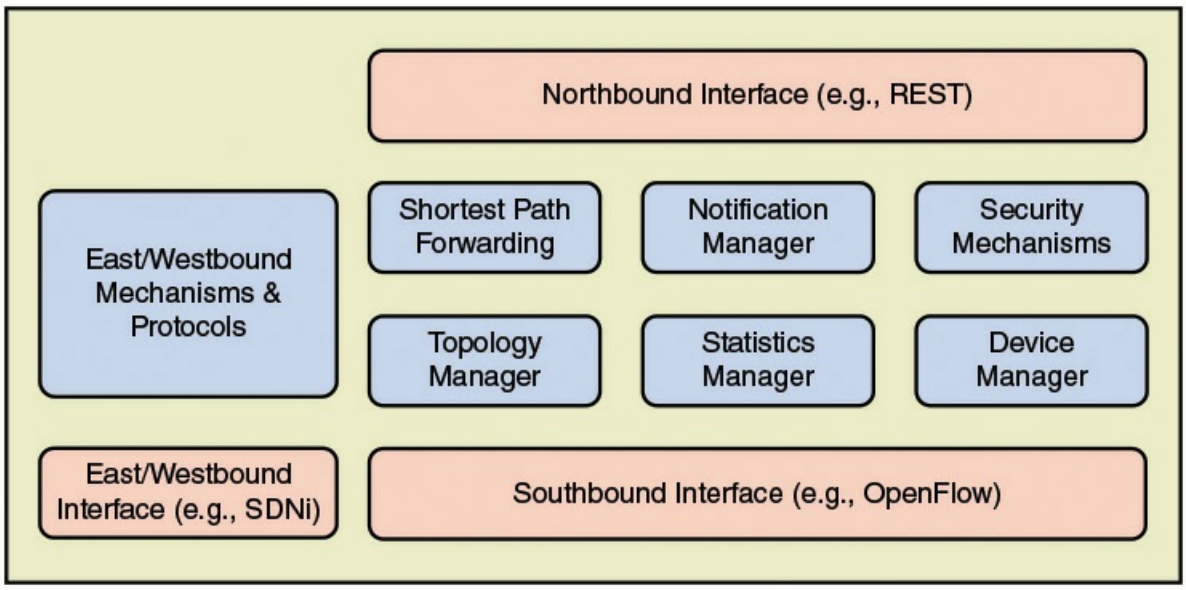


FIGURE 5.2 SDN Control Plane Functions and Interfaces

- **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.

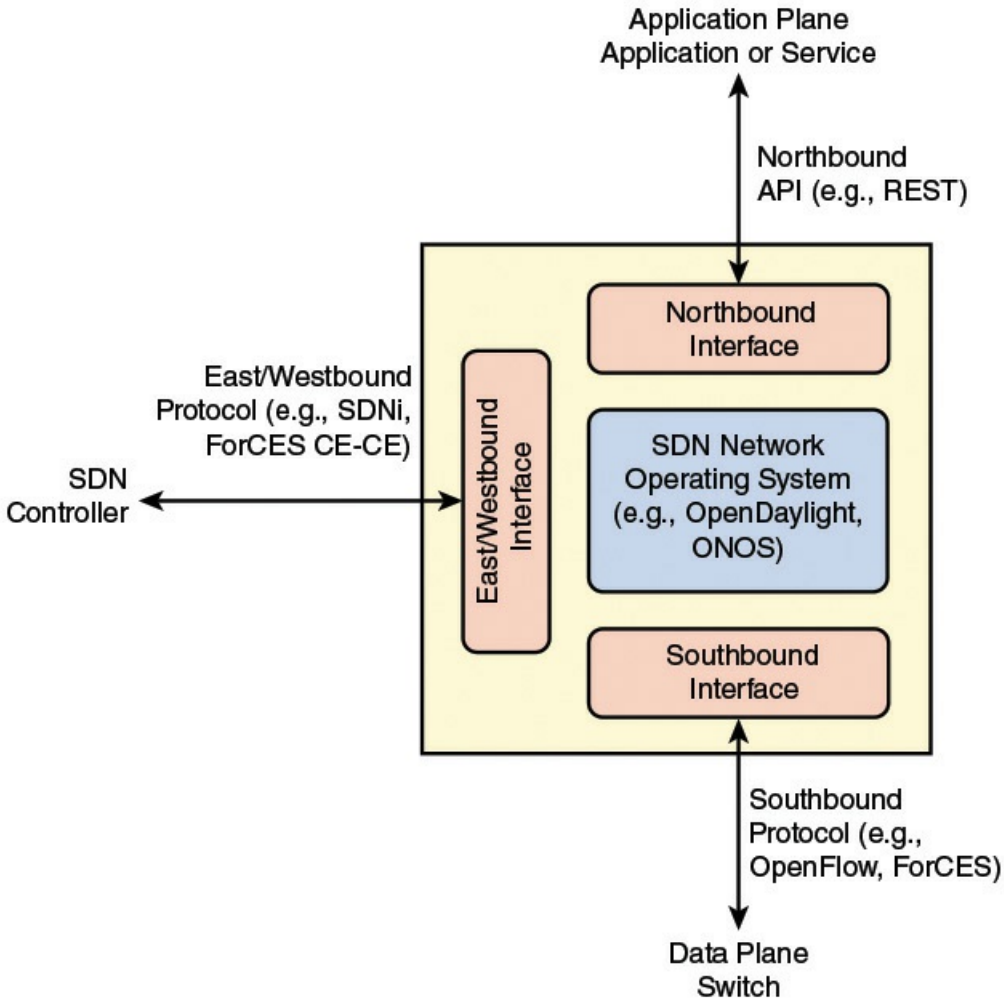


FIGURE 5.3 SDN Controller Interfaces

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.

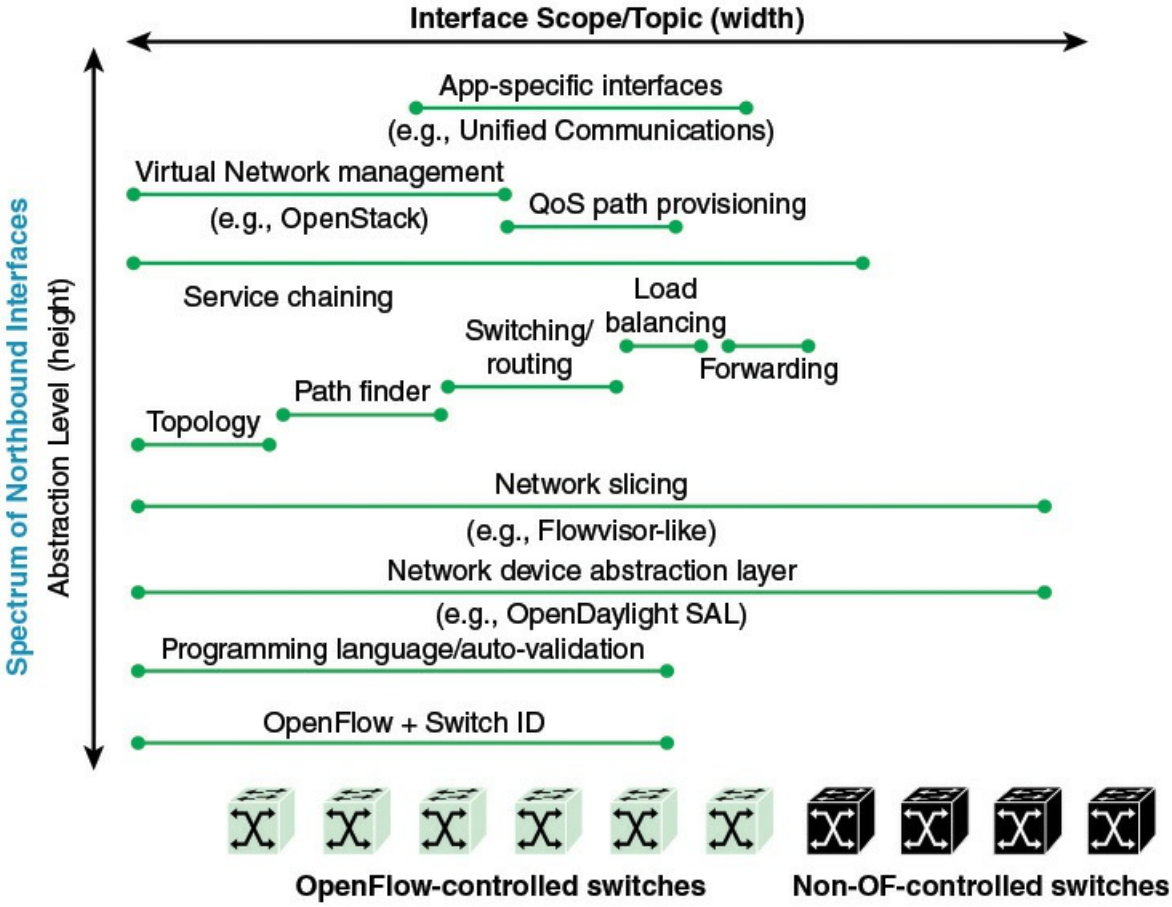


FIGURE 5.4 Latitude of Northbound Interfaces

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.

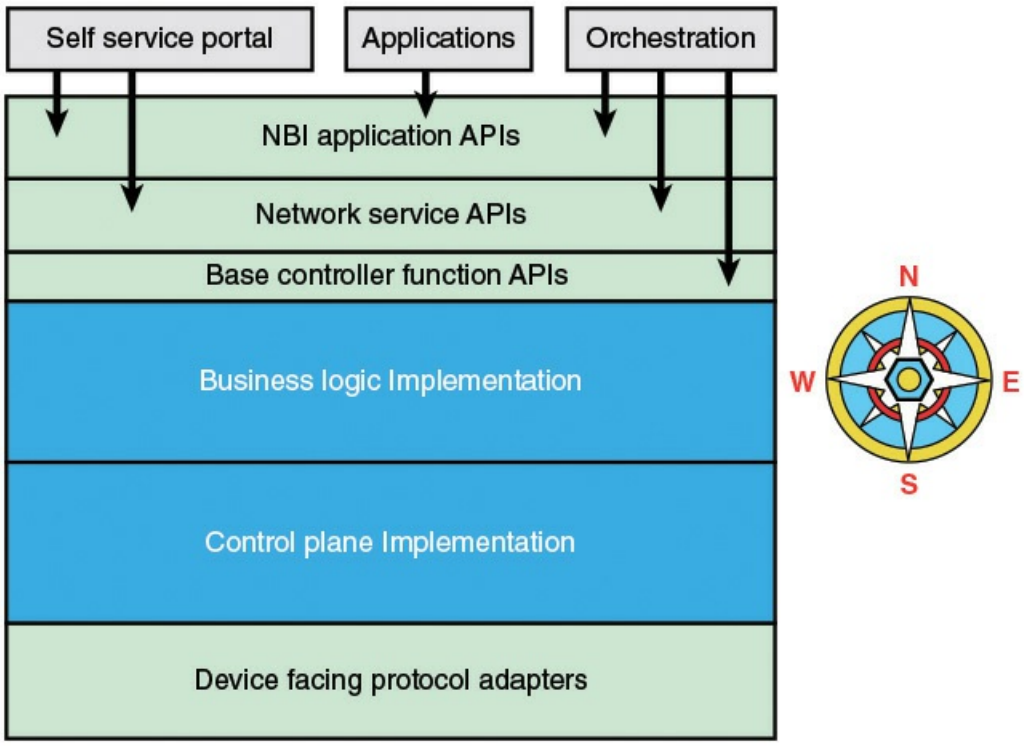


FIGURE 5.5 SDN Controller APIs

- **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 algorithm for designing routes through the network. Recall from Chapter 2, "Requirements and Technology," that there are two categories of routing protocols: interior router protocols (IRPs) that operate within an autonomous system (AS), and exterior router protocols (ERPs) that operate between autonomous systems.

➔ *See Section 2.4, "Routing"*

An IRP is concerned with discovering the topology of routers within an AS and then determining

the best route to each destination based on different metrics. Two widely used IRPs are Open Shortest Path First (OSPF) Protocol and Enhanced Interior Gateway Routing Protocol (EIGRP). An ERP need not collect as much detailed traffic information. Rather, the primary concern with an ERP is to determine reachability of networks and end systems outside of the AS. Therefore, the ERP is typically executed only in edge nodes that connect one AS to another. Border Gateway Protocol (BGP) is commonly used for the ERP.

Traditionally, the routing function is distributed among the routers in a network. Each router is responsible for building up an image of the topology of the network. For interior routing, each router as well must collect information about connectivity and delays and then calculate the preferred route for each IP destination address. However, in an SDN-controlled network, it makes sense to centralize the routing function within the SDN controller. The controller can develop a consistent view of the network state for calculating shortest paths, and can implement application-aware routing policies. The data plane switches are relieved of the processing and storage burden associated with routing, leading to improved performance.

The centralized routing application performs two distinct functions: link discovery and topology manager.

For **link discovery**, the routing function needs to be aware of links between data plane switches. Note that in the case of an internetwork, the links between routers are networks, whereas for Layer 2 switches, such as Ethernet switches, the links are direct physical links. In addition, link discovery must be performed between a router and a host system and between a router in the domain of this controller and a router in a neighboring domain. Discovery is triggered by unknown traffic entering the controller's network domain either from an attached host or from a neighboring router.

The **topology manager** maintains the topology information for the network and calculates routes in the network. Route calculation involves determining the shortest path between two data plane nodes or between a data plane node and a host.

5.2 ITU-T Model

Before proceeding to a discussion of an SDN controller design, it will be useful to look at the SDN high-level architecture defined in ITU-T Y.3300 (see Figure 5.6). As was depicted in Figure 3.3, the Y.3300 model consists of three layers, or planes: application, control, and resource. As defined in Y.3300, the **application layer** is where SDN applications specify network services or business applications by defining a service-aware behavior of network resources. The applications interact with the SDN control layer via APIs that form an application-control interface. The applications make use of an abstracted view of the network resources provided by the SDN control layer by means of information and data models exposed via the APIs.

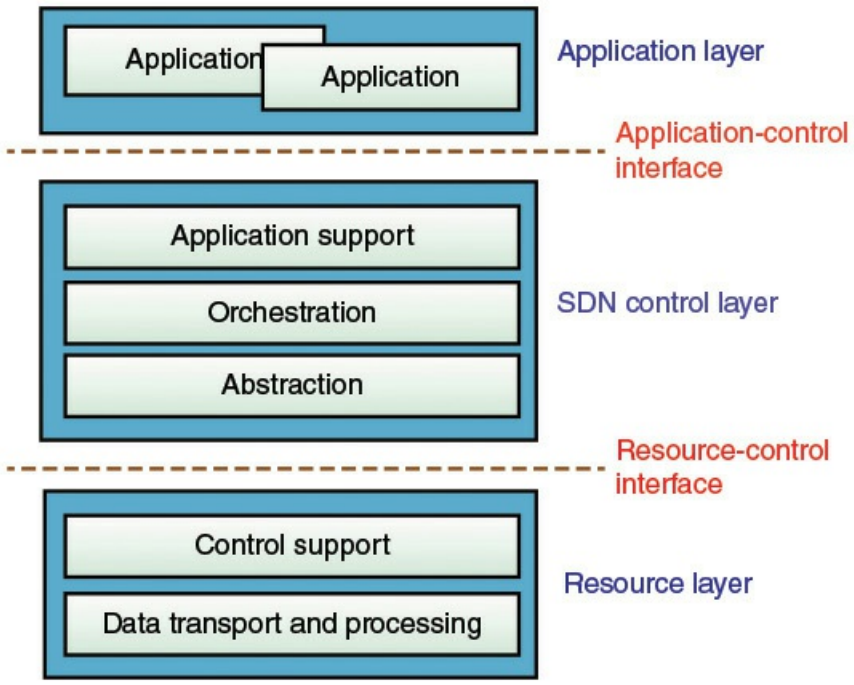


FIGURE 5.6 High-Level Architecture of SDN (ITU-T Y.3300)

The control layer provides a means to dynamically control the behavior of network resources, as instructed by the application layer. The control layer can be viewed as having the following sublayers:

- **Application support:** Provides an API for SDN applications to access network information and program application-specific network behavior.
- **Orchestration:** Provides the automated control and management of network resources and coordination of requests from the application layer for network resources. Orchestration encompasses physical and virtual network topologies, network elements, traffic control, and other network-related aspects.
- **Abstraction:** Interacts with network resources, and provides an abstraction of the network resources, including network capabilities and characteristics, to support management and orchestration of physical and virtual network resources. Such abstraction relies upon standard information and data models and is independent of the underlying transport infrastructure.

The **resource layer** consists of an interconnected set of data plane forwarding elements (switches). Collectively, these switches perform the transport and processing of data packets according to decisions made by the SDN control layer and forwarded to the resource layer via the resource-control interface. Most of this control is on behalf of applications. However, the SDN control layer, on its own behalf, may execute control of the resource layer for the sake of performance (for example, traffic engineering). The resource layer can be viewed as having the following sublayers:

- **Control support:** Supports programmability of resource-layer functions via the resource-control interface.

- **Data transport and processing:** Provides data forwarding and data routing functions.

The SDN design philosophy seeks to minimize the complexity and processing burden on the data switches. Accordingly, we can expect that many, if not most, of the commercial SDN switches will be equipped with a single southbound interface, such as OpenFlow, for simplicity of implementation and configuration. But different switches may support different southbound interfaces to the controller. Therefore, the SDN controller should support multiple protocols and interfaces to the data plane and be able to abstract all of these interfaces to a uniform network model to be used the application layer.

5.3 OpenDaylight

The OpenDaylight Project is an open source project hosted by the Linux Foundation and includes the involvement of virtually every major networking organization, including users of SDN technology and vendors of SDN products. Rather than hammer out new standards, the project aims to produce an extensible, open source, virtual networking platform atop such existing standards as OpenFlow. The approach of OpenDaylight is to enable industry participants to come together to develop core open source modules collaboratively, around which participants can add unique value. The goal is a common and open SDN platform for developers to utilize, contribute to, and build commercial products and technologies upon.

OpenDaylight

It is worthwhile to examine OpenDaylight in some detail, as it gives the reader a good idea of the scope of functionality of a typical SDN controller.

OpenDaylight Architecture

Figure 5.7 provides a top-level view of the OpenDaylight architecture. It consists of five logical layers, as further described in the list that follows.

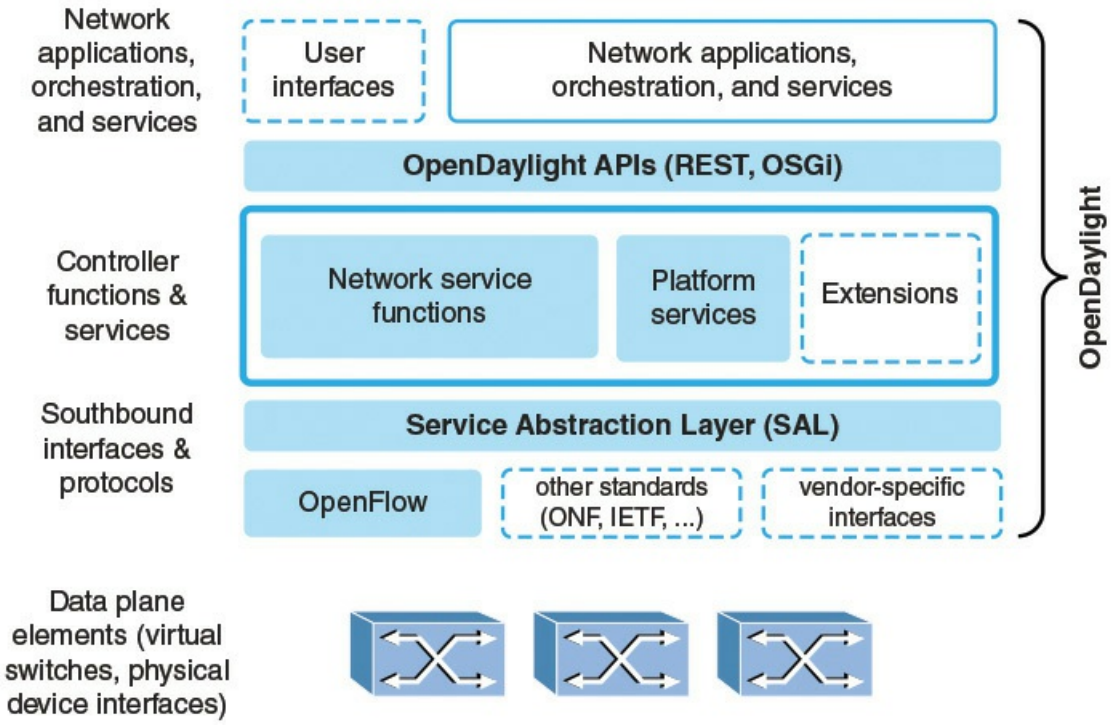


FIGURE 5.7 OpenDaylight Architecture

- **Network applications, orchestration, and services:** Consists of business and network logic applications that control and monitor network behavior. These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate the new rules, if any, throughout the network.
- **APIs:** A set of common interfaces to OpenDaylight controller functions. OpenDaylight supports the **Open Service Gateway Initiative (OSGi)** framework and bidirectional REST for the northbound API. The OSGi framework is used for applications that will run in the same address space as the controller, while the REST (web-based) API is used for applications that do not run in the same address space (or even necessarily on the same machine) as the controller.
- **Controller functions and services:** SDN control plane functions and services.
- **Service abstraction layer (SAL):** Provides a uniform view of data plane resources, so that control plane functions can be implemented independent of the specific southbound interface and protocol.
- **Southbound interfaces and protocols:** Supports OpenFlow, other standard southbound protocols, and vendor-specific interfaces.

There are several noteworthy aspects to the OpenDaylight architecture. First, OpenDaylight encompasses both control plane and application plane functionality. Thus, OpenDaylight is more than just an SDN controller implementation. This enables enterprise and telecommunications network managers to host open source software on their own servers to construct an SDN configuration. Vendors can use this software to create products with value-added additional application plane functions and services.

A second significant aspect of the OpenDaylight design is that it is not tied to OpenFlow or any other specific southbound interface. This provides greater flexibility in constructing SDN network configurations. The key element in this design is the SAL, which enables the controller to support multiple protocols on the southbound interface and provide consistent services for controller functions and for SDN applications. Figure 5.8 illustrates the operation of the SAL. The OSGi framework provides for dynamically linking plug-ins for the available southbound protocols. The capabilities of these protocols are abstracted to a collection of features that can be invoked by control plane services via a services manager in the SAL. The services manager maintains a registry that maps service requests to feature requests. Based on the service request, the SAL maps to the appropriate plug-in and thus uses the most appropriate southbound protocol to interact with a given network device.

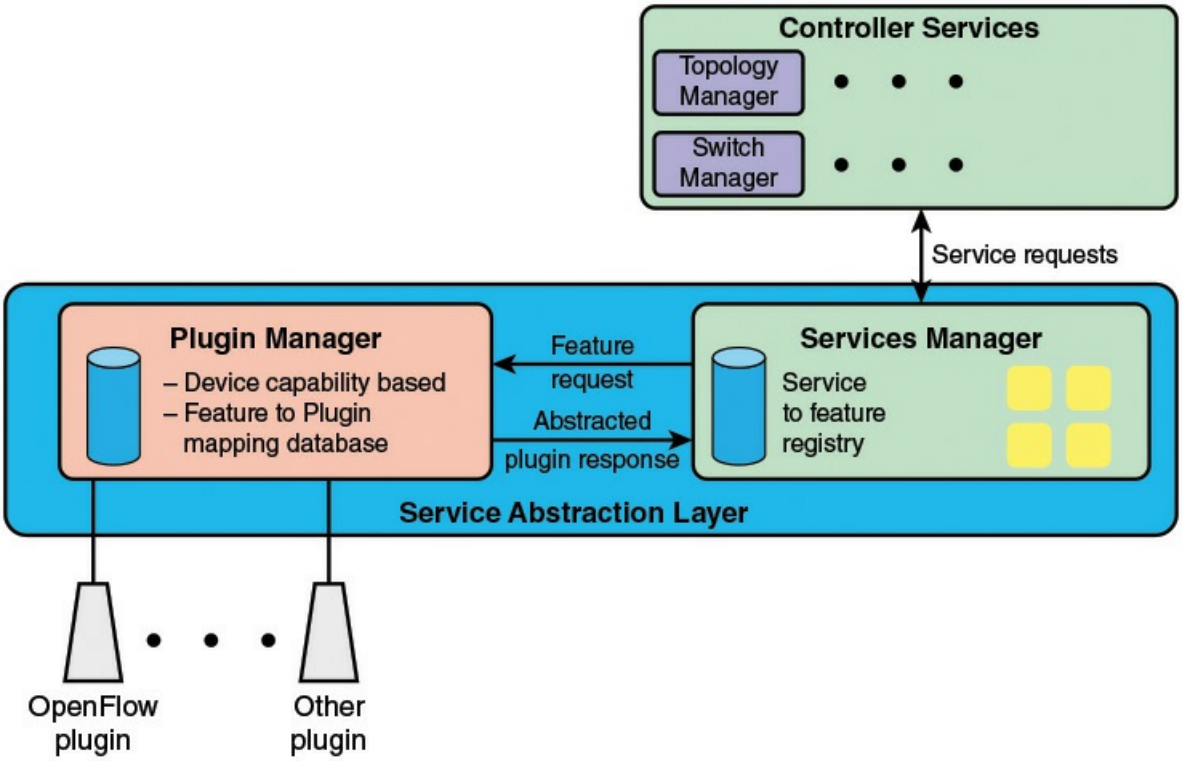


FIGURE 5.8 Service Abstraction Layer Model

The emphasis in the OpenDaylight project is that the software suite be modular, pluggable, and flexible. All of the code is implemented in Java and is contained within its own Java Virtual Machine (JVM). As such, it can be deployed on any hardware and operating system platform that supports Java.

OpenDaylight Helium

At the time of this writing, the most recent release of OpenDaylight is the Helium release, illustrated in Figure 5.9. The controller platform (exclusive of applications, which may also run on the controller) consists of a growing collection of dynamically pluggable modules, each of which performs one or more SDN-related functions and services. Five modules are considered

base network service functions, likely to be included in any OpenDaylight implementation, as described in the list that follows.

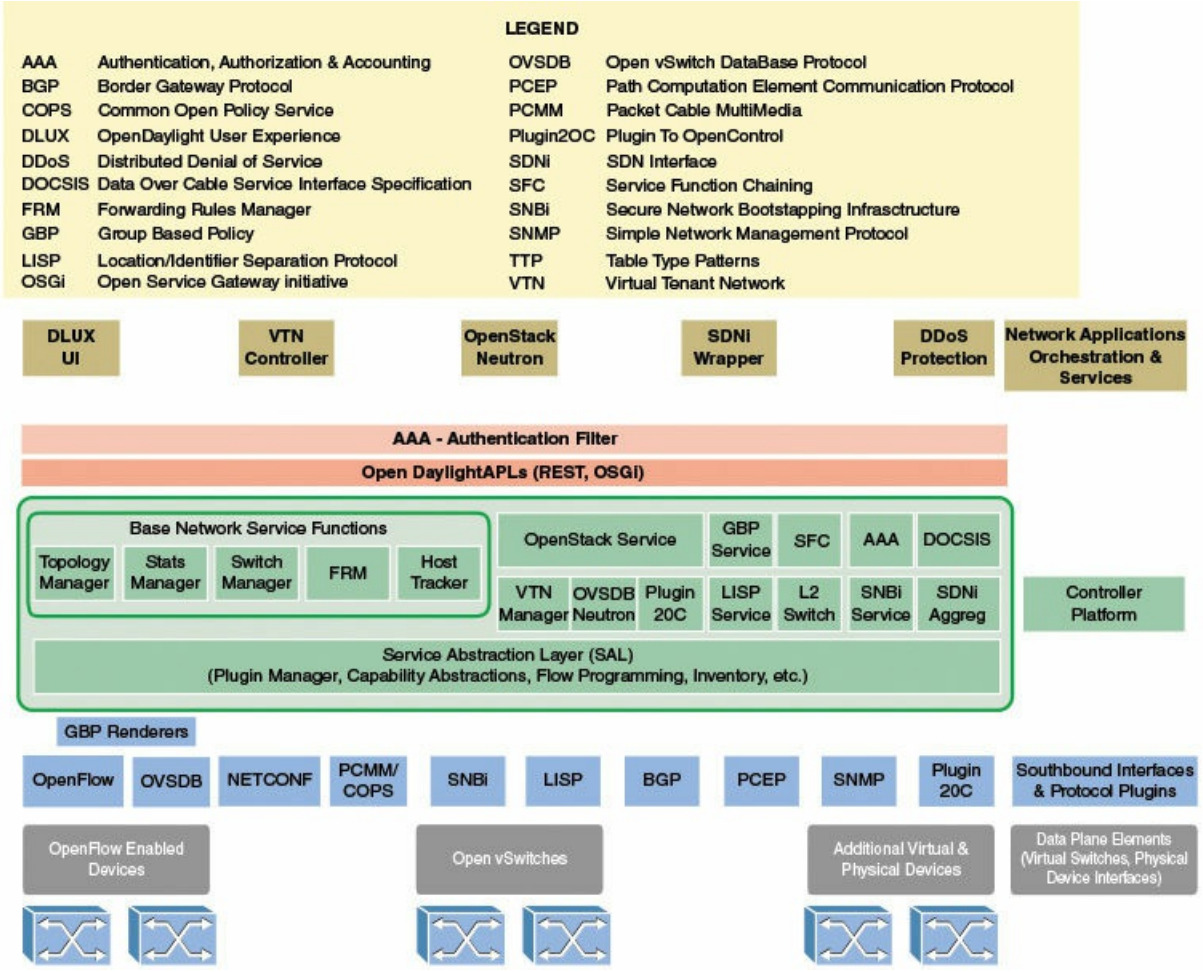


FIGURE 5.9 OpenDaylight Structure (Helium)

- **Topology manager:** A service for learning the network layout by subscribing to events of node addition and removal and their interconnection. Applications requiring network view can use this service.
- **Statistics manager:** Collects switch-related statistics, including flow statistics, node connector, and queue occupancy.
- **Switch manager:** Holds the details of the data plane devices. As a switch is discovered, its attributes (for example, what switch/router it is, software version, capabilities) are stored in a database by the switch manager.
- **Forwarding rules manager:** Installs routes and tracks next-hop information. Works in conjunction with switch manager and topology manager to register and maintain network flow state. Applications using this need not have visibility of network device specifics.
- **Host tracker:** Tracks and maintains information about connected hosts.

To augment these base services, a number of other modules have been developed to enable implementation of more sophisticated and feature-rich controllers, as described in Table 5.1.

Feature	Description
Southbound Interfaces and Protocol Plug-Ins	
OpenFlow	The OpenFlow protocol.
Open vSwitch DataBase Protocol (OVSDB)	A network configuration protocol for virtual switching.
NETCONF	A network management protocol developed by IETF. NETCONF provides mechanisms to install, manipulate, and delete the configuration of network devices.
Packet Cable MultiMedia (PCMM)	PCMM provides an interface to control and manage service flow for cable modem network elements.
Secure Network Bootstrapping Infrastructure (SNBi)	Provides a secure channel that can be used by other applications to securely connect to various devices.
Location/Identifier Separation Protocol (LISP)	An IETF proposed standard that separates the current IP address into two separate name spaces to show an IP location and identifier separately.
BGP	For routing service across multicarrier networks, the BGP-Link State (BGP-LS) protocol is run on the controller. BGP-LS learns the route information from adjacent autonomous systems and builds a consolidated and centralized routing database.
Path Computation Element Communication Protocol (PCEP)	Used to provision virtual private network (VPN) configuration information.
Simple Network Management Protocol (SNMP)	A collection of specifications for network management that include the protocol itself, the definition of a database, and associated concepts. SNMP enables a management station to monitor and control a network of devices.

Plugin2OC	A plug-in to the OpenContrail platform. OpenContrail is an open source project whose focus is as a network platform for cloud infrastructure.
Controller Modules	
OpenStack service	OpenStack is an open source project to develop a massively scalable cloud operating system platform. SDN architectures that enable virtual networks are a good fit for OpenStack.
Group Based Policy (GBP) service	GBP is an application-centric policy model for OpenDaylight that separates information about application connectivity requirements from information about the underlying details of the network infrastructure.
Service function chaining (SFC)	An IETF proposed standard for combining service functions. In the SFC model, service functions, whether physical or virtualized, are not required to reside on the direct data path, and traffic is instead steered through required service functions, wherever they are deployed.
Authentication, authorization, and accounting (AAA)	Provides three basic security services: Authentication means to authenticate the identity of both human and machine users independent of choice of binding (direct or federated); authorization means to authorize human or machine user access to resources, including RPCs, notification subscriptions, and subsets of the data tree; accounting means to record and access the records of human or machine user access to resources, including RPCs, notifications, and subsets of the data tree.
Data Over Cable Service Interface Specification (DOCSIS)	A standard protocol stack for cable modem interface to digital networks.
Virtual Tenant Network (VTN) Manager	VTN is an application that provides multitenant virtual networks on an SDN controller.

Open vSwitch DataBase (OVSDB) Protocol Neutron	Neutron interface to OVSDB.
Plugin2OC	Service interface to the Plugin2OC plug-in.
LISP service	Service interface to LISP plug-in.
L2Switch	This is an OSI Layer 2 switch routing functionality. The basic concept is to use controller intelligence to design routes through an Ethernet switched network that avoid the use of broadcast when the route to the destination is known.
SNBi service	Service interface to the SNBi plug-in.
SDN interface (SDNi) aggregator	The OpenDaylight- SDN Interface Application project aims at enabling inter-SDN controller communication by developing SDNi (Software-Defined Networking interface) as an application (ODL-SDNi App). This service acts as an aggregator for collecting network information such as topology, statistics, and host identity and location.
AAA authentication filter	Intercept requests or responses to or from the controller to verify tokens.
Network Applications, Orchestration, and Services	
DLUX UI	A JavaScript-based stateless user interface that provides a consistent and user-friendly interface to interact with OpenDaylight projects and base controller. DLUX is a web-based interface that provides easy access to a model of the network controlled by this OpenDaylight controller.
VTN Controller	Provides API of VTN for users.
OpenStack Neutron	Neutron is a subsystem of OpenStack that allows for model-based integration of the network via APIs (in support of the core Infrastructure as a Service [IaaS] capabilities).
SDNi wrapper	Responsible for sharing and collecting information to/from federated controllers.
Distributed denial-of-service (DDoS) protection	Instructs an OpenFlow controller to program virtual and physical switches to be OpenFlow counters that collect statistics on network traffic. The application learns baseline traffic patterns and then watches for anomalies indicative of a network-level DDoS attack. If the application detects an attack, it instructs the OpenFlow controller to send suspect flows to specialized mitigation appliances to filter out malicious traffic.

TABLE 5.1 OpenDaylight Modules

5.4 REST

REpresentational State Transfer (REST) is an architectural style used to define APIs. This has

become a standard way of constructing northbound APIs for SDN controllers. A REST API, or an API that is **RESTful** (adheres to the constraints of REST) is not a protocol, language, or established standard. It is essentially six constraints that an API must follow to be RESTful. The objective of these constraints is to maximize the scalability and independence/interoperability of software interactions, and to provide for a simple means of constructing APIs.

REST Constraints

REST assumes that the concepts of web-based access are used for interaction between the application and the service that are on either side of the API. REST does not define the specifics of the API but imposes constraints on the nature of the interaction between application and service. The six REST constraints are as follows:

- Client-server
- Stateless
- Cache
- Uniform interface
- Layered system
- Code on demand

The sections that follow cover these constraints in more detail.

Client-Server Constraint

This simple constraint dictates that interaction between application and server is in the client-server request/response style. The principle defined for this constraint is the separation of user interface concerns from data storage concerns. This separation allows client and server components to evolve independently and supports the portability of server-side functions to multiple platforms.

Stateless Constraint

The stateless constraint dictates that each request from a client to a server must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server. Similarly, each response from the server must contain all the desired information for that request. One consequence is that any “memory” of a transaction is maintained in a session state kept entirely on the client. Because the server does not retain any record of the client state, the result is a more efficient SDN controller. Another consequence is that if the client and server reside on different machines, and therefore communicate via a protocol, that protocol need not be connection oriented.

REST typically runs over Hypertext Transfer Protocol (HTTP), which is a stateless protocol.

Cache Constraint

The cache constraint requires that the data within a response to a request be implicitly or explicitly labeled as cacheable or noncacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests. That is, the client is given permission to remember this data because the data is not likely to change on the server side. Therefore, subsequent requests for the same data can be handled locally at the client, reducing communication overhead between client and server, and reducing the server's processing burden.

Uniform Interface Constraint

REST emphasizes a uniform interface between components, regardless of the specific client-server application API implemented using REST. This enables controller services to evolve independently and provides the ability for an SDN controller provider to use software components from various vendors to implement the controller.

To obtain a uniform interface, REST defines four interface constraints:

- **Identification of resources:** Individual resources are identified using a resource identifier (for example, a URI).
- **Manipulation of resources through representations:** Resources are represented in a format like JSON, XML, or HTML.
- **Self-descriptive messages:** Each message has enough information to describe how the message is to be processed.
- **Hypermedia as the engine of the application state:** A client needs no prior knowledge of how to interact with a server, because the API is not fixed but dynamically provided by the server.

The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations (verbs). Flexibility is provided by assigning resources (nouns) their own unique **Uniform Resource Identifier (URI)**. Because each verb has a specific meaning (GET, POST, PUT, and DELETE), REST avoids ambiguity.

The benefit of this constraint, for an SDN environment is that different applications, perhaps written in different languages, can invoke the same controller service via a REST API.

Layered System Constraint

The layered system constraint simply means that a given function is organized in layers, with each layer only having direct interaction with the layers immediately above and below. This is a fairly standard architecture approach for protocol architectures, OS design, and system services design.

Code-on-Demand Constraint

REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be

pre-implemented. Allowing features to be downloaded after deployment improves system extensibility.

Example REST API

To get a feel for the structure of a REST API, it is useful to look at an example. In this section, we discuss a REST API for the northbound interface of the Ryu SDN network operating system. The particular API switch manager service function in Ryu is designed to provide access to OpenFlow switches.

Each function that can be performed by the switch manager on behalf of an application is assigned a URI. For example, consider the function to get a description of all the entries in the group table of a particular switch. The URI for this function for this switch is as follows:

/stats/group/<*dpid*>

where stats (statistic) refers to the set of APIs for retrieving and updating switch statistics and parameters, group is the name of the function, and <*dpid*> (data path ID) is the unique identifier of the switch. To invoke the function for switch 1, the application issues the following command to the switch manager across the REST API:

GET http://localhost:8080/stats/groupdesc/1

The **localhost** portion of this command indicates that the application is running on the same server as the Ryu NOS. If the application were remote, the URI would be a URL that provides remote access via HTTP and the web. The switch manager responds to this command with a message whose message body includes the dpid then a sequence of blocks of values, one for each group defined in the switch dpid. The values are as follows:

- **type:** All, select, fast failover, or indirect (see Section 4.2).
- **group_id:** Identifier of an entry in the group table.
- **buckets:** A structured field consisting of the following subfields:
- **weight:** Relative weight of bucket (only for select type).
- **watch_port:** Port whose state affects whether this bucket is live (only required for fast failover groups).
- **watch_group:** Group whose state affects whether this bucket is live (only required for fast failover groups).
- **actions:** A list, possibly null, of actions.

The buckets portion of the message body is repeated, once for each group table entry.

Table 5.2 lists all the API functions for retrieving switch statistics and parameters that use the GET message type. There are also several functions that use the POST message type, in which the request message body includes a list of parameters that must be matched.

Request Type	Response Message Body Attributes
Get all switches	Data path ID
Get switch description	Data path ID, manufacturer description, hardware description, software description, serial number, human readable description of data path.
Get all flow stats of switch	Data path ID, length of this entry, table ID, time flow alive in seconds, time flow alive in nanoseconds, priority, number of seconds idle before expiration, number of seconds before expiration, flags, cookie, packet count, byte count, fields to match, actions
Get aggregate flow stats of switch	Data path ID, packet count, byte count, number of flows
Get port stats	Receive packet count, transmit packet count, receive byte count, transmit byte count, dropped receive packet count, dropped transmit packet count, receive error count, transmit error count, frame alignment error count, receive packet overrun count, CRC error count, collision count, time port alive in seconds, time port alive in nanoseconds
Get ports description	Data path ID, port number, Ethernet address, port name. config flags, state flag, current features, advertised features, supported features, features advertised by peer, current bit rate, max bitrate
Get queues stats	Data path ID, port number, queue ID, transmit byte count, transmit packet count, packet overrun count, time queue alive in seconds, time queue alive in nanoseconds
Get groups stats	Data path ID, length of this entry, group ID, number of flows or groups that forward to this group, packet count, byte count
Get group description	Data path ID, type, group ID, buckets (weight, watch_port, watch_group, actions)
Get group features	Data path ID, types, capabilities, max number of groups, actions supported
Get meters stats	Data path ID, meter ID, length of this entry, number of flows, input packet count, input byte count, time meter alive in seconds, time meter alive in nanoseconds, meter band (packet count, byte count)
Get meter configuration	Data path ID, flags, meter ID, bands (type, rate, burst size)
Get meter features	Data path ID, max number of meters, band types, capabilities, max bands per meter, max color value

TABLE 5.2 Ryu REST APIs for Retrieving Switch Statistics Using GET

The switch manager API also provides functions for updating switch parameters. These all use the POST message type. In this case, the request message body includes the parameters and their values to be updated. Table 5.3 lists the update API functions.

Request Type	Request Message Body Attributes
Add flow entry	Data path ID, cookie, cookie mask, table ID, idle timeout, hard timeout, priority, buffer id, flags, fields to match, actions
Modify matching flow entries	Data path ID, cookie, cookie mask, table ID, idle timeout, hard timeout, priority, buffer id, flags, fields to match, actions
Delete matching flow entries	Data path ID, cookie, cookie mask, table ID, idle timeout, hard timeout, priority, buffer id, flags, fields to match, actions
Delete all flow entries	Data path ID
Add group entry	Data path ID, type, group ID, buckets (weight, watch_port, watch_group, actions)
Modify group entry	Data path ID, type, group ID, buckets (weight, watch_port, watch_group, actions)
Delete group entry	Data path ID, group id
Add meter entry	Data path ID, flags, meter ID, bands (type, rate, burst size)
Modify meter entry	Data path ID, flags, meter ID, bands (type, rate, burst size)
Delete meter entry	Data path ID, meter ID

TABLE 5.3 Ryu REST APIs for Update Switch Statistics Filtered by Fields Using POST

5.5 Cooperation and Coordination Among Controllers

In addition to northbound and southbound interfaces, a typical SDN controller will have an east/westbound interface that enables communication with other SDN controllers and other networks. As yet, there has been no significant progress on open source or standardized east/west protocols or interfaces. This section surveys key design issues related to the east/westbound interface.

Centralized Versus Distributed Controllers

A key architectural design decision is whether a single centralized controller or a distributed set of controllers will be used to control the data plane switches. A centralized controller is a single server that manages all the data plane switches in the network.

In a large enterprise network, the deployment of a single controller to manage all network devices would prove unwieldy or undesirable. A more likely scenario is that the operator of a large enterprise or carrier network divides the whole network into a number of nonoverlapping SDN domains, also called SDN islands (Figure 5.10), managed by distributed controllers. Reasons for using SDN domains include those in the list that follows.

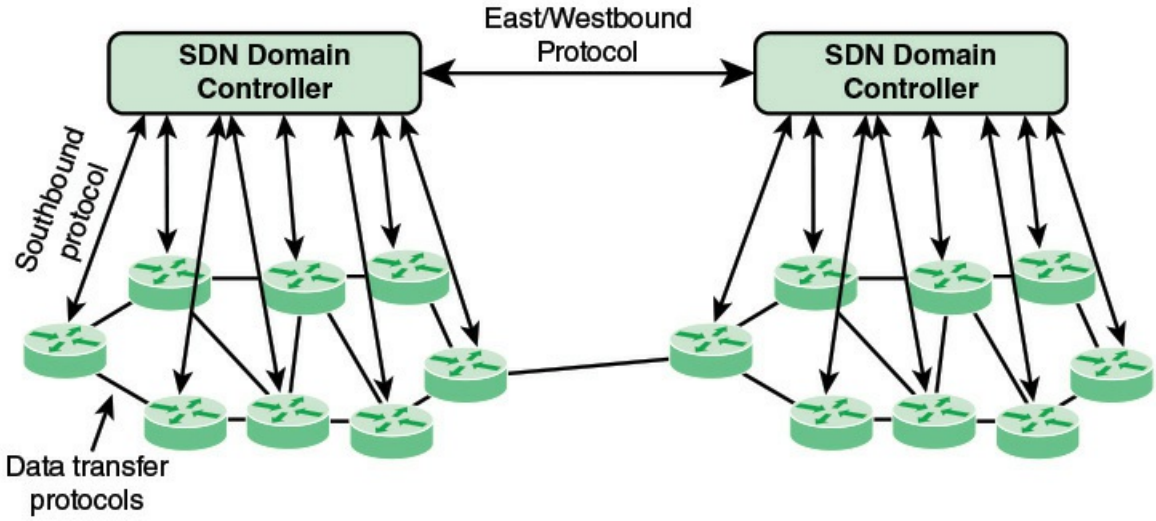


FIGURE 5.10 SDN Domain Structure

- **Scalability**: The number of devices an SDN controller can feasibly manage is limited. Therefore, a reasonably large network may need to deploy multiple SDN controllers.
- **Reliability**: The use of multiple controllers avoids the risk of a single point of failure.
- **Privacy**: A carrier may choose to implement different privacy policies in different SDN domains. For example, an SDN domain may be dedicated to a set of customers who implement their own highly customized privacy policies, requiring that some networking information in this domain (for example, network topology) should not be disclosed to an external entity.
- **Incremental deployment**: A carrier's network may consist of portions of legacy and nonlegacy infrastructure. Dividing the network into multiple individually manageable SDN domains allows for flexible incremental deployment.

Distributed controllers may be collocated in a small area, or widely dispersed, or a combination of the two. Closely placed controllers offer high throughput and are appropriate for data centers, whereas dispersed controllers accommodate multilocation networks.

Typically, controllers are distributed horizontally. That is, each controller governs a nonoverlapping subset of the data plane switches. A vertical architecture is also possible, in which control tasks are distributed to different controllers depending on criteria such as network view and locality requirements.

In a distributed architecture, a protocol is needed for communication among the controllers. In principle, a proprietary protocol could be used for this purpose, although an open or standard protocol would clearly be preferable for purposes of interoperability.

The functions associated with the east/westbound interface for a distributed architecture include maintaining either a partitioned or replicated database of network topology and parameters, and monitoring/notification functions. The latter function includes checking whether a controller is alive and coordinating changes in assignment of switches to controllers.

High-Availability Clusters

Within a single domain, the controller function can be implemented on a **high-availability (HA) cluster**. Typically, there would be two or more nodes that share a single IP address that is used by external systems (both north and southbound) to access the cluster. An example is the IBM SDN for Virtual Environments product, which uses two nodes. Each node is considered a peer of the other node in the cluster for data replication and sharing of the external IP address. When HA is running, the primary node is responsible for answering all traffic that is sent to the cluster's external IP address and holds a read/write copy of the configuration data. Meanwhile, the second node operates as a standby, with a read-only copy of the configuration data, which is kept current with the primary's copy. The secondary node monitors the state of the external IP. If the secondary node determines that the primary node is no longer answering the external IP, it triggers a failover, changing its mode to that of primary node. It assumes the responsibility for answering the external IP and changes its copy of configuration data to be read/write. If the old primary reestablishes connectivity, there is an automatic recovery process trigger to convert the old primary to secondary status so that configuration changes that are made during the failover period are not lost.

ODL Helium has HA built in, and Cisco XNC and the Open Network controller have HA features (up to five in a cluster).

Federated SDN Networks

The distributed SDN architecture discussed in the preceding paragraphs refers to a system of SDN domains that are all part of a single enterprise network. The domains may be collocated or on separate sites. In either case, the management of all the data plane switches is under the control of a single network management function.

It is also possible for SDN networks that are owned and managed by different organizations to cooperate using east/westbound protocols. Figure 5.11 is an example of the potential for inter-SDN controller cooperation.

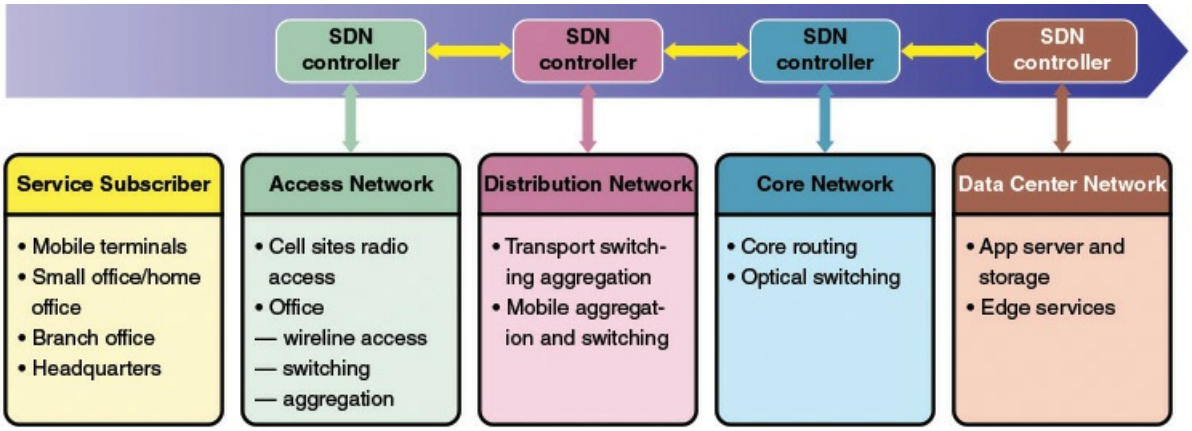


FIGURE 5.11 Federation of SDN Controllers [GUPT14]

In this configuration, we have a number of service subscribers to a data center network providing cloud-based services. Typically, as was illustrated previously in Figure 1.3, subscribers are connected to the service network through a hierarchy of access, distribution, and core networks.

These intermediate networks may all be operated by the data center network, or they may involve other organizations. In the latter case, if all the networks implement SDN, they need to share common conventions for share control plane parameters, such as quality of service (QoS), policy information, and routing information.

Border Gateway Protocol

Before proceeding further with our discussion, it will be useful to provide an overview of the Border Gateway Protocol (BGP). BGP was developed for use in conjunction with internets that use the TCP/IP suite, although the concepts are applicable to any internet. BGP has become the preferred **exterior router protocol (ERP)** for the Internet.

BGP enables routers, called gateways in the standard, in different autonomous systems to cooperate in the exchange of routing information. The protocol operates in terms of messages, which are sent over TCP connections. The current version of BGP is known as BGP-4.

Three functional procedures are involved in BGP:

- Neighbor acquisition
- Neighbor reachability
- Network reachability

Two routers are considered to be neighbors if they are attached to the same network or communication link. If they are attached to the same network, communication between the neighbor routers might require a path through other routers within the shared network. If the two routers are in different autonomous systems, they may want to exchange routing information. For this purpose, it is necessary first to perform **neighbor acquisition**. The term *neighbor* refers to two routers that share the same network. In essence, neighbor acquisition occurs when two neighboring routers in different autonomous systems agree to exchange routing information regularly. A formal acquisition procedure is needed because one of the routers may not want to participate. For example, the router may be overburdened and may not want to be responsible for traffic coming in from outside the AS. In the neighbor acquisition process, one router sends a request message to the other, which may either accept or refuse the offer. The protocol does not address the issue of how one router knows the address or even the existence of another router, nor how it decides that it needs to exchange routing information with that particular router. These issues must be dealt with at configuration time or by active intervention of a network manager.

To perform neighbor acquisition, one router sends an Open message to another. If the target router accepts the request, it returns a Keepalive message in response.

Once a neighbor relationship is established, the **neighbor reachability** procedure is used to maintain the relationship. Each partner needs to be assured that the other partner still exists and is still engaged in the neighbor relationship. For this purpose, the two routers periodically issue Keepalive messages to each other.

The final procedure specified by BGP is **network reachability**. Each router maintains a database of the networks that it can reach and the preferred route for reaching each network. Whenever a change is made to this database, the router issues an Update message that is broadcast to all other routers for which it has a neighbor relationship. Because the Update message is broadcast, all

BGP routers can build up and maintain their routing information.

Routing and QoS Between Domains

For routing outside a controller's domain, the controller establishes a BGP connection with each neighboring router. Figure 5.12 illustrates a configuration with two SDN domains that are linked only through a non-SDN AS.

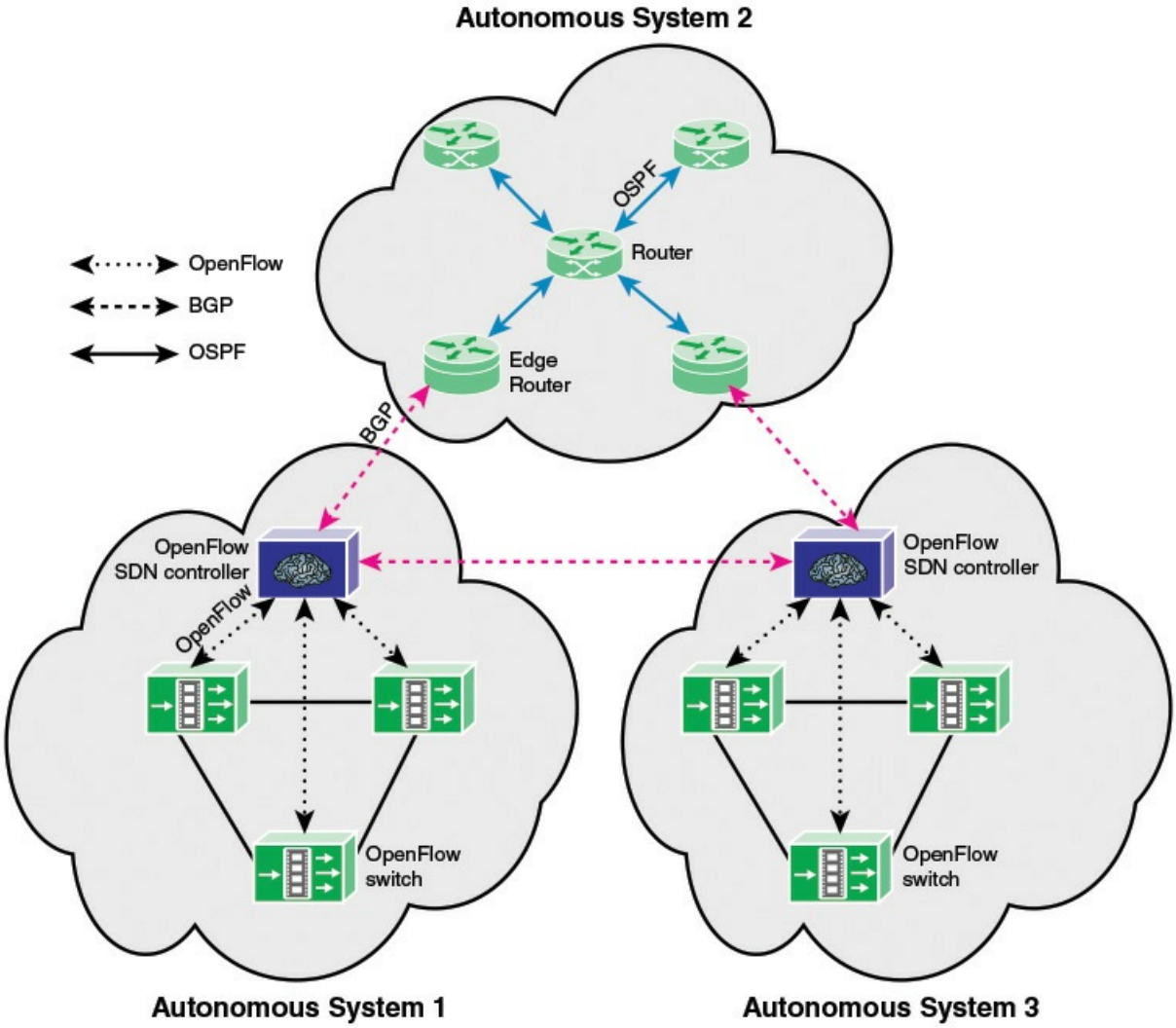


FIGURE 5.12 Heterogeneous Autonomous Systems with OpenFlow and Non-OpenFlow Domains

Within the non-SDN AS, OSPF is used for interior routing. OSPF is not needed in an SDN domain; rather, the necessary routing information is reported from each data plane switch to the centralized controller using a southbound protocol (in this case, OpenFlow). Between each SDN domain and the AS, BGP is used to exchange information, such as the following:

- **Reachability update:** Exchange of reachability information facilitates inter-SDN domain routing. This allows a single flow to traverse multiple SDNs and each controller can select

the most appropriate path in the network.

- **Flow setup, tear-down, and update requests:** Controllers coordinate flow setup requests, which contain information such as path requirements, QoS, and so on, across multiple SDN domains.
- **Capability Update:** Controllers exchange information on network-related capabilities such as bandwidth, QoS and so on, in addition to system and software capabilities available inside the domain.

Several additional points are worth observing with respect to Figure 5.12:

- The figure depicts each AS as a cloud containing interconnected routers and, in the case of an SDN domain, a controller. The cloud represents an internet, so that the connection between any two routers is a network within the internet. Similarly, the connection between two adjacent autonomous systems is a network, which may be part of one of the two adjacent autonomous systems, or a separate network.
- For an SDN domain, the BGP function is implemented in the SDN controller rather than a data plane router. This is because the controller is responsible for managing the topology and making routing decisions.
- The figure shows a BGP connection between autonomous systems 1 and 3. It may be that these networks are not directly connected by a single network. However, if the two SDN domains are part of a single SDN system, or if they are federated, it may be desirable to exchange additional SDN-related information.

Using BGP for QoS Management

A common practice for inter-AS interconnection is a best-effort interconnection only. That is, traffic forwarding between autonomous systems is without traffic class differentiation and without any forwarding guarantee. It is common for network providers to reset any IP packet traffic class markings to zero, the best-effort marking, at the AS ingress router, which eliminates any traffic differentiation. Some providers perform higher-layer classification at the ingress to guess the forwarding requirements and to match on their AS internal QoS forwarding policy. There is no standardized set of classes, no standardized marking (class encoding), and no standardized forwarding behavior, that cross-domain traffic could rely on. However RFC 4594 (*Configuration Guidelines for DiffServ Service Classes*, August 2006) provides a set of "best practices" related to this parameters. QoS policy decisions are taken by network providers independently and in an uncoordinated fashion. This general statement does not cover existing individual agreements, which do offer quality-based interconnection with strict QoS guarantees. However, such service level agreement (SLA)-based agreements are of bilateral or multilateral nature and do not offer a means for a general "better than best effort" interconnection.

IETF is currently at work on a standardized scheme for QoS marking using BGP (*BGP Extended Community for QoS Marking*, draft-knoll-idr-qos-attribute-12, July 10, 2015). Meanwhile, SDN providers have implemented their own capabilities using the extensible nature of BGP. In either case, the interaction between SDN controllers in different domains using BGP would involve the steps illustrated in Figure 5.13 and described in the list that follows.

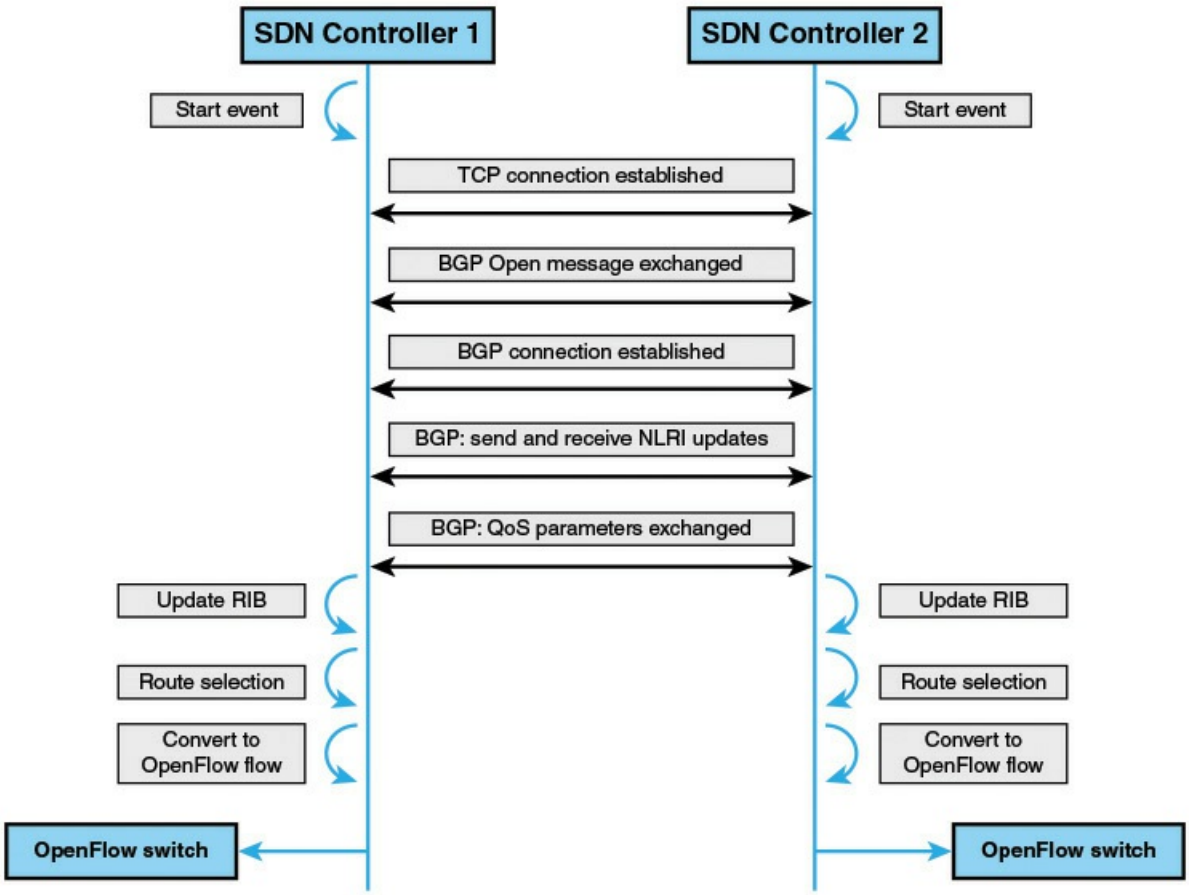


FIGURE 5.13 East-West Connection Establishment, Route, and Flow Setup

1. The SDN controller must be configured with BGP capability and with information about the location of neighboring BGP entices.

2. BGP is triggered by a start or activation event within the controller.

3. The BGP entity in the controller attempts to establish a TCP connection with each neighboring BGP entity.

4. Once a TCP connection is established, the controller's BGP entity exchanges Open messages with the neighbor. Capability information is exchanged with using the Open messages.

5. The exchange completes with the establishment of a BGP connection.

6. Update messages are used to exchange NLRI (network layer reachability information), indicating what networks are reachable via this entity. Reachability information is used in the selection of the most appropriate data path between SDN controllers. Information obtained through NLRI parameter is used to update the controller's Routing Information Base (RIB). This in turn enables the controller to set the appropriate flow information in the data plane switches.

7. The Update message can also be used to exchange QoS information, such as available capacity.

8. Route selection is done when more than one path is available based on BGP process decision. Once the path is established packets can traverse successfully between two SDN domains.

IETF SDNi

IETF has developed a draft specification that defines common requirements to coordinate flow setup and exchange reachability information across multiple domains, referred to as SDNi (*SDNi: A Message Exchange Protocol for Software Defined Networks across Multiple Domains*, draft-yin-sdn-sdni-00.txt, June 27, 2012). The SDNi specification does not define an east/westbound SDN protocol but rather provides some of the basic principles to be used in developing such a protocol.

SDNi functionality, as defined in the document, includes the following:

- Coordinate flow setup originated by applications, containing information such as path requirement, QoS, and service level agreements across multiple SDN domains.
- Exchange reachability information to facilitate inter-SDN routing. This will allow a single flow to traverse multiple SDNs and have each controller select the most appropriate path when multiple such paths are available.

SDNi depends on the types of available resources and capabilities available and managed by the different controllers in each domain. Therefore, it is important to implement SDNi in a descriptive and open manner so that new capabilities offered by different types of controllers will be supported. Because SDN in essence allows for innovation, it is important that data exchanged between controllers will be dynamic in nature; that is, there should be some metadata exchange that will allow SDNi to exchange information about unknown capabilities.

The message types for SDNi tentatively include the following:

- Reachability update
- Flow setup/teardown/update request (including application capability requirement such as QoS, data rate, latency, and so on)
- Capability update (including network-related capabilities, such as data rate and QoS, and system and software capabilities available inside the domain)

OpenDaylight SNDi

Included in the OpenDaylight architecture is an SDNi capability for connecting multiple OpenDaylight federated controllers in a network and sharing topology information among them. This capability appears to be compatible with the IETF specification for an SDNi function. The SDNi application deployable on an OpenDaylight controller consists of three components, as illustrated in Figure 5.14 and described in the list that follows.

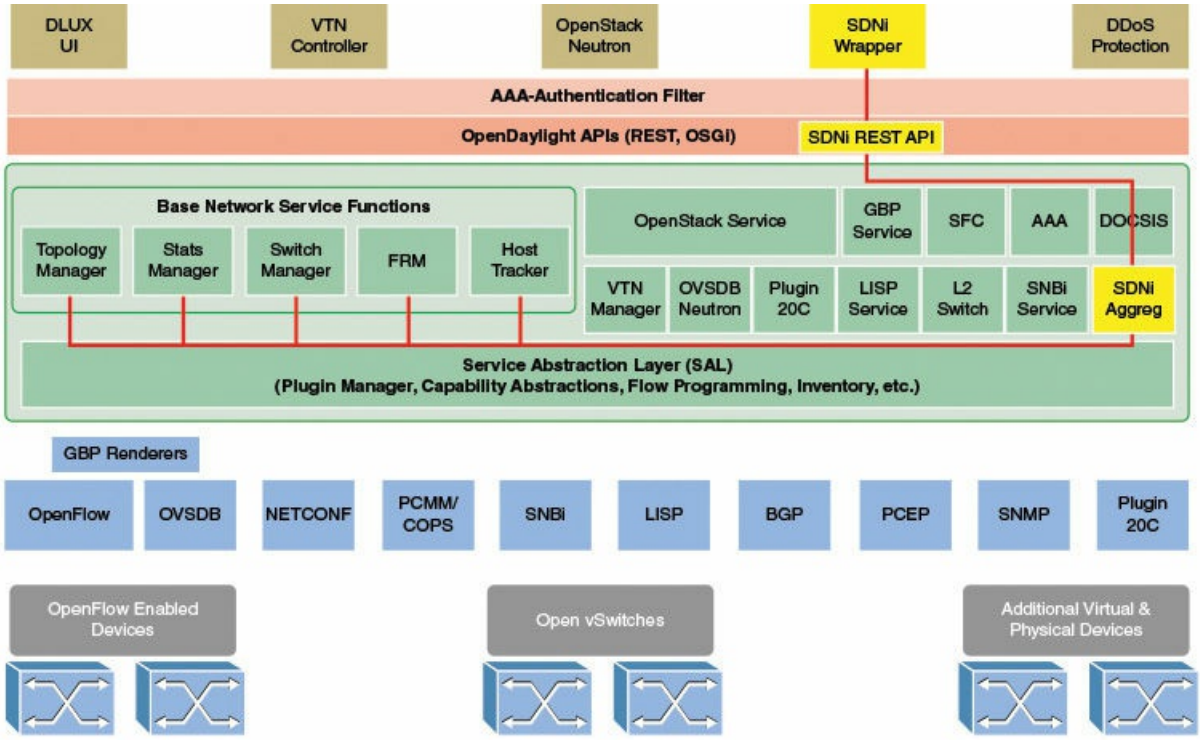


FIGURE 5.14 SDNi Components in OpenDaylight Structure (Helium)

- **SDNi aggregator:** Northbound SDNi plug-in acts as an aggregator for collecting network information such as topology, statistics, and host identifiers. This plug-in can evolve to meet the needs for network data requested to be shared across federated SDN controllers.
- **SDNi REST API:** SDNi REST APIs fetch the aggregated information from the northbound plug-in (SDNi aggregator).
- **SDNi wrapper:** SDNi BGP wrapper is responsible for the sharing and collecting information to/from federated controllers.

Figure 5.15 shows the interrelationship of the components, with a more detailed look at the SDNi wrapper. The SDNi aggregator collects statistics and parameters from the base network service functions, on behalf of requests via the REST API. The heart of the wrapper is an OpenDaylight implementation of the Border Gateway Protocol (BGP). BGP is an ERP suitable for exchanging routing information between routers that connect SDN domains.

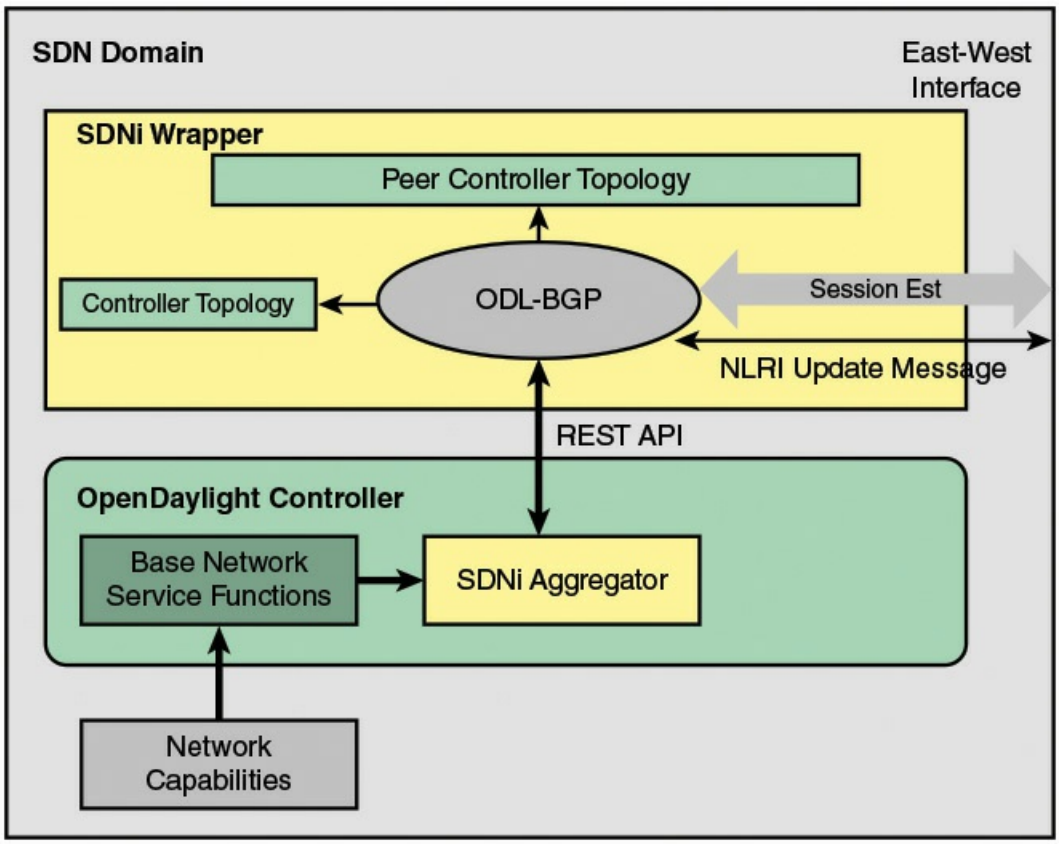


FIGURE 5.15 OpenDaylight SDNi Wrapper

5.6 Key Terms

After completing this chapter, you should be able to define the following terms.

Border Gateway Protocol (BGP)

centralized controller

distributed controller

east/westbound interface

exterior router protocol (ERP)

interior router protocol (IRP)

neighbor acquisition

neighbor reachability

network operating system (NOS)

network reachability

northbound interface

Open Service Gateway Initiative (OSGi)

OpenDaylight

OpenFlow

REpresentational State Transfer (REST)

RESTful

routing

Ryu

SDN control plane

SDNi

service abstraction layer (SAL)

southbound interface

Uniform Resource Identifier (URI)

5.7 References

GUPT14: Gupta, D., and Jahan, R. *Inter-SDN Controller Communication: Using Border Gateway Protocol.* Tata Consultancy Services White Paper, 2014. http://www.tcs.com.

KREU15: Kreutz, D., et al. "Software-Defined Networking: A Comprehensive Survey." *Proceedings of the IEEE*, January 2015.

Chapter 6. SDN Application Plane

> Life in the modern world is coming to depend more and more upon technical means of communication. Without such technical aids the modern city-state could not exist, for it is only by means of them that trade and business can proceed; that goods and services can be distributed where needed; that railways can run on schedule; that law and order are maintained; that education is possible. Communication renders true social life practicable, for communication means organization.
>
> —*On Human Communication*, Colin Cherry

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

- Present an overview of the SDN application plane architecture.
- Define the network services abstraction layers.
- List and explain three forms of abstraction in SDN.
- List and describe six major application areas of interest for SDN.

The power of the software-defined networking (SDN) approach to networking is in the support it provides for network applications to monitor and manage network behavior. The SDN control plane provides the functions and services that facilitate rapid development and deployment of network applications.

While the SDN data and control planes are well defined, there is much less agreement on the nature and scope of the application plane. At minimum, the application plane includes a number of network applications—that is, applications that specifically deal with network management and control. There is no agreed-upon set of such applications or even categories of such applications. Further, the application layer may include general-purpose network abstraction tools and services that might also be viewed as part of the functionality of the control plane.

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.

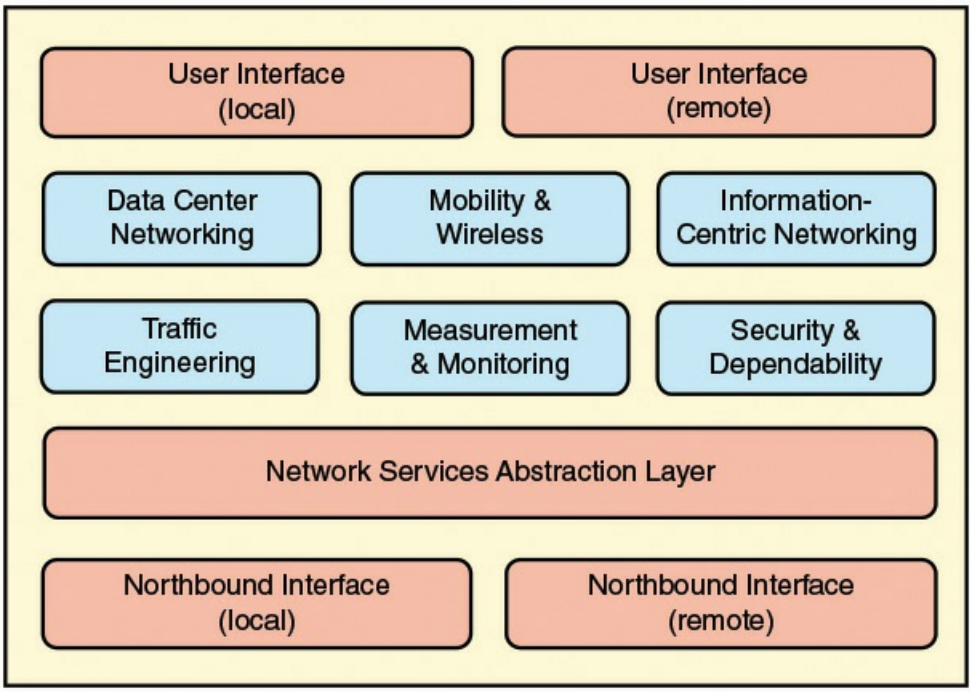


FIGURE 6.1 SDN Application Plane Functions and Interfaces

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.

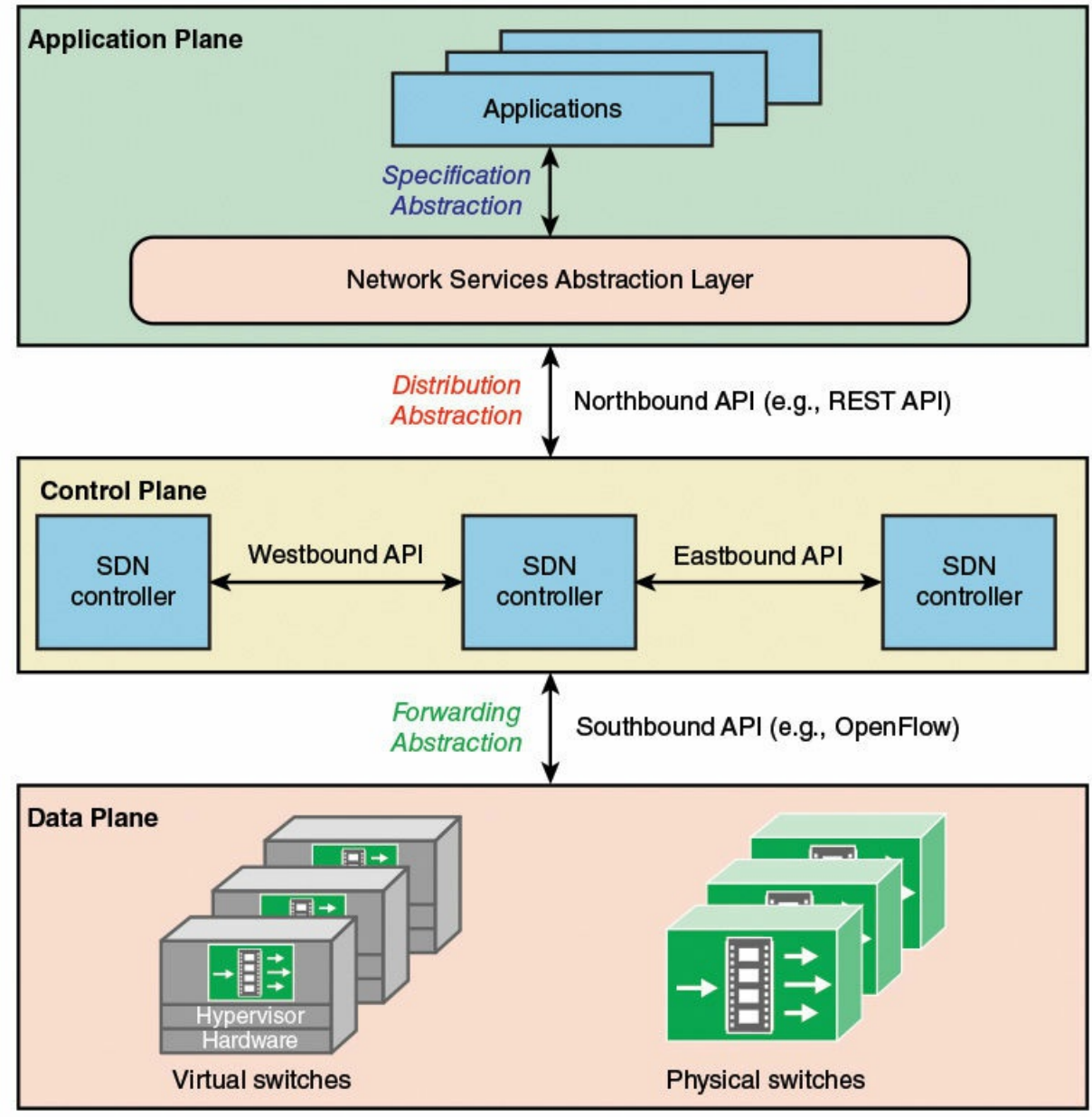


FIGURE 6.2 SDN Architecture and Abstractions

Forwarding Abstraction

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.

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

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.

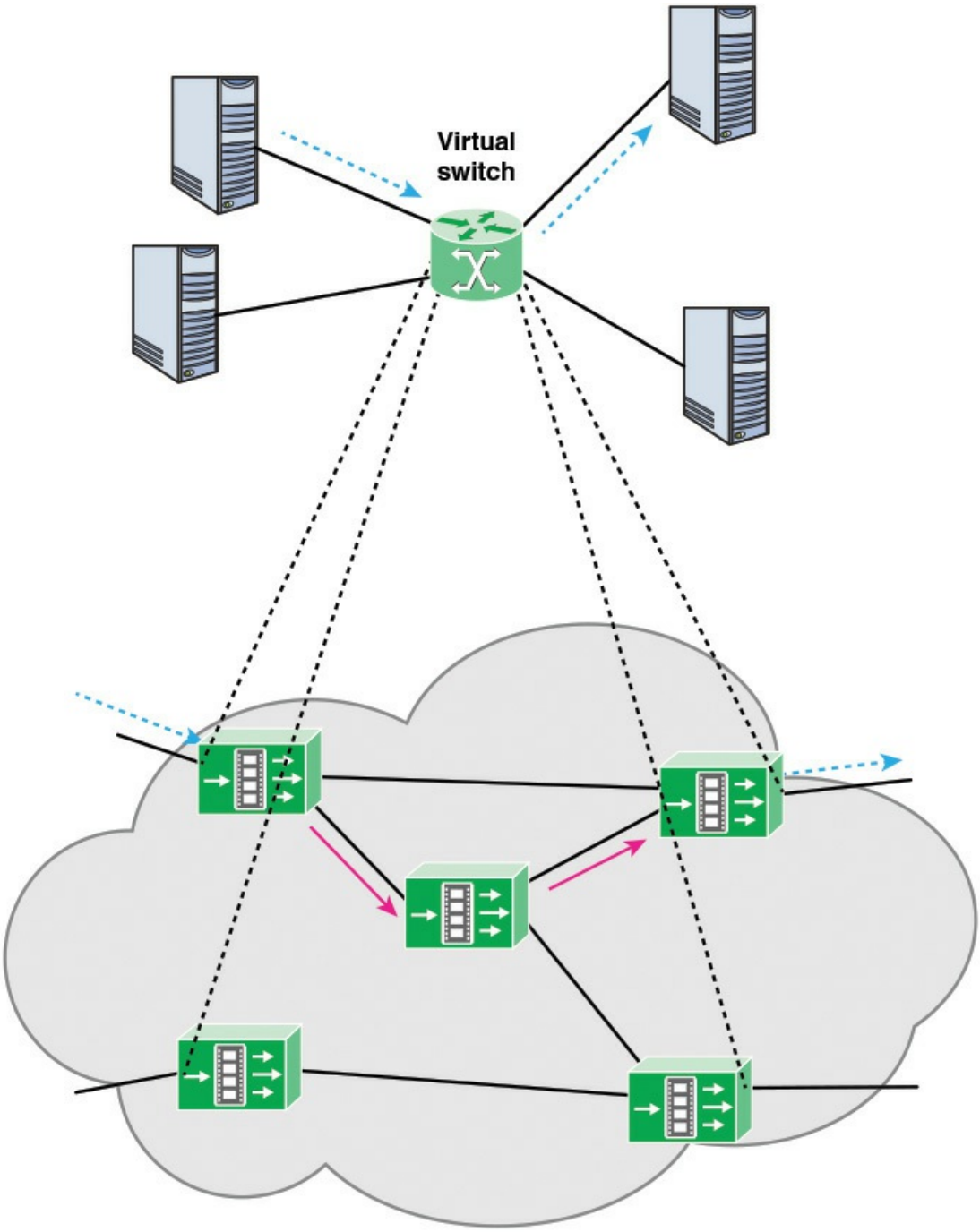


FIGURE 6.3 Virtualization of a Switching Fabric for MAC Learning

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 selecting, filtering, splitting, merging and aggregating the streams of packets. Another special feature of this language is that it enables the queries to be composed with forwarding policies. A compiler produces the control messages needed to query and tabulate the counters on switches.

Frenetic consists of two levels of abstraction, as illustrated in Figure 6.4. The upper level, which is the Frenetic source-level API, provides a set of operators for manipulating streams of network traffic. The query language provides means for reading the state of the network, merging different queries, and expressing high-level predicates for classifying, filtering, transforming, and aggregating the packet streams traversing the network. The lower level of abstraction is provided by a run-time system that operates in the SDN controller. It translates high-level policies and queries into low-level flow rules and then issues the needed OpenFlow commands to install these rules on the switches.

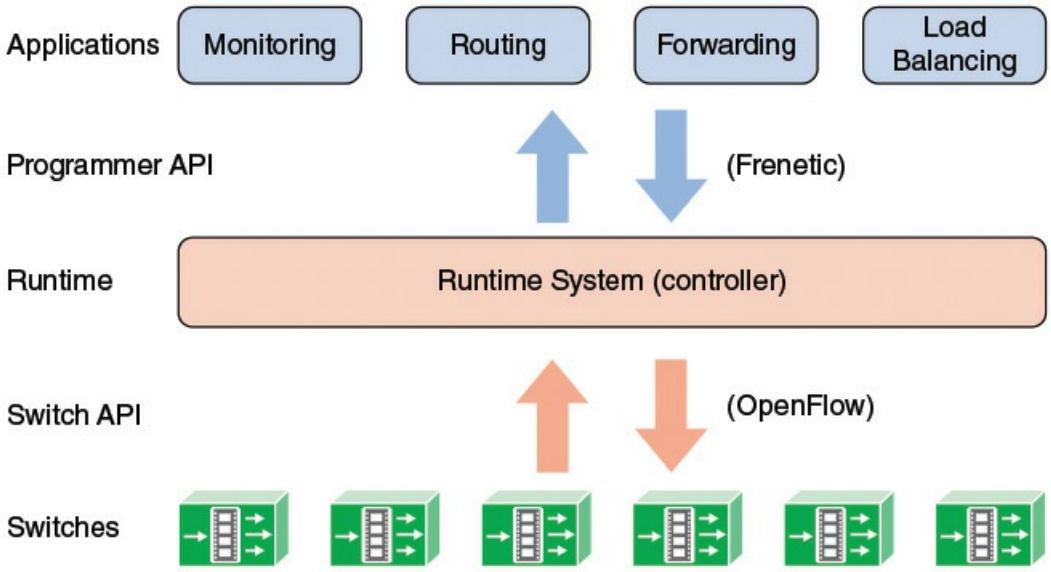


FIGURE 6.4 Frenetic Architecture

To get some idea of the two levels of abstraction, consider a simple example, from a paper by Foster in the February 2013 *IEEE Communications Magazine* [FOST13]. The program combines forwarding functionality with monitoring web traffic functionality. Consider the following Python program, which executes at the run-time level, to control OpenFlow switches:

Click here to view code image

```
def switch_join(s):
 pat1 = {inport:1}
 pat2web = {inport:2, srcport:80}
 pat2 = {inport:2}
 install(s, pat1, DEFAULT, [fwd(2)])
 install(s, pat2web, HIGH, [fwd(1)])
```

```
    install(s, pat2, DEFAULT, [fwd(1)])
    query_stats(s, pat2web)
def stats_in(s, xid, pat, pkts, bytes):
    print bytes
    sleep(30)
    query_stats(s, pat)
```

When a switch joins the network, the program installs three forwarding rules in the switch for three types of traffic: traffic arriving on port 1, web traffic arriving on port 2, and other traffic arriving on port 2. The second rule has HIGH priority and so takes precedence over the third rule, which has default priority. The call to `query_stats` generates a request for the counters associated with the `pat2web` rule. When the controller receives the reply, it invokes the `stats_in` handler. This function prints the statistics polled on the previous iteration of the loop, waits 30 seconds, and then issues a request to the switch for statistics matching the same rule.

The way the program is written, the logic for forwarding and web monitoring are intertwined. This reflects the nature of the underlying OpenFlow functionality. Any changes or additions to either function will affect the program in a complex way.

With Frenetic, these two functions can be expressed separately, as follows:

Click here to view code image

```
    def repeater():
     rules=[Rule(inport:1, [fwd(2)])
            Rule(inport:2, [fwd(1)])]
     register(rules)
def web monitor():
    q = (Select(bytes) *
       Where(inport=2 & srcport=80) *
       Every(30))
    q >> Print()
def main():
    repeater()
    monitor()
```

With this code, it would be easy to change the monitor program or swap it out for another monitor program without touching the repeater code, and similarly for the changes to the repeater program. Importantly, the responsibility for installing specific OpenFlow rules that realize both components simultaneously is delegated to the run-time system. For this example, the run-time system would generate the same rules as the manually constructed rules in the switch join function listed above.

6.3 Traffic Engineering

Traffic engineering is a method for dynamically analyzing, regulating, and predicting the behavior of data flowing in networks with the aim of performance optimization to meet service level agreements (SLAs). Traffic engineering involves establishing routing and forwarding policies based on QoS requirements. With SDN, the task of traffic engineering should be considerably simplified compared with a non-SDN network. SDN offers a uniform global view of heterogeneous equipment and powerful tools for configuring and managing network switches.

This is an area of great activity in the development of SDN applications. The SDN survey paper by Kreutz in the January 2015 *Proceedings of the IEEE* [KREU15] lists the following traffic engineering functions that have been implemented as SDN applications:

- On-demand virtual private networks
- Load balancing
- Energy-aware routing
- Quality of service (QoS) for broadband access networks
- Scheduling/optimization
- Traffic engineering with minimal overhead
- Dynamic QoS routing for multimedia apps
- Fast recovery through fast-failover groups
- QoS policy management framework
- QoS enforcement
- QoS over heterogeneous networks
- Multiple packet schedulers
- Queue management for QoS enforcement
- Divide and spread forwarding tables

PolicyCop

An instructive example of a traffic engineering SDN application is PolicyCop [BARI13], which is an automated QoS policy enforcement framework. It leverages the programmability offered by SDN and OpenFlow for

- Dynamic traffic steering
- Flexible Flow level control
- Dynamic traffic classes
- Custom flow aggregation levels

Key features of PolicyCop are that it monitors the network to detect policy violations (based on a QoS SLA) and reconfigures the network to reinforce the violated policy.

As shown in Figure 6.5, PolicyCop consists of eleven software modules and two databases, installed in both the application plane and the control plane. PolicyCop uses the control plane of SDNs to monitor the compliance with QoS policies and can automatically adjust the control plane rules and flow tables in the data plane based on the dynamic network traffic statistics.

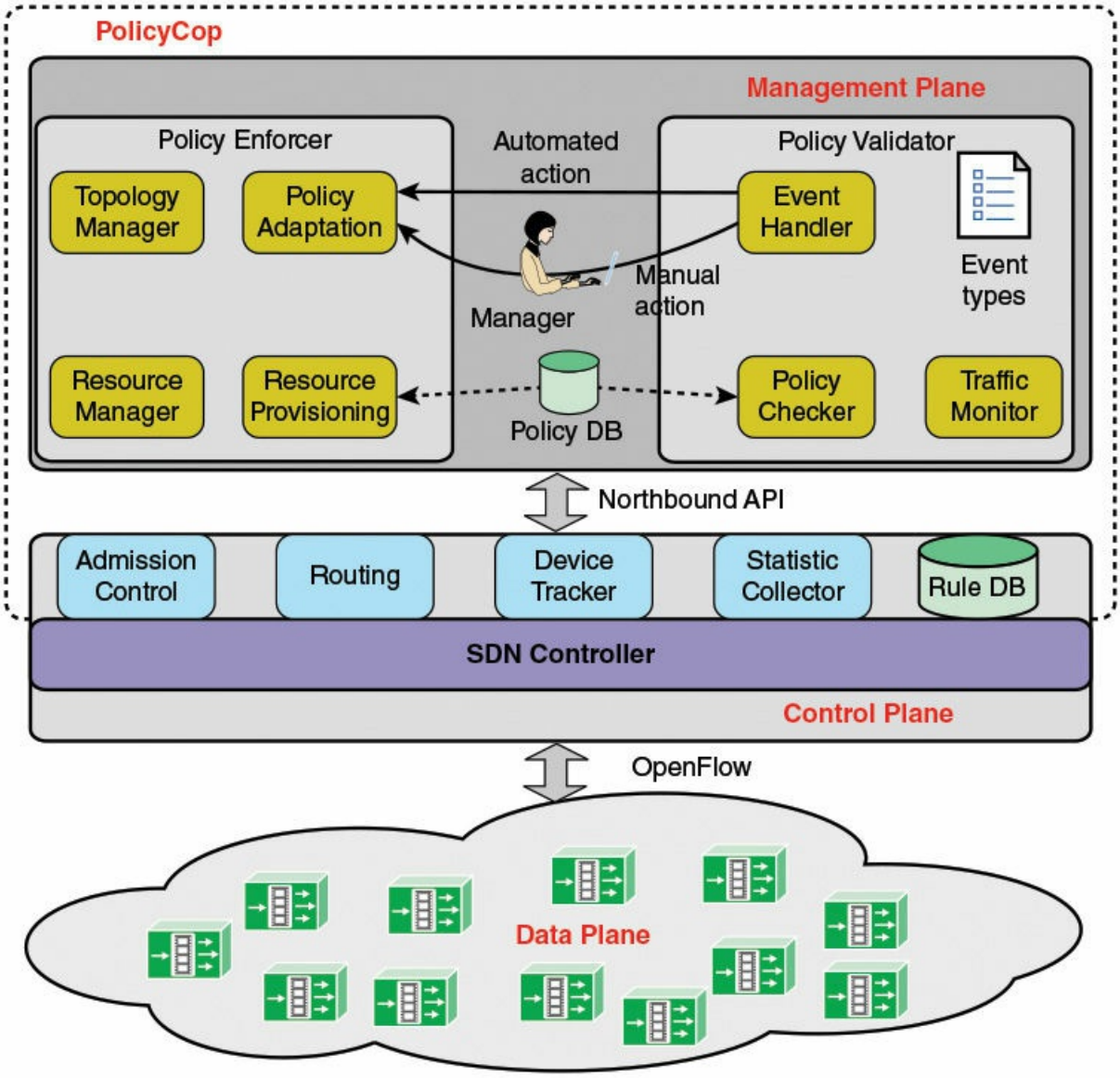


FIGURE 6.5 PolicyCop Architecture

In the control plane, PolicyCop relies on four modules and a database for storing control rules, described as follows:

- **Admission Control:** Accepts or rejects requests from the resource provisioning module for reserving network resources, such as queues, flow-table entries, and capacity.
- **Routing:** Determines path availability based on the control rules in the rule database.
- **Device Tracker:** Tracks the up/down status of network switches and their ports.
- **Statistics Collection:** Uses a mix of passive and active monitoring techniques to measure different network metrics.
- **Rule Database:** The application plane translates high-level network-wide policies to control rules and stores them in the rule database.

A RESTful northbound interface connects these control plane modules to the application plane modules, which are organized into two components: a policy validator that monitors the network to detect policy violations, and a policy enforcer that adapts control plane rules based on network conditions and high-level policies. Both modules rely on a policy database, which contains QoS policy rules entered by a network manager. The modules are as follows:

- **Traffic Monitor:** Collects the active policies from policy database, and determines appropriate monitoring interval, network segments, and metrics to be monitored.
- **Policy Checker:** Checks for policy violations, using input from the policy database and the Traffic Monitor.
- **Event Handler:** Examines violation events and, depending on event type, either automatically invokes the policy enforcer or sends an action request to the network manager.
- **Topology Manager:** Maintains a global view of the network, based on input from the device tracker.
- **Resource Manager:** Keeps track of currently allocated resources using admission control and statistics collection.
- **Policy Adaptation:** Consists of a set of actions, one for each type of policy violation. Table 6.1 shows the general functionality of some of the policy adaptation actions. The actions are pluggable components that can be specified by the network manager.

SLA Parameter	PAA Functionality
Packet loss	Modify queue configuration or reroute to a better path
Throughput	Modify rate limiters to throttle misbehaving flows
Latency	Schedule flow through a new path with less congestion and suitable delay
Jitter	Reroute flow through a less congested path
Device failure	Reroute flows through a different path to bypass the failure

TABLE 6.1 Functionality of Some Example Policy Adaptation Actions (PAAs)

- **Resource Provisioning:** This module either allocates more resources or releases existing ones or both based on the violation event.

Figure 6.6 shows the process workflow in PolicyCop.

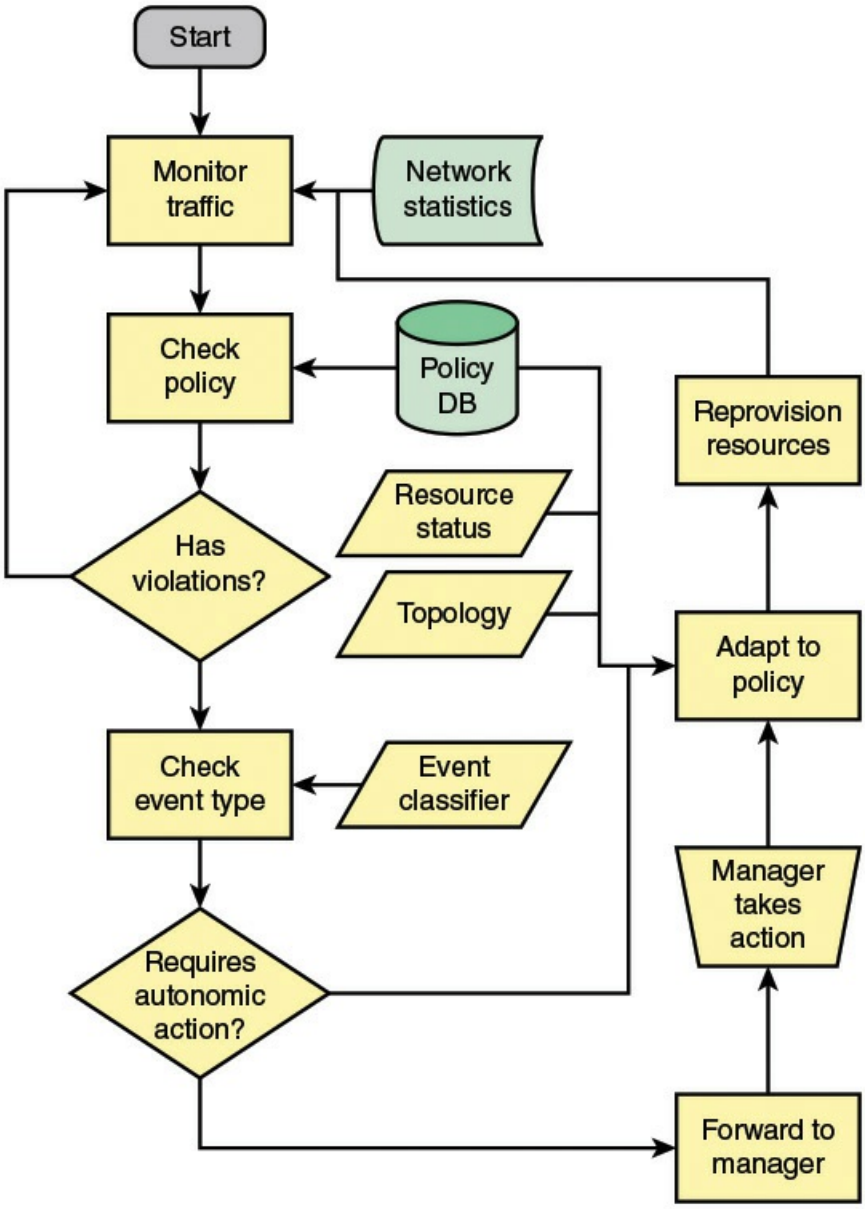


FIGURE 6.6 PolicyCop Workflow

6.4 Measurement and Monitoring

The area of measurement and monitoring applications can roughly be divided into two categories: applications that provide new functionality for other networking services, and applications that add value to OpenFlow-based SDNs.

An example of the first category is in the area of broadband home connections. If the connection is to an SDN-based network, new functions can be added to the measurement of home network traffic and demand, allowing the system to react to changing conditions. The second category typically involves using different kinds of sampling and estimation techniques to reduce the

burden of the control plane in the collection of data plane statistics.

6.5 Security

Applications in this area have one of two goals:

- **Address security concerns related to the use of SDN:** SDN involves a three-layer architecture (application, control, data) and new approaches to distributed control and encapsulating data. All of this introduces the potential for new vectors for attack. Threats can occur at any of the three layers or in the communication between layers. SDN applications are needed to provide for the secure use of SDN itself.
- **Use the functionality of SDN to improve network security:** Although SDN presents new security challenges for network designers and managers, it also provides a platform for implementing consistent, centrally managed security policies and mechanisms for the network. SDN allows the development of SDN security controllers and SDN security applications that can provision and orchestrate security services and mechanisms.

This section provides an example of an SDN security application the illustrates the second goal. We examine the topic of SDN security in detail in Chapter 16, "Security."

OpenDaylight DDoS Application

In 2014, Radware, a provider of application delivery and application security solutions for virtual and cloud data centers, announced its contribution to the OpenDaylight Project with Defense4All, an open SDN security application integrated into OpenDaylight. Defense4All offers carriers and cloud providers **distributed denial of service (DDoS)** detection and mitigation as a native network service. Using the OpenDaylight SDN Controller that programs SDN-enabled networks to become part of the DoS/DDoS protection service itself, Defense4All enables operators to provision a DoS/DDoS protection service per virtual network segment or per customer.

Defense4All uses a common technique for defending against DDoS attacks, which consists of the following elements:

- Collection of traffic statistics and learning of statistics behavior of protected objects during peacetime. The normal traffic baselines of the protected objects are built from these collected statistics.
- Detection of DDoS attack patterns as traffic anomalies deviating from normal baselines.
- Diversion of suspicious traffic from its normal path to attack mitigation systems (AMSs) for traffic scrubbing, selective source blockage, and so on. Clean traffic exiting out of scrubbing centers is re-injected back into the packet's original destination.

Figure 6.7 shows the overall context of the Defense4All application. The underlying SDN network consists of a number of data plane switches that support traffic among client and server devices. Defense4All operates as an application that interacts with the controller over an OpenDaylight controller (ODC) northbound API. Defense4All supports a user interface for network managers that can either be a command line interface or a RESTful API. Finally,

Defense4All has an API to communicate with one or more AMSs.

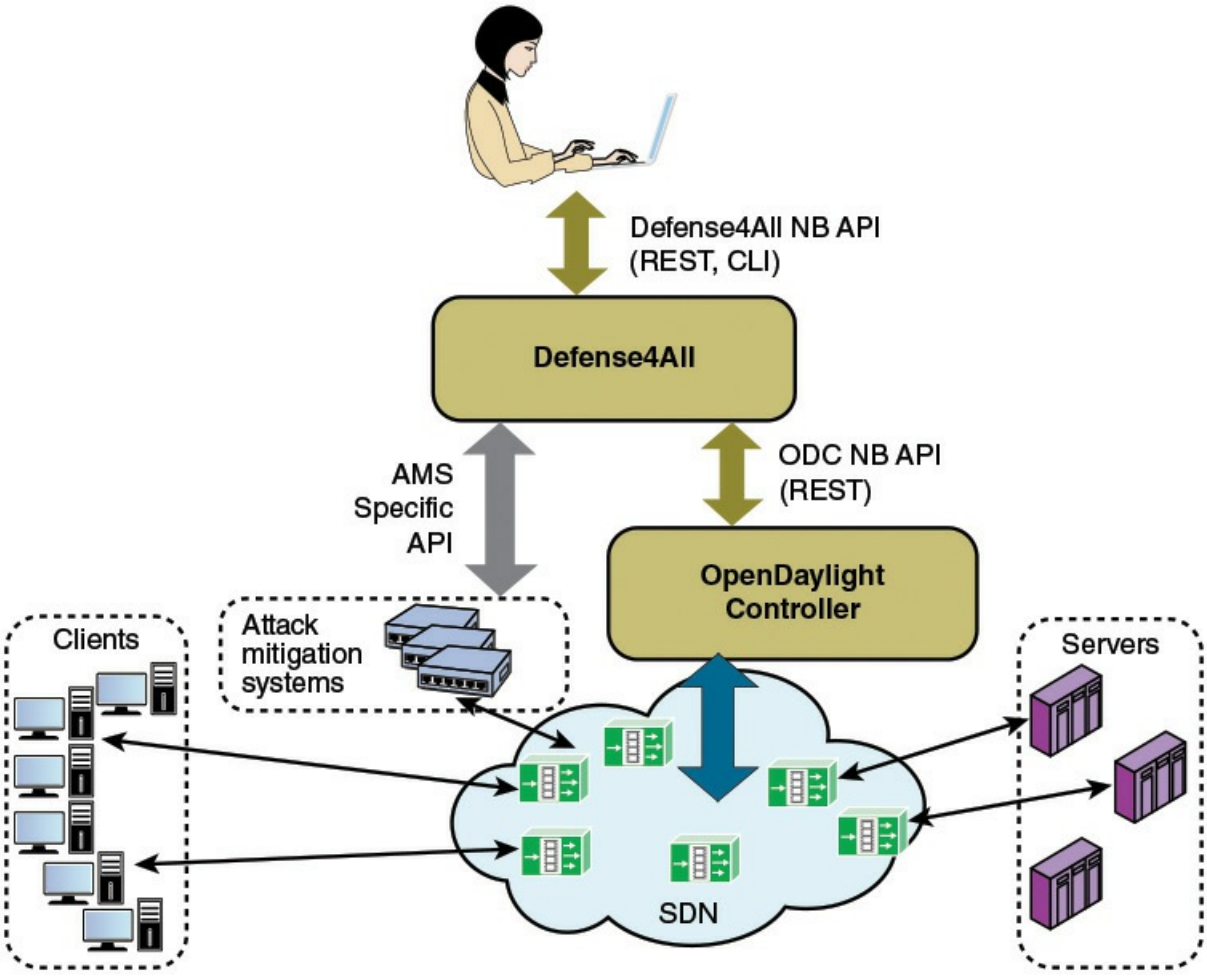


FIGURE 6.7 OpenDaylight DDoS Application

Administrators can configure Defense4All to protect certain networks and servers, known as protected networks (PNs) and protected objects (POs). The application instructs the controller to install traffic counting flows for each protocol of each configured PO in every network location through which traffic of the subject PO flows.

Defense4All then monitors traffic of all configured POs, summarizing readings, rates, and averages from all relevant network locations. If it detects a deviation from normal learned traffic behavior in a protocol (such as TCP, UDP, ICMP, or the rest of the traffic) of a particular PO, Defense4All declares an attack against that protocol in the subject PO. Specifically, Defese4All continuously calculates traffic averages for real time traffic it measured using OpenFlow; when real time traffic deviates by 80% from average then an attack is assumed.

To mitigate a detected attack, Defense4All performs the following procedure:

1. It validates that the AMS device is alive and selects a live connection to it. Currently, Defense4All is configured to work with Radware's AMS, known as DefensePro.
2. It configures the AMS with a security policy and normal rates of the attacked traffic. This provides the AMS with the information needed to enforce a mitigation policy until traffic

returns to normal rates.

3. It starts monitoring and logging syslogs arriving from the AMS for the subject traffic. As long as Defense4All continues receiving syslog attack notifications from the AMS regarding this attack, Defense4All continues to divert traffic to the AMS, even if the flow counters for this PO do not indicate any more attacks.
4. It maps the selected physical AMS connection to the relevant PO link. This typically involves changing link definitions on a virtual network, using OpenFlow.
5. It installs higher-priority flow table entries so that the attack traffic flow is redirected to the AMS and re-injects traffic from the AMS back to the normal traffic flow route. When Defense4All decides that the attack is over (no attack indication from either flow table counters or from the AMS), it reverts the previous actions: It stops monitoring for syslogs about the subject traffic, it removes the traffic diversion flow table entries, and it removes the security configuration from the AMS. Defense4All then returns to peacetime monitoring.

Figure 6.8 shows the principal software components of Defense4All. The overall application structure, referred to as a framework, contains the modules described in the list that follows.

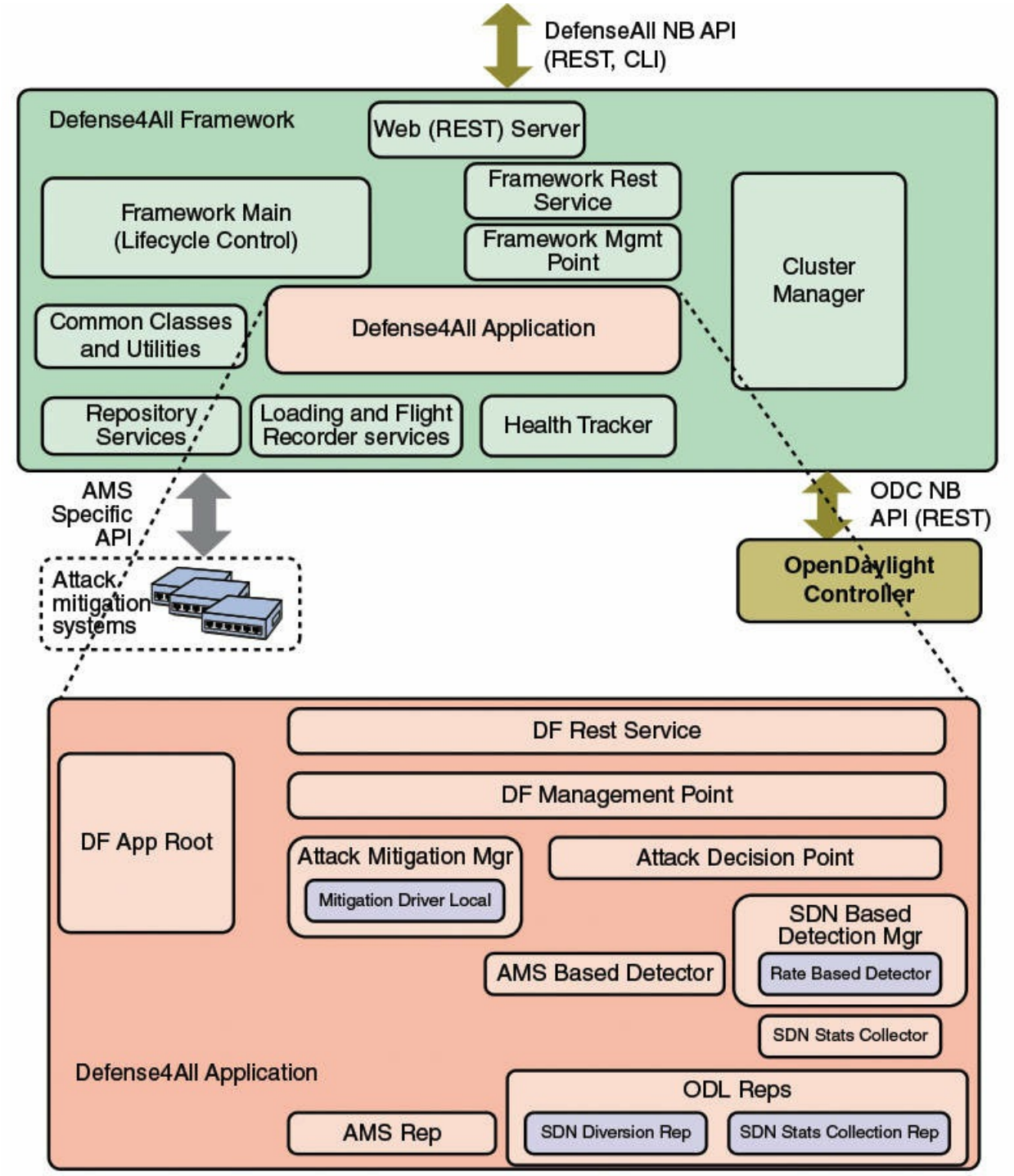


FIGURE 6.8 Defense4All Software Architecture Detail

- **Web (REST) Server:** Interface to network manager.
- **Framework Main:** Mechanism to start, stop, or reset the framework.
- **Framework REST Service:** Responds to user requests received through the web (REST) server.

- **Framework Management Point:** Coordinates and invokes control and configuration commands.
- **Defense4All Application:** Described subsequently.
- **Common Classes and Utilities:** A library of convenient classes and utilities from which any framework or SDN application module can benefit.
- **Repository Services:** One of the key elements in the framework philosophy is decoupling the compute state from the compute logic. All durable states are stored in a set of repositories that can be then replicated, cached, and distributed, with no awareness of the compute logic (framework or application).
- **Logging and Flight Recorder Services:** The logging service uses logs error, warning, trace, or informational messages. These logs are mainly for Defense4All developers. The Flight Recorder records events and metrics during run time from Java applications.
- **Health Tracker:** Holds aggregated run-time indicators of the operational health of Defense4All and acts in response to severe functional or performance deteriorations.
- **Cluster Manager**: Responsible for managing coordination with other Defense4All entities operating in a cluster mode.

The Defense4All Application module consists of the following elements.

- **DF App Root:** The root module of the application.
- **DF Rest Service:** Responds to Defense4All application REST requests.
- **DF Management Point:** The point to drive control and configuration commands. DFMgmtPoint in turn invokes methods against other relevant modules in the right order.
- **ODL Reps:** A pluggable module set for different versions of the ODC. Comprises two functions in two submodules: stats collection for and traffic diversion of relevant traffic.
- **SDN Stats Collector:** Responsible for setting “counters” for every PN at specified network locations (physical or logical). A counter is a set of OpenFlow flow entries in ODC-enabled network switches and routers. The module periodically collects statistics from those counters and feeds them to the SDNBasedDetectionMgr. The module uses the SDNStatsCollectionRep to both set the counters and read latest statistics from those counters. A stat report consists of read time, counter specification, PN label, and a list of trafficData information, where each trafficData element contains the latest bytes and packet values for flow entries configured for <protocol,port,direction> in the counter location. The protocol can be {tcp,udp,icmp,other ip}, the port is any Layer 4 port, and the direction can be {inbound, outbound}.
- **SDN Based Detection Manager:** A container for pluggable SDN-based detectors. It feeds stat reports received from the SDNStatsCollector to plugged-in SDN based detectors. It also feeds all SDN based detectors notifications from the AttackDecisionPoint about ended attacks (so as to allow reset of detection mechanisms). Each detector learns for each PN its normal traffic behavior over time, and notifies AttackDecisionPoint when it detects traffic anomalies.
- **Attack Decision Point:** Responsible for maintaining attack lifecycle, from declaring a

new attack, to terminating diversion when an attack is considered over.

- **Mitigation Manager:** A container for pluggable mitigation drivers. It maintains the lifecycle of each mitigation being executed by an AMS. Each mitigation driver is responsible for driving attack mitigations using AMSs in their sphere of management.
- **AMS Based Detector:** This module is responsible for monitoring/querying attack mitigation by AMSs.
- **AMS Rep:** Controls the interface to AMSs.

Figure 6.8 suggests the complexity of even a relatively straightforward SDN application.

Finally, it is worth noting that Radware has developed a commercial version of Defese4All, named DefenseFlow. DefenseFlow implements more sophisticated algorithms for attack detection based on fuzzy logic. The main benefit is that DefenseFlow has a greater ability to distinguish attack traffic from abnormal but legitimate high volume of traffic.

6.6 Data Center Networking

So far we've discussed three areas of SDN applications: traffic engineering, measurement and monitoring, and security. The provided examples of these applications suggest the broad range of use cases for them, in many different kinds of networks. The remaining three applications areas (data center networking, mobility and wireless, and information-centric networking) have use cases in specific types of networks.

Cloud computing, big data, large enterprise networks, and even in many cases, smaller enterprise networks, depend strongly on highly scalable and efficient data centers. [KREU15] lists the following as key requirements for data centers: high and flexible **cross-section bandwidth** and low latency, QoS based on the application requirements, high levels of resilience, intelligent resource utilization to reduce energy consumption and improve overall efficiency, and agility in provisioning network resources (for example, by means of network virtualization and orchestration with computing and storage).

With traditional network architectures, many of these requirements are difficult to satisfy because of the complexity and inflexibility of the network. SDN offers the promise of substantial improvement in the ability to rapidly modify data center network configurations, to flexibly respond to user needs, and to ensure efficient operation of the network.

The remainder of this subsection, examines two example data center SDN applications.

Big Data over SDN

A paper by Wang, et al., in the Proceedings of HotSDN'12 [WANG12], reports on an approach to use SDN to optimize data center networking for big data applications. The approach leverages the capabilities of SDN to provide application-aware networking. It also exploits characteristics of structured big data applications as well as recent trends in dynamically reconfigurable optical circuits. With respect to structured big data applications, many of these applications process data according to well-defined computation patterns, and also have a centralized management structure that makes it possible to leverage application-level information to optimize the network.

That is, knowing the anticipated computation patterns of the big data application, it is possible to intelligently deploy the data across the big data servers and, more significantly, react to changing application patterns by using SDN to reconfigure flows in the network.

Compared to electronic switches, optical switches have the advantages of greater data rates with reduced cabling complexity and energy consumption. A number of projects have demonstrated how to collect network-level traffic data and intelligently allocate optical circuits between endpoints (for example, top-of-rack switches) to improve application performance. However, circuit utilization and application performance can be inadequate unless there is a true application-level view of traffic demands and dependencies. Combining an understanding of the big data computation patterns with the dynamic capabilities of SDN, efficient data center networking configurations can be used to support the increasing big data demands.

Figure 6.9 shows a simple hybrid electrical and optical data center network, in which OpenFlow-enabled top-of-rack (ToR) switches are connected to two aggregation switches: an Ethernet switch and an optical circuit switch (OCS). All the switches are controlled by a SDN controller that manages physical connectivity among ToR switches over optical circuits by configuring the optical switch. It can also manage the forwarding at ToR switches using OpenFlow rules.

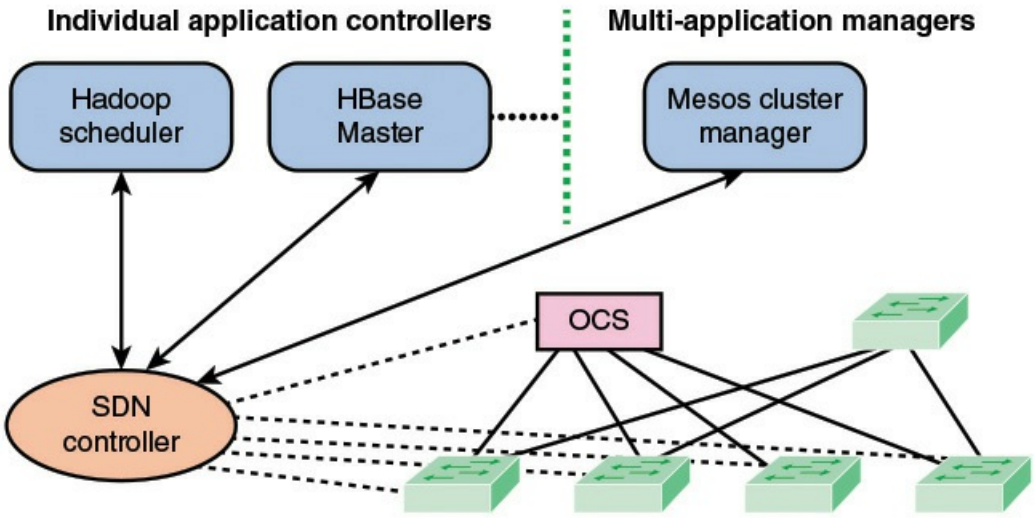


FIGURE 6.9 Integrated Network Control for Big Data Applications [WANG12]

The SDN controller is also connected to the Hadoop scheduler, which forms queues of jobs to be scheduled and the HBase Master controller of a relational database holding data for the big data applications. In addition, the SDN controller connects to a Mesos cluster manager. Mesos is an open source software package that provides scheduling and resource allocation services across distributed applications.

The SDN controller makes available network topology and traffic information to the Mesos cluster manager. In turn, the SDN controller accepts traffic demand request from Mesos managers.

With the organization of Figure 6.8, it is possible to set up a scheme whereby the traffic demands of big data applications are used to dynamically manage the network, using the SDN controller to manage this task.

Cloud Networking over SDN

Cloud Network as a Service (CloudNaaS) is a cloud networking system that exploits OpenFlow SDN capabilities to provide a greater degree of control over cloud network functions by the cloud customer [BENS11]. CloudNaaS enables users to deploy applications that include a number of network functions, such as virtual network isolation, custom addressing, service differentiation, and flexible interposition of various middleboxes. CloudNaaS primitives are directly implemented within the cloud infrastructure itself using high-speed programmable network elements, making CloudNaaS highly efficient.

Figure 6.10 illustrates the principal sequence of events in the CloudNaaS operation, as described in the list that follows.

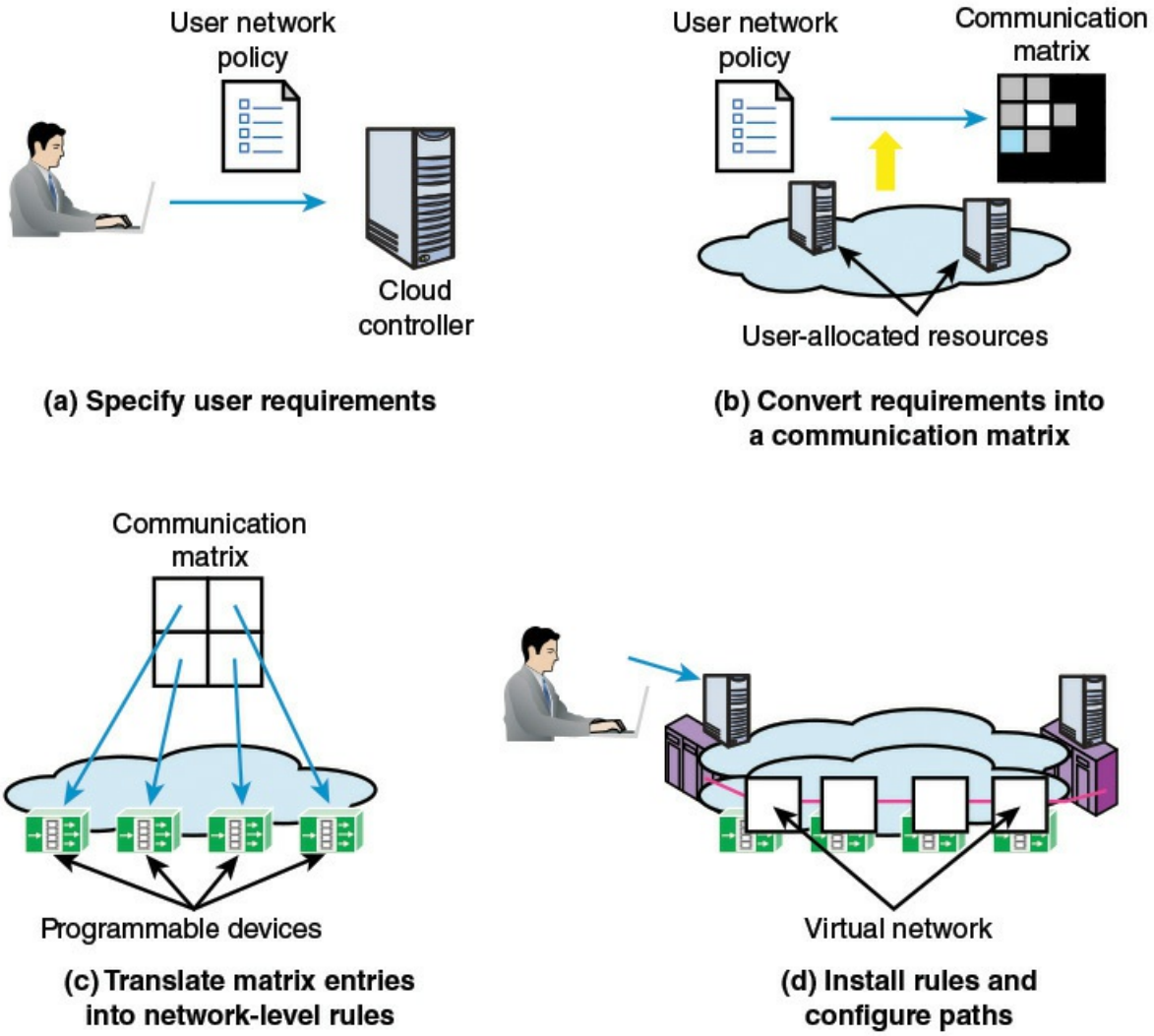


FIGURE 6.10 Various Steps in the CloudNaaS Framework

a. A cloud customer uses a simple policy language to specify network services required by the customer applications. These policy statements are issued to a cloud controller server operated by the cloud service provider.

b. The cloud controller maps the network policy into a communication matrix that defines desired communication patterns and network services. The matrix is used to determine the optimal placement of virtual machines (VMs) on cloud servers such that the cloud can satisfy the largest number of global policies in an efficient manner. This is done based on the knowledge of other customers' requirements and their current levels of activity.

c. The logical communication matrix is translated into network-level directives for data plane forwarding elements. The customer's VM instances are deployed by creating and placing the specified number of VMs.

d. The network-level directives are installed into the network devices via OpenFlow.

The abstract network model seen by the customer consists of VMs and virtual network segments that connect VMs together. Policy language constructs identify the set of VMs that comprise an application and define various functions and capabilities attached to virtual network segments. The main constructs are as follows:

- **address:** Specify a customer-visible custom address for a VM.
- **group:** Create a logical group of one or more VMs. Grouping VMs with similar functions makes it possible for modifications to apply across the entire group without requiring changing the service attached to individual VMs.
- **middlebox:** Name and initialize a new virtual middlebox by specifying its type and a configuration file. The list of available middleboxes and their configuration syntax is supplied by the cloud provider. Examples include intrusion detection and audit compliance systems.
- **networkservice:** Specify capabilities to attach to a virtual network segment, such as Layer 2 broadcast domain, link QoS, and list of middleboxes that must be traversed.
- **virtualnet:** Virtual network segments connect groups of VMs and are associated with network services. A virtual network can span one or two groups. With a single group, the service applies to traffic between all pairs of VMs in the group. With a pair of groups, the service is applied between any VM in the first group and any VM in the second group. Virtual networks can also connect to some predefined groups, such as EXTERNAL, which indicates all endpoints outside of the cloud.

Figure 6.11 provides an overview of the architecture of CloudNaaS. Its two main components are a cloud controller and a network controller. The cloud controller provides a base **Infrastructure as a Service (IaaS)** service for managing VM instances. The user can communicate standard IaaS requests, such as setting up VMs and storage. In addition, the network policy constructs enable the user to define the virtual network capabilities for the VMs. The cloud controller manages a software programmable virtual switch on each physical server in the cloud that supports network services for tenant applications, including the management of the user-defined virtual network segments. The cloud controller constructs the communication matrix and transmits this to the network controller.

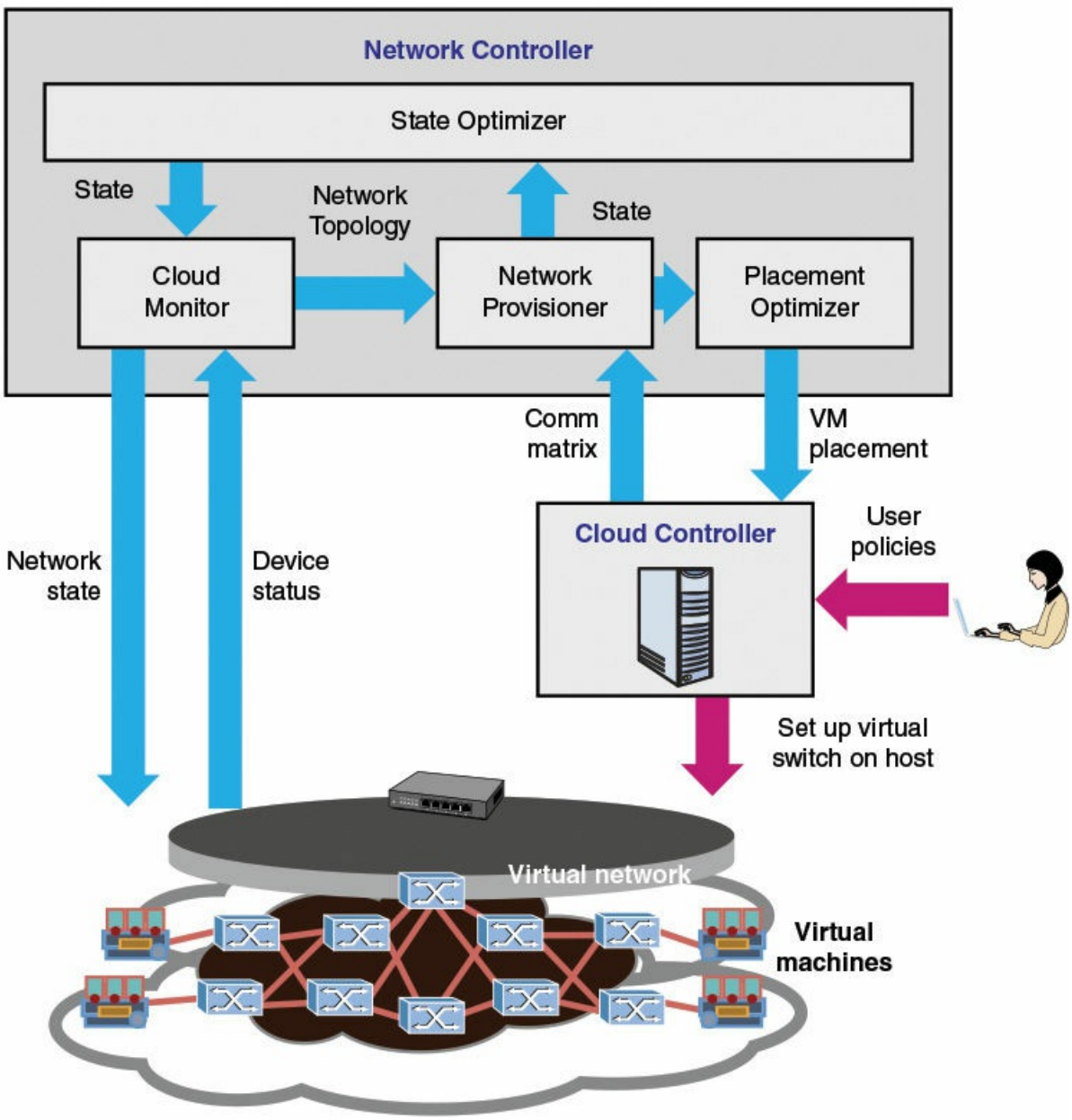


FIGURE 6.11 CloudNaaS Architecture

The network controller uses the communication matrix to configure data plane physical and virtual switches. It generates virtual networks between VMs and provides VM placement directives to the cloud controller. It monitors the traffic and performance on the cloud data plane switches and makes changes to the network state as needed to optimize use of resources to meet tenant requirements. The controller invokes the placement optimizer to determine the best location to place VMs within the cloud (and reports it to the cloud controller for provisioning). The controller then uses the network provisioner module to generate the set of configuration commands for each of the programmable devices in the network and configures them accordingly to instantiate the tenant's virtual network segment.

Thus, CloudNaaS provides the cloud customer with the ability to go beyond simple requesting a

processing and storage resource, to defining a virtual network of VMs and controlling the service and QoS requirements of the virtual network.

6.7 Mobility and Wireless

In addition to all the traditional performance, security, and reliability requirements of wired networks, wireless networks impose a broad range of new requirements and challenges. Mobile users are continuously generating demands for new services with high quality and efficient content delivery independent of location. Network providers must deal with problems related to managing the available spectrum, implementing handover mechanisms, performing efficient load balancing, responding to QoS and QoE requirements, and maintaining security.

SDN can provide much-needed tools for the mobile network provider and in recent years a number of SDN-based applications for wireless network providers have been designed. [KREU15] lists the following SDN application areas, among others: seamless mobility through efficient handovers, creation of on-demand virtual access points, load balancing, downlink scheduling, dynamic spectrum usage, enhanced intercell interference coordination, per client / base station resource block allocations, simplified administration, easy management of heterogeneous network technologies, interoperability between different networks, shared wireless infrastructures, and management of QoS and access control policies.

SDN support for wireless network providers is an area of intense activity, and a wide range of application offerings is likely to continue to appear.

6.8 Information-Centric Networking

Information-centric networking (ICN), also known as content-centric networking, has received significant attention in recent years, mainly driven by the fact that distributing and manipulating information has become the major function of the Internet today. Unlike the traditional host-centric networking paradigm where information is obtained by contacting specified named hosts, ICN is aimed at providing native network primitives for efficient information retrieval by directly naming and operating on information objects.

With ICN, a distinction exists between location and identity, thus decoupling information for its sources. The essence of this approach is that information sources can place, and information users can find, information anywhere in the network, because the information is named, addressed, and matched independently of its location. In ICN, instead of specifying a source-destination host pair for communication, a piece of information itself is named. In ICN, after a request is sent, the network is responsible for locating the best source that can provide the desired information. Routing of information requests thus seeks to find the best source for the information, based on a location-independent name.

Deploying ICN on traditional networks is challenging, because existing routing equipment would need to be updated or replace with ICN-enabled routing devices. Further, ICN shifts the delivery model from host to user to content to user. This creates a need for a clear separation between the task of information demand and supply, and the task of forwarding. SDN has the potential to provide the necessary technology for deploying ICN because it provides for programmability of

the forwarding elements and a separation of control and data planes.

A number of projects have proposed using SDN capabilities to implement ICNs. There is no consensus approach to achieving this coupling of SDN and ICN. Suggested approaches include substantial enhancements/modifications to the OpenFlow protocol, developing a mapping of names into IP addresses using a hash function, using the IP option header as a name field, and using an abstraction layer between an OpenFlow (OF) switch and an ICN router, so that the layer, OF switch, and ICN router function as a single programmable ICN router.

The remainder of this section briefly introduces this last approach [NGUY13, NGUY14]. This approach is designed to provide OF switches with ICN functionality, without having to modify the OF switches. The approach is built on an open protocol specification and a software reference implementation of ICN known as CCNx. Before looking at the abstraction layer approach, a brief background on CCNx is needed.

CCNx

CCNx is being developed by the Palo Alto Research Center (PARC) as an open source project, and a number of implementations have been experimentally deployed.

CCNx

Communication in CCN is via two packet types: **Interest packets** and **Content packets**. A consumer requests content by sending an Interest packet. Any CCN node that receives the Interest and has named data that satisfies the Interest responds with a Content packet (also known as a Content). Content satisfies an Interest if the name in the Interest packet matches the name in the Content Object packet. If a CCN node receives an Interest, and does not already have a copy of the requested Content, it may forward the Interest toward a source for the content. The CCN node has forwarding tables that determine which direction to send the Interest. A provider receiving an Interest for which it has matching named content replies with a Content packet. Any intermediate node can optionally choose to cache the Content Object, and it can respond with a cached copy of the Content Object the next time it receives an Interest packet with the same name.

The basic operation of a CCN node is similar to an IP node. CCN nodes receive and send packets over faces. A **face** is a connection point to an application, or another CCN node, or some other kind of channel. A face may have attributes that indicate expected latency and bandwidth, broadcast or multicast capability, or other useful features. A CCN node has three main data structures:

- **Content Store:** Holds a table of previously seen (and optionally cached) Content packets.
- **Forwarding Information Base (FIB):** Used to forward Interest packets toward potential data sources.

- **Pending Interest Table (PIT):** Used to keep track of Interests forwarded upstream by that CCN node toward the content source so that Content packets later received can be sent back to their requestors.

The details of how content sources become known and how routes are set up through the CCN network are beyond our scope. Briefly, content providers advertise names of content and routes are established through the CCN network by cooperation among the CCN nodes.

ICN relies substantially on in-network caching—that is, to cache content on the path from content providers to requesters. This **on-path caching** achieves good overall performance but is not optimal as content may be replicated on routers, thus reducing the total volume of content that can be cached. To overcome this limitation, **off-path caching** can be used, which allocates content to well-defined off-path caches within the network and deflects the traffic off the optimal path toward these caches that are spread across the network. Off-path caching improves the global hit ratio by efficiently utilizing the network-wide available caching capacity and permits to reduce egress links' bandwidth usage.

Use of an Abstraction Layer

The central design issue with using an SDN switch (in particular an OF switch) to function as an ICN router is that the OF switch forwards on the basis of fields in the IP packet, especially the destination IP address, and an ICN router forwards on the basis of a content name. In essence, the proposed approach hashes the name inside the fields with an OF switch can process.

Figure 6.12 shows the overall architecture of the approach. To link a CCNx node software module with an OF switch, an abstraction layer, called the wrapper, is used. The wrapper pairs a switch interface to a CCNx face, decodes and hashes content names in CCN messages into fields that an OF switch can process (for example, IP addresses, port numbers). The large naming space offered by these fields limits the probability of having collisions between two different content names. The forwarding tables in the OF switch are set to forward based on the contents of the hashed fields. The switch does not "know" that the contents of these fields are no longer legitimate IP addresses, TCP port numbers, and so forth. It forwards as always, based on the values found in the relevant fields of incoming IP packets.

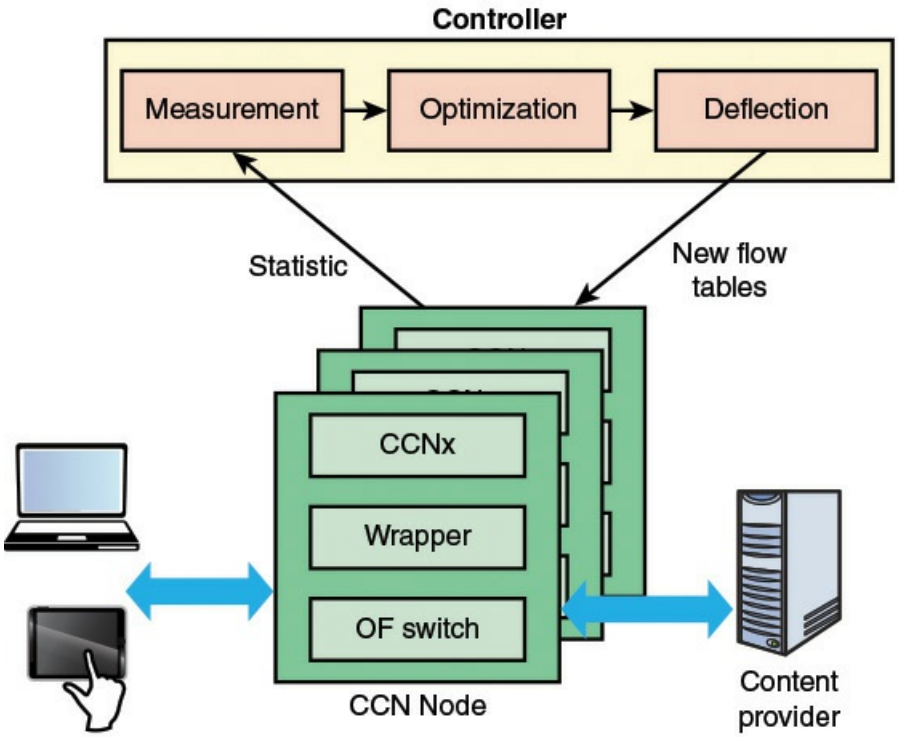


FIGURE 6.12 ICN Wrapper Approach

The abstraction layer solves the problem of how to provide CCN functionality using current OF switches. For efficient operation, two additional challenges need to be addressed: how to measure the popularity of content accurately and without a large overhead, and how to build and optimize routing tables to perform deflection. To address these issues, the architecture calls for three new modules in the SDN controller:

- **Measurement:** Content popularity can be inferred directly from OF flow statistics. The measurement module periodically queries and processes statistics from ingress OF switches to return the list of most popular content.
- **Optimization:** Uses the list of most popular contents as an input for the optimization algorithm. The objective is to minimize the sum of the delays over deflected contents under the following constraints: (1) each popular content is cached at exactly one node, (2) caching contents at a node does not exceed node's capacity, and (3) caching should not cause link congestion.
- **Deflection:** Uses the optimization results to build a mapping, for every content, between the content name (by means of addresses and ports computed from the content name hash) and an outgoing interface toward the node where the content is cached (for example, ip.destination = hash(content name), action = forward to interface 1).

Finally, mappings are installed on switches' flow tables using the OF protocol such that subsequent Interest packets can be forwarded to appropriate caches.

Figure 6.13 shows the flow of packets. The OpenFlow switch forwards every packet it receives from other ports to the wrapper, and the wrapper forwards it to the CCNx module. The OpenFlow switch needs to help the wrapper identify the switch source port of the packet. To

achieve this, the OF switch is configured to set the ToS value of all packets it receives to the corresponding incoming port value and then forward all of them to the wrapper's port.

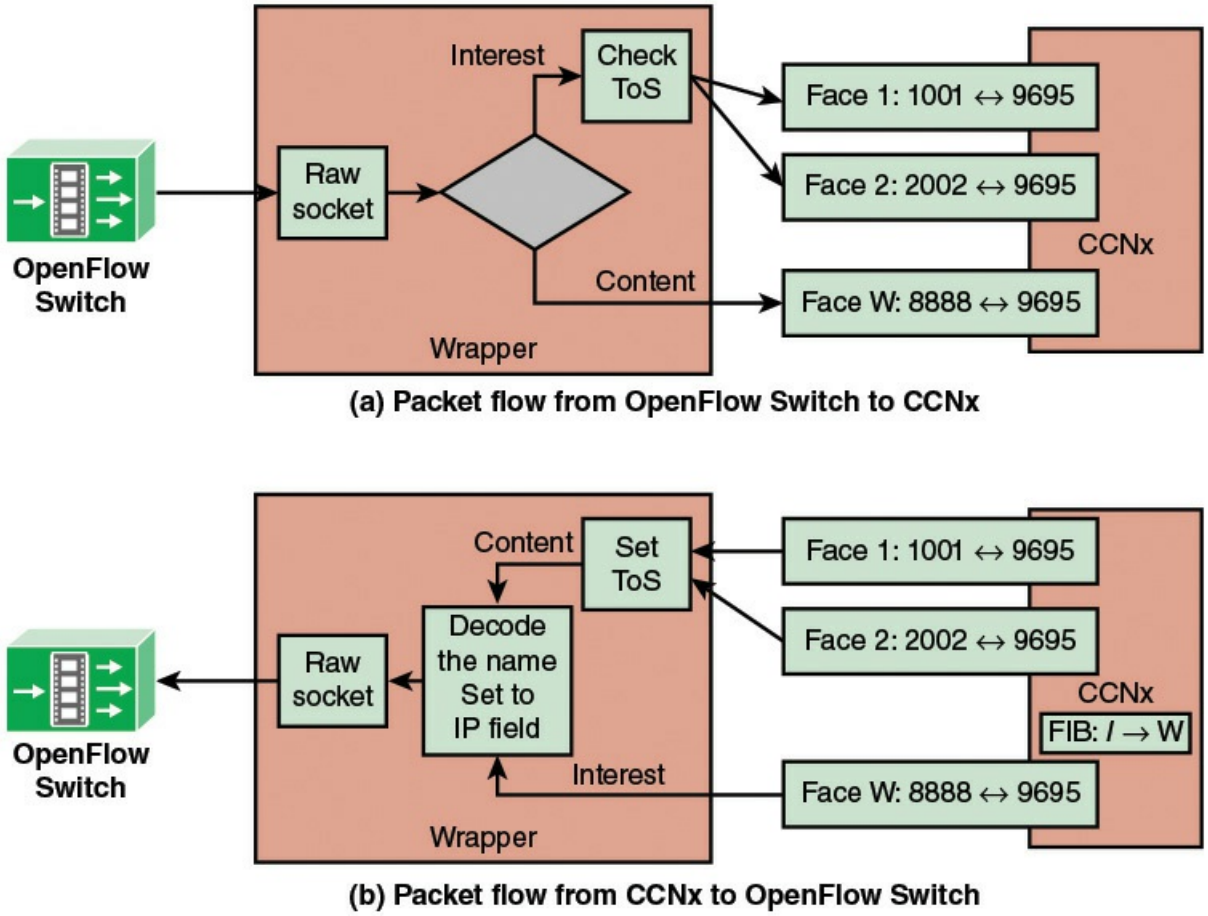


FIGURE 6.13 Packet Flow Between CCNx and OpenFlow Switch

The wrapper maps a face of CCNx to an interface (that is, port) of OpenFlow switches using ToS value. Face W is a special face between wrapper and the CCNx module. W receives every Content packet from the wrapper and is used to send every Interest packet from CCNx to the wrapper.

Part a of Figure 6.13 shows how incoming packets from the OF switch are handled by the wrapper. For an Interest packet, the wrapper extracts the face value from the ToS field and forwards the packet to the corresponding CCNx face. If the CCNx node holds a copy of the requested content, it composes a Content packet and returns it back to the incoming face. Otherwise, it forwards this Interest to face W and updates its PIT accordingly. Upon Content packet arrival from the OF switch, the wrapper forwards it directly to face W.

Part b of Figure 6.13 shows the operation of the wrapper on packets received from the CCNx module. For content packets, it sets the ToS field accordingly, specifying the output port. Then, for any packet, it decodes the packet to extract the content name related to the packet. The name is hashed and the source IP address of the packet is set to correspond to the hashed value. Finally, the wrapper forwards the packets to OF switches. Content packets are returned to their corresponding incoming face. Interest packets have the ToS value set to zero so they are

forwarded to next hop by the OF switch.

Thus, the use of the wrapper abstraction layer provides basic ICN functionality plus deflection functionality without needing to modify the CCNx module or the OpenFlow switch.

6.9 Key Terms

After completing this chapter, you should be able to define the following terms.

abstraction

abstraction layer

CloudNaaS

content-centric networking (CCN)

cross-section bandwidth

distributed denial of service (DDoS)

distribution abstraction

forwarding abstraction

Frenetic

information-centric networking (ICN)

Infrastructure as a Service (IaaS)

measurement and monitoring

network services abstraction layer

off-path caching

on-path caching

PolicyCop

specification abstraction

traffic engineering

Part III: Virtualization

The basic idea is that the several components in any complex system will perform particular subfunctions that contribute to the overall function.

—*The Sciences of the Artificial*, Herbert Simon

Interest in, and work on, network functions virtualization (NFV) began later than the corresponding software-defined network (SDN) effort. However, NFV and a broader conception of virtual networking have come to play a role of equal importance to that of SDN in modern networking. Part III is devoted to a broad and thorough presentation of NFV concepts, technology, and applications, and a discussion of network virtualization. Chapter 7 begins by introducing the concept of virtual machine and then looks of the use of virtual machine technology to develop NFV-based networking environments. Chapter 8 is a detailed looks at the functionality of NFV elements and also relates NFV to SDN. Chapter 9 looks at traditional concepts of virtual networks, and then at the more modern approach to network virtualization, and finally introduces the concept of software defined infrastructure.

Chapter 7. Network Functions Virtualization: Concepts and Architecture

> It has been found useful in many installations to use an operating system to simulate the existence of several machines on a single physical set of hardware. The IBM VM/370 operating system is one example. This technique allows an installation to multiprogram several different operating systems (or different versions of the same operating system) on a single physical machine. The dynamic-address-translation hardware allows such a simulator to be efficient enough to be used, in many cases, in production mode.
>
> —Architecture of the IBM System/370, *Communications of the ACM*, January 1978, Richard Case and Adris Padegs

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

- Understand the concept of virtual machine.
- Explain the difference between Type 1 and Type 2 hypervisors.
- List and explain the key benefits of NFV.
- List and explain the key requirements for NFV.
- Present an overview of the NFV architecture.

This chapter and the succeeding two chapters focus on the application of **virtualization** for modern networking. Virtualization encompasses a variety of technologies for managing computing resources, providing a software translation layer, known as an abstraction layer, between the software and the physical hardware. Virtualization turns physical resources into logical, or virtual, resources. Virtualization enables users, applications, and management software operating above the abstraction layer to manage and use resources without needing to be aware of the physical details of the underlying resources. In this chapter and the next, we focus on the use of virtual machine (VM) technology as a basis for the new concept of network functions virtualization (NFV). Chapter 9, "Network Virtualization," deals with virtual networks and the concept of network virtualization.

7.1 Background and Motivation for NFV

NFV originated from discussions among major network operators and carriers about how to improve network operations in the high-volume multimedia era. These discussions resulted in the publication of the original NFV white paper, *Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action* [ISGN12]. In this white paper, the group listed as the overall objective of NFV is leveraging standard IT virtualization technology to consolidate many network equipment types onto industry standard high-volume servers, switches, and storage, which could be located in data centers, network nodes, and in the end-user premises.

The white paper highlights that the source of the need for this new approach is that networks include a large and growing variety of proprietary hardware appliances, leading to the following negative consequences:

- New network services may require additional different types of hardware appliances, and finding the space and power to accommodate these boxes is becoming increasingly difficult.
- New hardware means additional capital expenditures.
- Once new types of hardware appliances are acquired, operators are faced with the rarity of skills necessary to design, integrate, and operate increasingly complex hardware-based appliances.
- Hardware-based appliances rapidly reach end of life, requiring much of the procure-design-integrate-deploy cycle to be repeated with little or no revenue benefit.
- As technology and services innovation accelerates to meet the demands of an increasingly network-centric IT environment, the need for an increasing variety of hardware platforms inhibits the introduction of new revenue-earning network services.

The NFV approach moves away from dependence on a variety of hardware platforms to the use of a small number of standardized platform types, with virtualization techniques used to provide the needed network functionality. In the white paper, the group expresses the belief that the NFV approach is applicable to any data plane packet processing and control plane function in fixed and mobile network infrastructures.

In addition to providing a way to overcome the problems cited in the preceding list, NFV provides a number of other benefits. These are best examined in Section 7.4, after an introduction to VM technology and NFV concepts, in Sections 7.2 and 7.3, respectively.

7.2 Virtual Machines

Traditionally, applications have run directly on an operating system (OS) on a personal computer (PC) or on a server. Each PC or server would run only one OS at a time. Therefore, application vendors had to rewrite parts of its applications for each OS/platform they would run on and support, which increased time to market for new features/functions, increased the likelihood of defects, increased quality testing efforts, and usually led to increased price. To support multiple operating systems, application vendors needed to create, manage, and support multiple hardware and operating system infrastructures, a costly and resource-intensive process. One effective strategy for dealing with this problem is known as **hardware virtualization**. Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS. A machine running virtualization software can host numerous applications, including those that run on different operating systems, on a single hardware platform. In essence, the host operating system can support a number of **virtual machines (VMs)**, each of which has the characteristics of a particular OS and, in some versions of virtualization, the characteristics of a particular hardware platform.

Virtualization is not a new technology. During the 1970s, IBM mainframe systems offered the first capabilities that would allow programs to use only a portion of a system’s resources.

Various forms of that ability have been available on platforms since that time. Virtualization came into mainstream computing in the early 2000s when the technology was commercially available on x86 servers. Organizations were suffering from a surfeit of servers because of a Microsoft Windows-driven "one application, one server" strategy. Moore's Law drove rapid hardware improvements outpacing software's ability, and most of these servers were vastly underutilized, often consuming less than 5 percent of the available resources in each server. In addition, this overabundance of servers filled data centers and consumed vast amounts of power and cooling, thereby straining a corporation's ability to manage and maintain their infrastructure. Virtualization helped relieve this stress.

The Virtual Machine Monitor

The solution that enables virtualization is a **virtual machine monitor (VMM)**, or commonly known today as a **hypervisor**. This software sits between the hardware and the VMs acting as a resource broker (see Figure 7.1). Simply put, the hypervisor allows multiple VMs to safely coexist on a single physical server host and share that host's resources. The number of guests that can exist on a single host is measured as a **consolidation ratio**. For example, a host that is supporting six VMs is said to have a consolidation ration of 6 to 1, also written as 6:1 (see Figure 7.2). The initial hypervisors in the commercial space could provide consolidation ratios of between 4:1 and 12:1, but even at the low end, if a company virtualized all of their servers, they could remove 75 percent of the servers from their data centers. More important, they could remove the cost as well, which often ran into the millions or tens of millions of dollars annually. With fewer physical servers, less power and less cooling was needed. Also this leads to fewer cables, fewer network switches, and less floor space. Server consolidation became, and continues to be, a tremendously valuable way to solve a costly and wasteful problem. Today, more virtual servers are deployed in the world than physical servers, and virtual server deployment continues to accelerate.

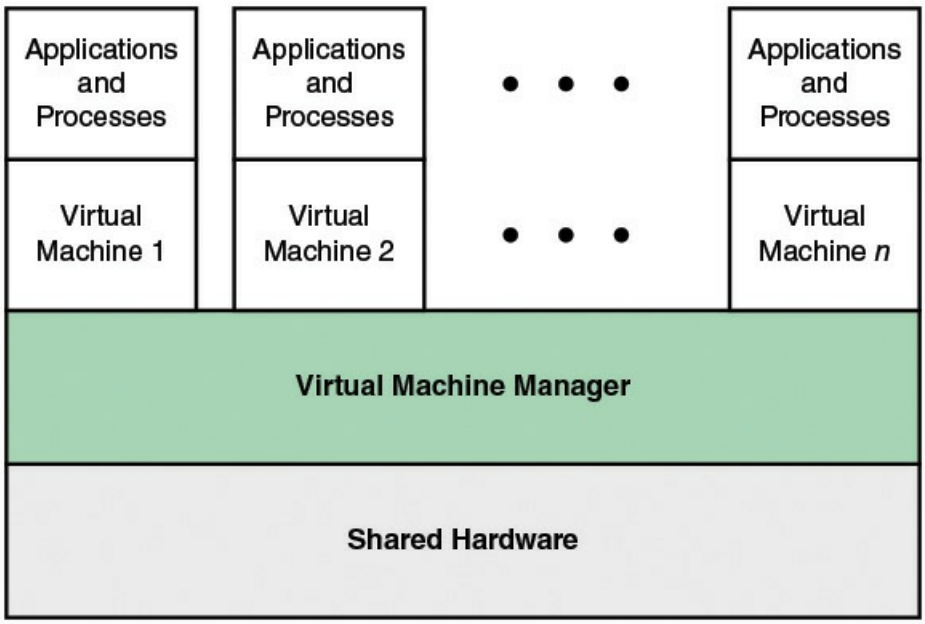


FIGURE 7.1 Virtual Machine Concept

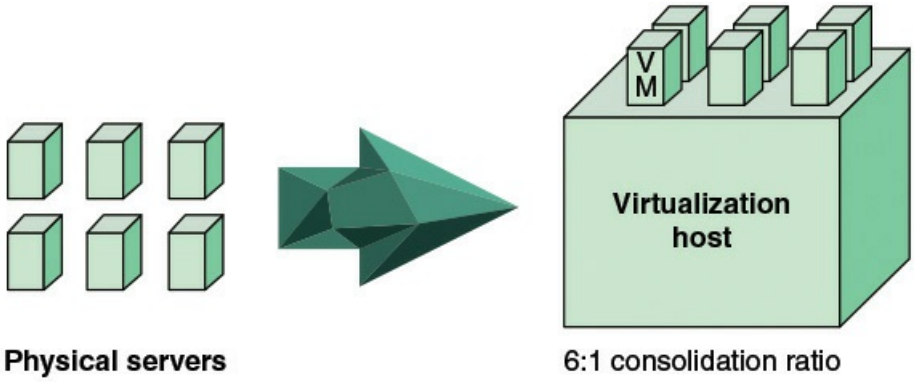


FIGURE 7.2 Virtual Machine Consolidation

The VM approach is a common way for businesses and individuals to deal with legacy applications and to optimize their hardware usage by maximizing the various kinds of applications that a single computer can handle. Commercial hypervisor offerings by companies such as VMware and Microsoft are widely used, with millions of copies having been sold. A key aspect of server virtualization is that, in addition to the capability of running multiple VMs on one machine, VMs can be viewed as network resources. Server virtualization masks server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. This makes it possible to partition a single host into multiple independent servers, conserving hardware resources. It also makes it possible to quickly migrate a server from one machine to another for load balancing or for dynamic switchover in the case of machine failure. Server virtualization has become a central element in dealing with big data applications and in implementing cloud computing infrastructures.

Architectural Approaches

Virtualization is all about abstraction. Much like an operating system abstracts the disk I/O commands from a user through the use of program layers and interfaces, virtualization abstracts the physical hardware from the VMs it supports. As noted already, virtual machine monitor, or hypervisor, is the software that provides this abstraction. It acts as a broker, or traffic cop, acting as a proxy for the guests (VMs) as they request and consume resources of the physical host.

A VM is a software construct that mimics the characteristics of a physical server. It is configured with some number of processors, some amount of RAM, storage resources, and connectivity through the network ports. Once that VM is created, it can be powered on like a physical server, loaded with an operating system and applications, and used in the manner of a physical server. Unlike a physical server, this virtual server sees only the resources it has been configured with, not all the resources of the physical host itself. This isolation allows a host machine to run many VMs, each running the same or different copies of an operating system, sharing RAM, storage, and network bandwidth, without problems. An operating system in a VM accesses the resource that is presented to it by the hypervisor. The hypervisor facilitates the translation of I/O from the VM to the physical server devices, and back again to the correct VM. To achieve this, certain privileged instructions that a “native” operating system would be executing on its host’s

hardware now trigger a hardware trap and are run by the hypervisor as a proxy for the VM. This creates some performance degradation in the virtualization process though over time both hardware and software improvements have minimalized this overhead.

VMs are made up of files. A typical VM can consist of just a few files. There is a configuration file that describes the attributes of the VM. It contains the server definition, how many virtual processors (vCPUs) are allocated to this VM, how much RAM is allocated, which I/O devices the VM has access to, how many network interface cards (NICs) are in the virtual server, and more. It also describes the storage that the VM can access. Often that storage is presented as virtual disks that exist as additional files in the physical file system. When a VM is powered on, or instantiated, additional files are created for logging, for memory paging, and other functions. That a VM consists of files makes certain functions in a virtual environment much simpler and quicker than in a physical environment. Since the earliest days of computers, backing up data has been a critical function. Because VMs are already files, copying them produces not only a backup of the data but also a copy of the entire server, including the operating system, applications, and the hardware configuration itself.

To create a copy of a physical server, additional hardware needs to be acquired, installed, configured, loaded with an operating system, applications, and data, and then patched to the latest revisions, before being turned over to the users. This provisioning can take weeks or even months depending on the processes in places. Because a VM consists of files, by duplicating those files, in a virtual environment there is a perfect copy of the server available in a matter of minutes. There are a few configuration changes to make (server name and IP address to name two), but administrators routinely stand up new VMs in minutes or hours, as opposed to months.

Another method to rapidly provision new VMs is through the use of templates. A template provides a standardized group of hardware and software settings that can be used to create new VMs configured with those settings. Creating a new VM from a template consists of providing unique identifiers for the new VM and having the provisioning software build a VM from the template and adding in the configuration changes as part of the deployment.

In addition to consolidation and rapid provisioning, virtual environments have become the new model for data center infrastructures for many reasons. One of these is increased availability. VM hosts are clustered together to form pools of computer resources. Multiple VMs are hosted on each of these servers and in the case of a physical server failure, the VMs on the failed host can be quickly and automatically restarted on another host in the cluster. Compared with providing this type of availability for a physical server, virtual environments can provide higher availability at significantly lower cost and less complexity. For servers that require greater availability, fault tolerance is available in some solutions through the use of shadowed VMs in running lockstep to ensure that no transactions are lost in the event of a physical server failure, again without increased complexity. One of the most compelling features of virtual environments is the capability to move a running VM from one physical host to another, without interruption, degradation, or impacting the users of that VM. vMotion, as it is known in a VMware environment, or Live Migration, as it is known in others, is used for a number of crucial tasks. From an availability standpoint, moving VMs from one host to another without incurring downtime allows administrators to perform work on the physical hosts without impacting operations. Maintenance can be performed on a weekday morning instead of during scheduled downtime on a weekend. New servers can be added to the environment and older servers removed without impacting the applications. In addition to these manually initiated migrations,

migrations can be automated depending on resource usage. If a VM starts to consume more resources than normal, other VMs can be automatically relocated to hosts in the cluster where resources are available, ensuring adequate performance for all the VMs and better overall performance. These are simple examples that only scratch the surface of what virtual environments offer.

As mentioned earlier, the hypervisor sits between the hardware and the VMs. There are two types of hypervisors, distinguished by whether there is another operating system between the hypervisor and the host. A Type 1 hypervisor (see part a of Figure 7.3) is loaded as a thin software layer directly into a physical server, much like an operating system is loaded. Once it is installed and configured, usually within a matter of minutes, the server can then support VMs as guests. In mature environments, where virtualization hosts are clustered together for increased availability and load balancing, a hypervisor can be staged on a new host, the new host can be joined to an existing cluster, and VMs can be moved to the new host without any interruption of service. Some examples of Type 1 hypervisors are VMware ESXi, Microsoft Hyper-V, and the various open source Xen variants. This idea that the hypervisor is loaded onto the "bare metal" of a server is usually a difficult concept for people to understand. They are more comfortable with a solution that works as a traditional application, program code that is loaded on top of a Microsoft Windows or UNIX/Linux operating system environment. This is exactly how a Type 2 hypervisor (see part b of Figure 7.3) is deployed. Some examples of Type 2 hypervisors are VMware Workstation and Oracle VM Virtual Box.

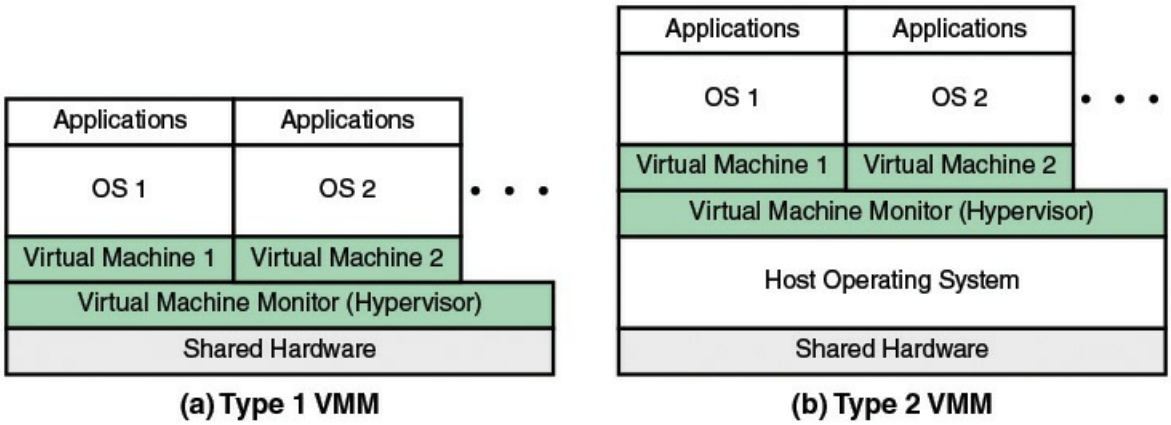


FIGURE 7.3 Type 1 and Type 2 Virtual Machine Monitors

There are some important differences between the Type 1 and the Type 2 hypervisors. A Type 1 hypervisor is deployed on a physical host and can directly control the physical resources of that host, whereas a Type 2 hypervisor has an operating system between itself and those resources and relies on the operating system to handle all the hardware interactions on the hypervisor's behalf. Typically, Type 1 hypervisors perform better than Type 2 because Type 1 hypervisors do not have that extra layer. Because a Type 1 hypervisor doesn't compete for resources with an operating system, there are more resources available on the host, and by extension, more VMs can be hosted on a virtualization server using a Type 1 hypervisor. Type 1 hypervisors are also considered to be more secure than the Type 2 hypervisors. VMs on a Type 1 hypervisor make resource requests that are handled external to that guest, and they cannot affect other VMs or the hypervisor by which they are supported. This is not necessarily true for VMs on a Type 2 hypervisor and a malicious guest could potentially affect more than itself. A Type 1 hypervisor