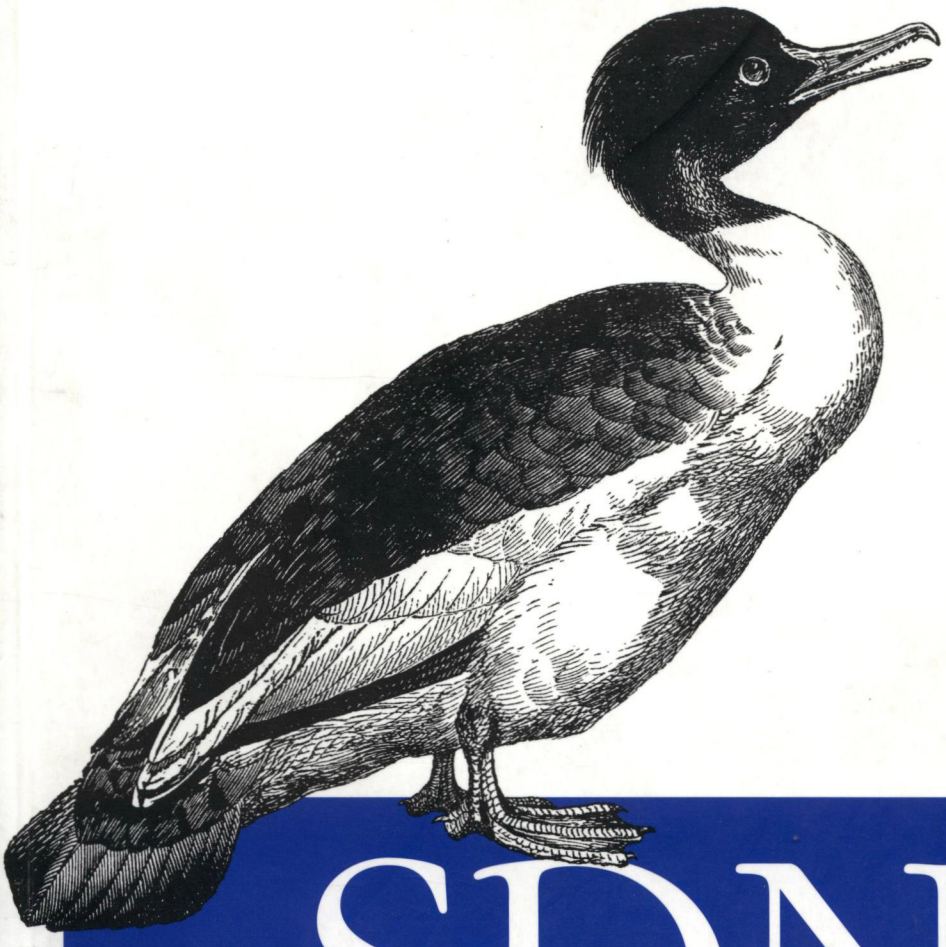
软件定义网络（影印版）

An Authoritative Review of Network Programmability Technologies



SDN

Software Defined Networks

[美] *Thomas D. Nadeau & Ken Gray* 著

O'REILLY®

人民邮电出版社
POSTS & TELECOM PRESS

软件定义网络（影印版）

SDN: Software Defined Networks

[美] Thomas D. Nadeau & Ken Gray 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社

北京

图书在版编目（CIP）数据

软件定义网络 : 英文 / (美) 纳多 (Nadeau, T. D.) , (美) 格雷 (Gray, K.) 著. -- 影印本. -- 北京 : 人民邮电出版社, 2014.1
ISBN 978-7-115-33574-6

Ⅰ. ①软… Ⅱ. ①纳… ②格… Ⅲ. ①计算机网络－研究－英文 Ⅳ. ①TP393

中国版本图书馆CIP数据核字(2013)第263301号

版 权 声 明

◆ 著　[美]Thomas D.Nadeau　Ken Gray

责任编辑　傅道坤

责任印制　程彦红　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号

邮编　100164　电子邮件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

三河市海波印务有限公司印刷

◆ 开本：787×1000　1/16

印张：24

字数：447 千字　2014 年 1 月第 1 版

印数：1－2 500 册　2014 年 1 月河北第 1 次印刷

著作权合同登记号　图字：01-2013-8314 号

定价：69.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

Foreword by David Meyer

Although the ideas underlying software-defined networking (SDN) have only recently come into the public consciousness, a few of us who are active in the research, operator, and vendor communities immediately saw the applicability of SDN-like techniques to data center and service provider environments (and beyond). In addition to the explosion of innovative thinking going on in the research community, we also saw SDN as a programmatic way to optimize, monetize, and scale networks of all kinds.

In 2011, the first organization dedicated to the growth and success of SDN began with the Open Networking Foundation (ONF). Among its stated missions was to evolve the OpenFlow protocol from its academic roots to a commercially viable substrate for building networks and networking products. Within two years, the ONF's membership had grown to approximately 100 entities, representing the diverse interest and expectations for SDN. Against this backdrop, many of us were looking at the wider implications of the ideas underlying SDN, and in the process, generalized SDN to include not only OpenFlow but other forms of network programmability as well.

Early on in this process, both Tom Nadeau and Ken Gray realized that SDN was really about general network programmability and the associated interfaces, protocols, data models, and APIs. Using this insight, they helped to organize the SDN Birds of a Feather session at IETF 82, in Taipei, to investigate this more general SDN model. At that meeting, Tom presented a framework for software-defined networks that envisioned SDN as a generalized mechanism for network programmability. This work encouraged the community to take a more general view of SDN and eventually led to the formation of the Interface to the Routing System Working Group in the IETF.

Since that time, in addition to their many contributions to Internet technologies, Tom and Ken have become well-respected senior members of the SDN community. They are active participants in the core SDN industry activities and develop products for the SDN market. Some of the key industry activities that Tom and Ken drive include the ONF, IETF, ETSI, industry events such as SDN Summit 2012/2013, as well as open source consortia such as the Open Daylight Project. This book draws on their deep

understanding and experience in the field and offers a unique perspective on SDN. It will help you understand not only the technology but also how it is being developed, standardized, and deployed.

Tom and Ken are eminently qualified to give you a lucid understanding of the technology and the common-sense use and deployment of network programmability techniques. In particular, their book is an excellent and practical introduction to the fundamentals of SDN and is filled with innumerable anecdotes explaining the ideas and the background behind the development of SDN. So if you are interested in writing SDN applications, building SDN capable networks, or just understanding what SDN is, this book is for you!

—David Meyer
CTO and Chief Scientist, Brocade Communications

Foreword by David Ward

Technological shifts that affect how developers and engineers build and design their business architectures are monumental. These shifts are not applicable to Moore's law and tend to be transformations that affect not only the IT landscape but the business landscape as well. These shifts tend to occur every 8 to 10 years and have a long-lasting impact on how people build, consume, and distribute technologies. They also force people to frame their business opportunities in new ways.

In 1996, Gartner coined the term "service-oriented architecture." By 2000, it had taken center stage with the core purpose of allowing for the easy cooperation of a large number of computers connected over a network to exchange information via services without human interaction. There was no need to make underlying changes to the program or application itself. Essentially, it took on the same role as a single operating system on one machine and applied it to the entire infrastructure of servers, allowing for more usable, flexible, and scalable applications and services to be built, tested, deployed, and managed. It introduced web services as the de facto way to make functional building blocks accessible over standard Internet protocols independent of platforms and languages—allowing for faster and easier development, testing, deployment, and manageability of IT infrastructures. SOA drastically changed the way developers, their managers, and the business looked at technology.

When you look at software-defined networking, you see similarities. The network is the cornerstone of IT in that it can enable new architectures that in turn create new business opportunities. In essence, it allows IT to become more relevant than ever and the enabler of new business. The network is now the largest business enabler if architected and utilized in the correct way—allowing for the network, server, and storage to be tied together to enable the principles of SOA to be executed at the network layer. SDN and APIs to the network change the accessibility to programming intent and receiving state from the network and services, thus overcoming the traditional view that the network has to be built and run by magicians. However, when SOA principles become applied to the networking layer, the network becomes more accessible, programmable, and

flexible, allowing organizations to actually shift IT at the speed that the business moves, all while adding increased value to the business in new ways.

But what is a software-defined network? There are many camps that have varying definitions. When broken down into simple terms, it needs to be looked at as an approach or architecture to not only simplify your network but also to make it more reactive to the requirements of workloads and services placed in the network. IT infrastructure needs to move at the speed of business opportunities and must enable new ways to do business quickly, flexibly, and faster than before. A pragmatic definition is this: SDN functionally enables the network to be accessed by operators programmatically, allowing for automated management and orchestration techniques; application of configuration policy across multiple routers, switches, and servers; and the decoupling of the application that performs these operations from the network device's operating system.

As SDN becomes increasingly the buzzword of multiple industries, it's worthwhile to take a look at why SDN came about. Historically, network configuration state has remained largely static, unchanged, and commonly untouchable. Manual configuration and CLI-based configuration on a device-by-device basis was the norm, and network management constituted the basic "screen scraping" or use of Expect scripts as a way to solve manageability problems and core scalability issues (cut-and-paste methodology). The highest end of programmatic interfaces included XML interfaces and onboard Perl, Tk/Tcl, and Expect. However, when you're dealing with multiple routers, switches, and servers working as a system (and services that are routing traffic across multiple domains with different users, permissions, and policies), control and management state needs to be applied across the network as an operation. Element-by-element management simply doesn't provide enough flexibility and agility or the notion of dynamic or ephemeral data (configuration and state not persistently held in the config file). But as service-oriented architecture principles started to shift southbound down the stack and the realization of their application at the networking layer was recognized, new architectures—coupled with advancements in networking—allowed for software-defined networking to emerge and users to realize the power that the network was capable of in new ways.

Yes, it's true that there is a history of protocol interfaces to routers, switches, servers, gateways, and so on. Decades of deployment of the current Internet that program dynamic data associated with subscribers, sessions, and applications does currently exist and is widely deployed. These protocol servers (e.g., Radius, Diameter, PCMM, COPS, 3GPP) all could be considered early forms of SDN, so why aren't they? What's a bit different now is that one major functionality of the SDN architecture is the ability to write applications on top of a platform that customizes data from different sources or data bases into one network-wide operation.

SDN is also an architecture that allows for a centrally managed and distributed control, management, and data plane, where policy that dictates the forwarding rules is

centralized, while the actual forwarding rule processing is distributed among multiple devices. In this model, application policy calculation (e.g., QoS, access control lists, and tunnel creation) happens locally in real time and the quality, security, and monitoring of policies are managed centrally and then pushed to the switching/routing nodes. This allows for more flexibility, control, and scalability of the network itself, and the use of templates, variables, multiple databases of users, and policies all working in combination to derive or compile the desired configuration and state to be downloaded to the routers and switches. What's key to understand is that SDN doesn't replace the control plane on the router or switch. It augments them. How? By having a view of the entire network all at once versus only from one position in the topology (e.g., the router or switch). The marriage of dynamic routing and signaling and a centralized view is incredibly powerful. It enables the fastest possible protection in the event of a failure, the greatest resiliency, and the ability to place services into a network in one command. The two technologies working together are really a major step forward that wasn't previously in our toolbox.

There are a few variations on the SDN theme and some oft spoken components to be considered. OpenFlow is one, which architecturally separates the control and management planes from the data plane on the networking device. This allows for a centralized controller to manage the flows in the forwarding nodes. However, OpenFlow is only one protocol and one element of SDN. There are many other protocols now. Some examples include I2RS, PCE-P, BGP-LS, FORCES, OMI, and NetConf/Yang. All of these are also open standards. What's important to remember is that SDN is not a protocol; it's an operational and programming architecture.

What do we get from SDN? The architecture brings the network and networking data closer to the application layer and the applications closer to the networking layer. As practiced in SOA, no longer is there the need for a human element or scripting languages to act as humans to distribute data and information bidirectionally because APIs and tooling now have evolved in a way that this can be delivered in a secure and scalable way via open interfaces and interoperability. The data in the network (e.g., stats, state, subscriber info, service state, security, peering, etc.) can be analyzed and used by an application to create policy intent and program the network into a new configuration. It can be programmed this way persistently or only ephemerally.

Programmability (i.e., the ability to access the network via APIs and open interfaces) is central to SDN. The notion of removing the control and management planes to an off-switch/router application connected to the networking device by SDN protocols is equally important. This off-box application is really what software developers would call a "platform," as it has its own set of APIs, logic, and the ability for an application to make requests to the network, receive events, and speak the SDN protocols. What's key here is that programmers don't need to know the SDN protocols because they write to the controller's APIs. Programmers don't need to know the different configuration syntax or semantics of different networking devices because they program to a set of APIs

on the controller that can speak to many different devices. Different vendors, eras of equipment, and classes of equipment (e.g., transport, simple switches, wireless base stations, subscriber termination gateways, peering routers, core routers, and servers) all are on the trajectory to be able to be programmed by the SDN protocols that plug into the bottom of the controller. The programmer only uses the APIs on the top of the controller to automate, orchestrate, and operate the network. This doesn't necessarily mean there is a grand unification theory of controllers and one to serve all layers and functions of networking, but what it does mean is that the network now has been abstracted and is being programmed off box. Thus, when integrated into an IaaS (Infrastructure as a Service) layer in a stack, OSS, or IT system, the network is being automated and orchestrated as fast as users log onto the net and as fast as workloads are being spun up on servers.

The use of new tooling practices typically utilized by system administrators and new available to network operators are related to the whole SDN movement. Tools such as Puppet, Chef, CFEngine, and others are being used to automate and orchestrate the network in new ways as plug-ins can now be created to utilize the network data via the open interfaces of the network. Controller APIs also allow for easier and faster ways to build and apply policy across the network in multiple languages and with integration into existing tools such as IDEs (NetBeans, Eclipse, et al.). This allows for a better user experience for network engineers versus the traditionally used CLI model.

Before we dig into examples, it's important to understand what SDN actually solves and why there is a shift to this particular architecture. As networks evolve and new services are deployed, it's critical to implement new ways for users to more easily provision and orchestrate network resources in real time. By implementing this, cost can be reduced by the automation of moving resources around faster and more reliably, and by allowing the network to respond directly to a request from an application (versus the intervention by a human). This allows for operators to use programmatic (scalable) control versus manual to create and apply these services in a way that is simpler than a command-line interface. Additionally, it enables the ability to utilize new resources from the network (user data, traffic path information, etc.) and create new types of applications that can control policy for the network in a scalable fashion. It also allows for the optimization of infrastructure, services, and applications by allowing for new network data and capabilities to be extended and applied into the aforementioned architecture, creating new ways to not only optimize existing applications but also to insert new services or offerings that can provide a better user experience or create a new offering or advanced feature that could be monetized.

As SDN evolves, it's important to look at some implementations to understand why it's so critical for multiple industries (e.g., video delivery, user services and mobile, cable and broadband, security, and provider edge) to embrace. Where SDN reaches its potential, however, is when you look at it for not just programming the network functions and scaling those across your infrastructure, but also for actually tying server, storage,

and the network together for new use cases. In this case, systems can actually interact with each other, allowing for more infrastructure flexibility, whether physical, virtual, or hybrid.

Traffic policy and rerouting based on network conditions and/or regulation shifts are also common applications, as are the insertion of new services or data into applications that may be able to more clearly prioritize bandwidth for a user that pays a premium amount for faster connection speeds. When you apply SDN and a centralized management plane that is separate from the data plane, you can more quickly make decisions on where data traffic can be rerouted, as this can occur programmatically with software interfaces (APIs), versus on-the-box CLI methodology.

One advanced use case is the hybrid cloud. In this case, an application may run in a private cloud or data center yet utilize the public cloud when the demand for computing capacity spikes or cost can be reduced. Historically, cloud bursting was typically used only in environments with non-mission critical applications or services, but with the network tie-in and software principles applied, the use case shifts. Applications now remain in compliance with the IT organizations' policies and regulations. The application can also retain its dependency model if it is reliant on different data or information that it typically has on premises versus off, or in the public cloud environment. It also allows for the application to run across different platforms regardless of where the application was built.

As we look at SDN, we must also consider Network Functions Virtualization and how this ties into the broader infrastructure and virtualization picture. The transition from physical to virtual is one that is leading many of these changes in the industry. By tying the hardware (physical) to software (virtual), including network, server, and storage, there's the opportunity to virtualize network services and have them orchestrated as fast as any other workload. Tie this via programmatic interfaces to the WAN, and you can absolutely guarantee service delivery. SDN coupled with NFV is a pivotal architectural shift in both computing and networking. This shift is marked by dynamic changes to infrastructure to closely match customer demand, analytics to assist in predicting performance requirements, and a set of management and orchestration tools that allow network functions and applications to scale up, down, and out with greater speed and less manual intervention. This change affects how we build cloud platforms for applications and at the most basic level must provide the tools and techniques that allow the network to respond to changing workload requirements as quickly as the platforms that leverage them. It also allows workload requirements to include network requirements and have them satisfied.

It's important to note that not all networks are the same, and that's why it's critical to understand the importance of the underlying infrastructure when abstracting control from the network—either from physical or virtual devices. Network Functions Virtualization is simply the addition of virtual or off-premises devices to augment traditional

infrastructure. However, the tie to both the on- and off-premises offerings must be considered when running applications and services to ensure a seamless experience not just for the organization running the applications or services but also for the consumer of the services (whether they be enterprise and in-house users or external customers).

So why should you care? From a technical perspective, SDN allows for more flexibility and agility as well as options for your infrastructure. By allowing data to be controlled centrally and tied into not just the network, but also the storage and server, you get a more cohesive view on performance, speed, traffic optimization, and service guarantees. With programmatic interfaces (APIs) that can be exposed in multiple languages and utilized with tools, your operators and administrators can more quickly respond to the demand of the business side of the house or external customer needs. They can now apply policies for other development organizations in-house to allow them network data to more effectively spin up server farms or even build applications with network intelligence built in for faster, better performing applications. By allowing for the data to be exposed in a secure and scalable way, the entire IT organization benefits, and with faster development and deployment cycles and easier delivery of new services, so too does the business. The promise that SOA gave developers—write once, run anywhere—can now be fully realized with the underlying network's ability to distribute information across the enterprise, access, WAN, and data center (both physical and virtual). This allows for applications to break free from the boundaries of the OSS and management platforms that had previously limited their ability to run in different environments.

The IT industry is going through a massive shift that will revolutionize the way users build, test, deploy, and monetize their applications. With SDN, the network is now closer to applications (and vice versa), allowing for a new breed of smarter, faster, and better performing applications. It enables the network to be automated in new ways, providing more flexibility and scalability for users, and unleashes the potential for business cost savings and revenue-generating opportunities. It's a new era in networking and the IT industry overall, and it will be a game-changing one. Check out this book—it's required reading.

—David Ward
CTO, Cisco Systems

Preface

The first question most readers of an O'Reilly book might ask is about the choice of the cover animal. In this case, "why a duck?" Well, for the record, our first choice was a unicorn decked out in glitter and a rainbow sash.

That response always gets a laugh (we are sure you just giggled a little), but it also brings to the surface a common perception of software-defined networks among many experienced network professionals. Although we think there is some truth to this perception, there is certainly more meat than myth to this unicorn.

So, starting over, the better answer to that first question is that the movement of a duck[1] is not just what one sees on the water; most of the action is under the water, which

1. The real answer is that one of the authors has a fondness for ducks, as he raises Muscovy Ducks on his family farm.

you can't easily see. Under the waterline, some very muscular feet are paddling away to move that duck along. In many ways, this is analogous to the progress of software-defined networks.

The surface view of SDN might lead the casual observer to conclude a few things. First, defining what SDN is, or might be, is something many organizations are frantically trying to do in order to resuscitate their business plans or revive their standards-developing organizations (SDOs). Second, that SDN is all about the active rebranding of existing products to be this mythical thing that they are not. Many have claimed that products they built four or five years ago were the origins of SDN, and therefore everything they have done since is SDN, too.

Along these lines, the branding of seemingly everything anew as SDN and the expected hyperbole of the startup community that SDN has been spawning for the past three or four years have also contributed negatively toward this end.

If observers are predisposed by their respective network religions and politics to dismiss SDN, it may seem like SDN is an idea adrift.

Now go ahead and arm yourself with a quick pointer to the Gartner hype-cycle.[2] We understand that perspective and can see where that cycle predicts things are at.

Some of these same aspects of the present SDN movement made us lobby *hard* for the glitter-horned unicorn just to make a point—that we see things *differently*.

For more than two years, our involvement in various customer meetings, forums, consortia, and SDOs discussing the topic, as well as our work with many of the startups, converts, and early adopters in the SDN space, leads us to believe that something worth noting *is* going on *under* the waterline. This is where much of the real work is going on to push the SDN effort forward toward a goal of what we think is optimal operational efficiency and flexibility for networks and applications that utilize those networks.

There is real evidence that SDN has finally started a *new* dialogue about network programmability, control models, the modernization of application interfaces to the network, and true openness around these things.

In that light, SDN is not constrained to a single network domain such as the data center —although it is true that the tidal wave of manageable network endpoints hatched via virtualization is a prime mover of SDN at present. SDN is also not constrained to a single customer type (e.g., research/education), a single application (e.g., data center orchestration), or even a single protocol/architecture (e.g., OpenFlow). Nor is SDN constrained to a single architectural model (e.g., the canonical model of a centralized controller and a group of droid switches). We hope you see that in this book.

2. *http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp*

At the time of writing of the first edition of this book, both Thomas Nadeau and Ken Gray work at Juniper Networks in the Platform Systems Division Chief Technologist's Office. We both also have extensive experience that spans roles both with other vendors, such as Cisco Systems, and service providers, such as BT and Bell Atlantic (now Verizon). We have tried our best to be inclusive of everyone that is relevant in the SDN space without being encyclopedic on the topic still providing enough breadth of material to cover the space. In some cases, we have relied on references or examples that came from our experiences with our most recent employer (Juniper Networks) in the text, only because they are either part of a larger survey or because alternative examples on the topic are net yet freely available for us to divulge. We hope the reader finds any bias to be accidental and not distracting or overwhelming. If this can be corrected or enhanced in a subsequent revision, we will do so. We both agree that there are likely to be many updates to this text going forward, given how young SDN still is and how rapidly it continues to evolve.

Finally, we hope the reader finds the depth and breadth of information presented herein to be interesting and informative, while at the same time evocative. We give our opinions about topics, but only after presenting the material and its pros and cons in as unbiased a manner as possible.

We do hope you find unicorns, fairy dust, and especially lots of paddling feet in this book.

Assumptions

SDN is a new approach to the current world of networking, but it is still networking. As you get into this book, we're assuming a certain level of networking knowledge. You don't have to be an engineer, but knowing how networking principles work—and frankly, don't work—will aid your comprehension of the text.

You should be familiar with the following terms/concepts:

OSI model
: The Open Systems Interconnection (OSI) model defines seven different layers of technology: physical, data link, network, transport, session, presentation, and application. This model allows network engineers and network vendors to easily discuss and apply technology to a specific OSI level. This segmentation lets engineers divide the overall problem of getting one application to talk to another into discrete parts and more manageable sections. Each level has certain attributes that describe it and each level interacts with its neighboring levels in a very well-defined manner. Knowledge of the layers above layer 7 is not mandatory, but understanding that interoperability is not always about electrons and photons will help.

Switches

These devices operate at layer 2 of the OSI model and use logical local addressing to move frames across a network. Devices in this category include Ethernet in all its variations, VLANs, aggregates, and redundancies.

Routers

These devices operate at layer 3 of the OSI model and connect IP subnets to each other. Routers move packets across a network in a hop-by-hop fashion.

Ethernet

These broadcast domains connect multiple hosts together on a common infrastructure. Hosts communicate with each other using layer 2 media access control (MAC) addresses.

IP addressing and subnetting

Hosts using IP to communicate with each other use 32-bit addresses. Humans often use a dotted decimal format to represent this address. This address notation includes a network portion and a host portion, which is normally displayed as 192.168.1.1/24.

TCP and UDP

These layer 4 protocols define methods for communicating between hosts. The Transmission Control Protocol (TCP) provides for connection-oriented communications, whereas the User Datagram Protocol (UDP) uses a connectionless paradigm. Other benefits of using TCP include flow control, windowing/buffering, and explicit acknowledgments.

ICMP

Network engineers use this protocol to troubleshoot and operate a network, as it is the core protocol used (on some platforms) by the ping and traceroute programs. In addition, the Internet Control Message Protocol (ICMP) is used to signal error and other messages between hosts in an IP-based network.

Data center

A facility used to house computer systems and associated components, such as telecommunications and storage systems. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning and fire suppression), and security devices. Large data centers are industrial-scale operations that use as much electricity as a small town.

MPLS

Multiprotocol Label Switching (MPLS) is a mechanism in high-performance networks that directs data from one network node to the next based on short path labels rather than long network addresses, avoiding complex lookups in a routing table. The labels identify virtual links (*paths*) between distant nodes rather than

endpoints. MPLS can encapsulate packets of various network protocols. MPLS supports a range of access technologies.

Northbound interface
: An interface that conceptualizes the lower-level details (e.g., data or functions) used by, or in, the component. It is used to interface with higher-level layers using the southbound interface of the higher-level component(s). In architectural overview, the northbound interface is normally drawn at the top of the component it is defined in, hence the name northbound interface. Examples of a northbound interface are JSON or Thrift.

Southbound interface
: An interface that conceptualizes the opposite of a northbound interface. The southbound interface is normally drawn at the bottom of an architectural diagram. Examples of southbound interfaces include I2RS, NETCONF, or a command-line interface.

Network topology
: The arrangement of the various elements (links, nodes, interfaces, hosts, etc.) of a computer network. Essentially, it is the topological structure of a network and may be depicted physically or logically. *Physical topology* refers to the placement of the network's various components, including device location and cable installation, while *logical topology* shows how data flows within a network, regardless of its physical design. Distances between nodes, physical interconnections, transmission rates, and/or signal types may differ between two networks, yet their topologies may be identical.

Application programming interfaces
: A specification of how some software components should interact with each other. In practice, an API is usually a library that includes specification for variables, routines, object classes, and data structures. An API specification can take many forms, including an international standard (e.g., POSIX), vendor documentation (e.g., the JunOS SDK), or the libraries of a programming language.

What's in This Book?

Chapter 1, Introduction
: This chapter introduces and frames the conversation this book engages in around the concepts of SDN, where they came from, and why they are important to discuss.

Chapter 2, Centralized and Distributed Control and Data Planes
: SDN is often framed as a decision between a distributed/consensus or centralized network control-plane model for future network architectures. In this chapter, we visit the fundamentals of distributed and central control, how the data plane is

generated in both, past history with both models,[3] some assumed functionality in the present distributed/consensus model that we may expect to translate into any substitute, and the merits of these models.

Chapter 3, OpenFlow

OpenFlow has been marketed either as equivalent to SDN (i.e., OpenFlow is SDN) or a critical component of SDN, depending on the whim of the marketing of the Open Networking Foundation. It can certainly be credited with sparking the discussion of the centralized control model. In this chapter, we visit the current state of the OpenFlow model.

Chapter 4, SDN Controllers

For some, the discussion of SDN technology is all about the management of network state, and that is the role of the SDN controller. In this chapter, we survey the controllers available (both open source and commercial), their structure and capabilities, and then compare them to an idealized model (that is developed in Chapter 9).

Chapter 5, Network Programmability

This chapter introduces network programmability as one of the key tenets of SDN. It first describes the problem of *the network divide* that essentially boils down to older management interfaces and paradigms keeping applications at arm's length from the network. In the chapter, we show why this is a bad thing and how it can be rectified using modern programmatic interfaces. This chapter firmly sets the tone for what concrete changes are happening in the real world of applications and network devices that are following the SDN paradigm shift.

Chapter 6, Data Center Concepts and Constructs

This chapter introduces the reader to the notion of the modern data center through an initial exploration of the historical evolution of the desktop-centric world of the late 1990s to the highly distributed world we live in today, in which applications—as well as the actual pieces that make up applications—are distributed across multiple data centers. Multitenancy is introduced as a key driver for virtualization in the data center, as well as other techniques around virtualization. Finally, we explain why these things form some of the keys to the SDN approach and why they are driving much of the SDN movement.

Chapter 7, Network Function Virtualization

In this chapter, we build on some of the SDN concepts that were introduced earlier, such as programmability, controllers, virtualization, and data center concepts. The chapter explores one of the cutting-edge areas for SDN, which takes key concepts and components and puts them together in such a way that not only allows one to

3. Yes, we have had centralized control models in the past!