CHAPTER

WHY SDN?

2

Networking devices have been successfully developed and deployed for several decades. Repeaters and bridges, followed by routers and switches, have been used in a plethora of environments, performing their functions of filtering and forwarding packets throughout the network toward their ultimate destination. Despite the impressive track record of these traditional technologies, the size and complexity of many modern deployments leaves them lacking. The reasons for this include the ever-increasing costs of owning and operating networking equipment, the need to accelerate innovation in networking and, in particular, the increasing demands of the modern data center. This chapter investigates these trends and describes how they are nudging networking technology away from traditional methods and protocols toward the more open and innovation-friendly paradigm of SDN.

2.1 EVOLUTION OF SWITCHES AND CONTROL PLANES

We begin with a brief review of the evolution of switches and control planes that has culminated in a fertile playing field for SDN. This complements the material presented in Sections 1.4 and 1.5. The reader may find it useful to employ Fig. 2.1 as a visual guide through the following sections, as it provides a graphical summary of this evolution and allows the reader to understand the approximate timeframes when different components of switching moved from software to hardware.

2.1.1 SIMPLE FORWARDING AND ROUTING USING SOFTWARE

In Chapter 1 we discussed the early days of computer networking, where almost everything other than the physical layer (layer one) was implemented in software. This was true for end-user systems as well as for networking devices. Whether the devices were bridges, switches, or routers, software was used extensively inside the devices in order to perform even the simplest of tasks, such as MAC-level forwarding decisions. This remained true even through the early days of the commercialized Internet in the early 1990s.

2.1.2 INDEPENDENCE AND AUTONOMY IN EARLY DEVICES

Early network device developers and standards-creators wanted each device to perform in an autonomous and independent manner, to the greatest extent possible. This was because networks were generally small and fixed, with large shared domains. A goal also was to simplify rudimentary management tasks and to make the networks as *plug and play* as possible. Their relatively static

Software Defined Networks. http://dx.doi.org/10.1016/B978-0-12-804555-8.00002-8

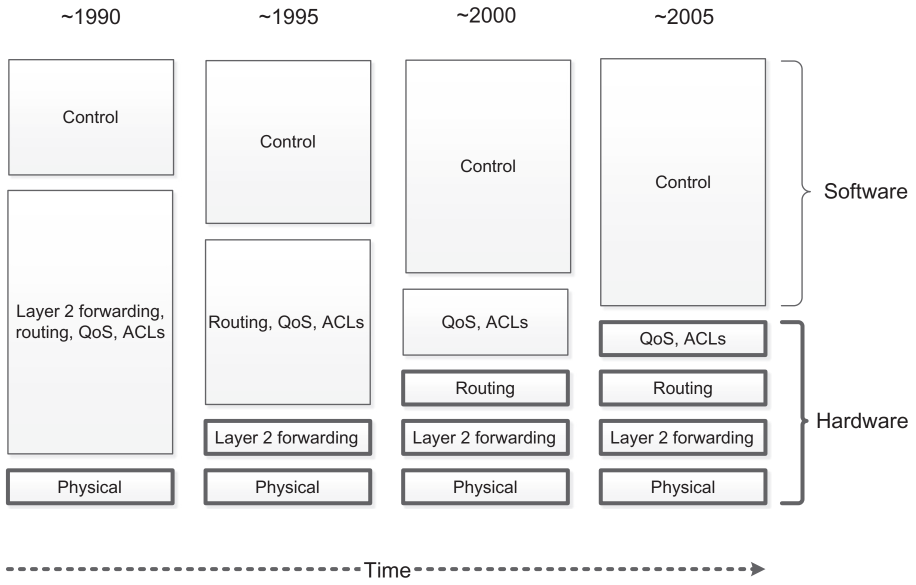


FIG. 2.1

Networking functionality migrating to hardware.

configuration needs were performed manually. Developers went to great lengths to implement this distributed environment with intelligence resident in every device. Whenever coordination between devices was required, collective decisions could be made through the collaborative exchange of information between devices.

Interestingly, many of the goals of this distributed model, such as simplicity, ease-of-use, and automatic recovery, are similar to the goals of SDN, but as the scale and complexity of networks grew, the current distributed model has become increasingly dysfunctional.

Examples of this distributed intelligence are the layer two (bridging) and layer three (routing) protocols, which involved negotiation between the devices in order to reach a consensus of how forwarding and routing would be performed. We introduced these protocols in Chapter 1 and provide more details on them below.

- **Spanning Tree Protocol**
  Basic layer two forwarding, also known as *transparent bridging*, can be performed independently by each switch in the network. However, certain topologies require an imposition of a hierarchy on the network in order to prevent loops which would cause broadcast radiation. The *Spanning Tree Protocol* (STP) is an example of the operation of autonomous devices participating in a distributed decision-making process in order to create and enforce a hierarchy on the network. The result is the correct operation of transparent bridging throughout the domain, at the expense of convergence latency and possibly arbitrary configuration. This solution was a trade-off between cost and complexity. Multiple paths could have been supported but at greater cost. While STP was adequate

when networks were of smaller scale, as networks grew the spanning tree solution has become problematic. These problems manifest themselves in a striking fashion when networks reach the scale of the modern data center. For example, IEEE 802.1D specifies the following default timers for STP: fifteen seconds for listening, fifteen seconds for learning, and twenty seconds for max-age timeout. In older networks, convergence times of thirty to fifty seconds were common. Such delays are not acceptable in today's data centers. As the scale of the layer two network grows, the likelihood of greater delays increases. The *Rapid Spanning Tree Protocol* (RSTP) protocol, specified in IEEE 802.1D-2004 [1], improves this latency significantly but unfortunately is not deployed in many environments.

- **Shortest Path Bridging**
  STP allowed only one active path to a destination, suffered from relatively slow convergence times, and was restricted to small network topologies. While the newer implementations of STP have improved the convergence times, the single active path shortcoming has been addressed in a new layer two protocol, SPB, introduced in Section 1.5.1. SPB is a mechanism for allowing multiple concurrent paths through a layer two fabric through collaborative and distributed calculation of shortest and most efficient paths, and sharing that information amongst the participating nodes in the meshed network. This characteristic is called *multipath*. SPB accomplishes this by utilizing IS-IS to construct a graph representing the layer two link-state topology. Once this graph exists, shortest path calculations are straightforward, though more complex than with spanning tree.
  To elaborate on what we mean by shortest path calculations, in Fig. 2.2 we depict a simple graph that can be used for calculating shortest paths in a network with six switches. The costs assigned to the various links may be assigned their values according to different criteria. A simple criterion is

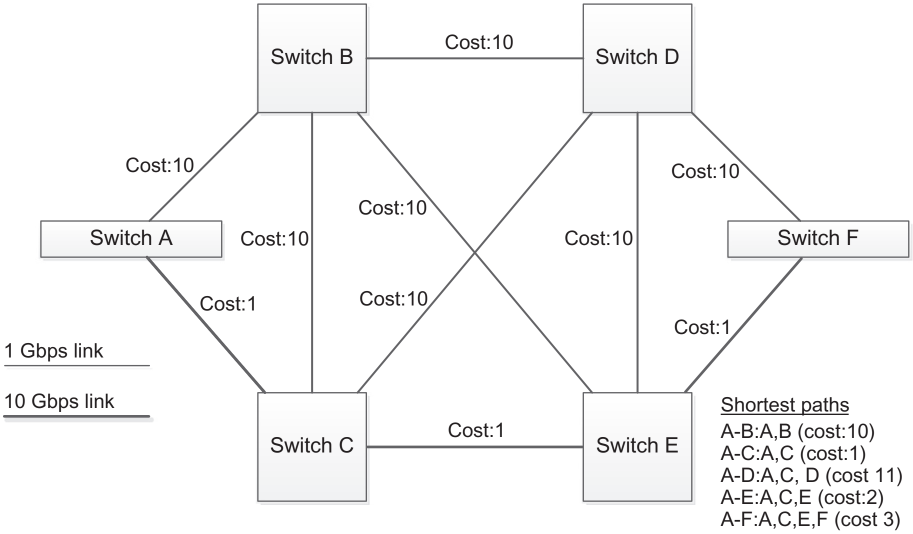


FIG. 2.2

Example of graph of network for shortest path calculation.

to make the cost of a network link inversely proportional to its bandwidth. Thus, the cost of transiting a ten Gbps link is one-tenth that of transiting a one Gbps link. When the shortest path calculation is complete, the node performing the calculation knows the least-cost path to any of the other nodes in the network. The least-cost path is considered the shortest path. For the sake of clarity, we should point out that IS-IS is used in the SPB context strictly for layer two path calculation. This differs from its classical application in calculating layer three routes, as described below. In the trivial example of Fig. 2.2, there is a single shortest path from node *A* to every other node. In real life networks it is common for there to be more than one least cost path between two nodes. The multipath characteristic of SPB would allow the traffic to be distributed across those multiple paths.

- **RIP, BGP, OSPF, and IS-IS**
  Routing at layer three requires cooperation between devices in order to know which routers are attaching which subnets to the network. In Chapter 1 we provided background on four routing protocols: RIP, BGP, OSPF, and IS-IS. These routing protocols involve the sharing of local routing information by each device, either at the edge of the network or as an intermediate node. Their collective sharing of information allows the routing state to converge as devices share their information with each other. Each router remains autonomous in terms of its ability to make routing decisions as packets arrive. This process is one of peers sharing and negotiating amongst themselves, without a centralized entity aiding in the decision.

2.1.3 SOFTWARE MOVES INTO SILICON

Vendors originally had to write their own software to implement even the basic functions like layer two forwarding and routing. Fig. 2.1 shows that over time these more basic functions moved from software into hardware. We now see most forwarding and filtering decisions implemented entirely in hardware. These decisions are driven by configured tables, set by the control plane software above. This shift of the lower-level decision-making from software to hardware has yielded tremendous improvements in the performance/cost ratio of switching equipment.

Today, switching devices are typically composed of hardware components such as *Application-Specific Integrated Circuits* (ASICs), *Field-Programmable Gate Arrays* (FPGAs), and *Ternary Content-Addressable Memories* (TCAMs). The combined power of these integrated circuits allows for the forwarding decisions to be made entirely in the hardware at line rate. This has become more critical as network speeds have increased from one Gbps to ten Gbps, to forty Gbps, and beyond. The hardware is now capable of handling all forwarding, routing, *Access Control List* (ACL), and QoS decisions. Higher-level control functions, responsible for network-wide collaboration with other devices, are implemented in software. This control software runs independently in each network device.

2.1.4 HARDWARE FORWARDING AND CONTROL IN SOFTWARE

The network device evolution we have recounted thus far has yielded the following current situation:

- **Bridging** (Layer Two Forwarding)
  Basic layer two MAC forwarding of packets is handled in the hardware tables.
- **Routing** (Layer Three Forwarding)

In order to keep up with today's high-speed links and to route packets at link speeds, layer three forwarding functionality is also implemented in hardware tables.

- **Advanced Filtering and Prioritization**
  General traffic management rules, such as ACLs, which filter, forward, and prioritize packets, are handled via hardware tables located in the hardware (e.g., in TCAMs), and accessed through low-level software.
- **Control**
  The control software used to make broader routing decisions and to interact with other devices in order to converge on topologies and routing paths is implemented in software that runs autonomously inside the devices. Since the current control plane software in networking devices lacks the ability to distribute policy information about such things as security, QoS and ACLs, these features must still be provisioned through relatively primitive configuration and management interfaces.

Given this landscape of (1) layer two and layer three hardware handling most forwarding tasks, (2) software in the device providing control plane functionality, and (3) policy implemented via configuration and management interfaces, an opportunity presents itself to simplify networking devices and move forward to the next generation of networking.

2.1.5 THE GROWING NEED FOR SIMPLIFICATION

In [2] the authors state that one of the major drivers for SDN is simplification. As time has passed, networking devices have become increasingly more complex. This is due in part to the existing independent and autonomous design of devices that make it necessary that so much intelligence be placed inside each device. Placing more functionality in hardware in some ways simplifies the device, but in other ways makes it more complicated because of the difficult handshakes and tradeoffs between handling packets in hardware versus software.

Attempting to provide simplicity by adding features to legacy devices tends to complicate implementations rather than simplifying them. An analogy to the evolution of the *Central Processing Unit* (CPU) can be made here. Over time CPUs became highly complex as they attempted to support more and more functions. Ultimately, another simpler, easier to use CPU model emerged which was called the *Reduced Instruction Set Computing* (RISC) model. In the same way the RISC architecture served as a reset to CPU architecture, so, too, SDN may serve as a simplifying reset for network equipment design.

In addition to simplifying the devices themselves, there is an opportunity to simplify the management of the networks of these devices. Rather than using primitive network management tools such as SNMP and CLI, network operators would prefer to use policy-based management systems. SDN may enable such solutions [3].

DISCUSSION QUESTION:

Explain the analogy between RISC CPU architecture in computation and SDN in networking.

2.1.6 MOVING CONTROL OFF OF THE DEVICE

We remind the reader that control software in our context is the intelligence that determines optimal paths and responds to outages and new networking demands. At its core, SDN is about moving that control software off of the device and into a centrally located compute resource which is capable of seeing the entire network and making decisions which are optimal, given a complete understanding of the situation. While we will discuss this in much greater detail in the chapters that follow, basically, SDN attempts to segregate network activities in the following manner:

- **Forwarding, Filtering, and Prioritization**
  Forwarding responsibilities, implemented in hardware tables, remain on the device. In addition, features such as filtering based on ACLs and traffic prioritization, are enforced locally on the device as well.
- **Control**
  Complicated control software is removed from the device and placed into a centralized controller, which has a complete view of the network and the ability to make optimal forwarding and routing decisions. There is a migration to a programming paradigm for the control plane. The basic forwarding hardware on the networking device is available to be programmed by external software on the controller. The control plane is no longer embedded, closed, closely coupled with the hardware, or optimized for particular embedded environments.
- **Application**
  Above the controller is where the network applications run, implementing higher-level functions and, additionally, participating in decisions about how best to manage and control packet forwarding and distribution within the network.

Subsequent chapters will examine in greater detail how this can be achieved, with a minimum of investment and change by networking vendors, while providing the maximum control and capability by the controller and its applications. The next section of this chapter will discuss another major reason why SDN is needed today—the cost of networking devices.

2.2 COST

Arguments related to the need for SDN often include cost as a driving factor for this shift [4,5]. In this section we consider the impact of the status quo in networking on the cost of designing, building, purchasing, and operating network equipment.

2.2.1 INCREASED COST OF DEVELOPMENT

Today’s autonomous networking devices must store, manage, and run the complicated control plane software that we discussed in the previous section. Over time, the result of this increased control plane sophistication is an increase in the amount of control plane software in the device, as can be seen in Fig. 2.1. Despite the overall downward trend in the cost of networking hardware, this growing complexity acts as an upward pressure on the hardware component costs due to the processing power required to run that advanced software as well as the storage capacity to hold it.

In Chapter 1 we described how software development outside the networking realm benefits greatly from the readily available open source software. For example, application server frameworks provide platforms which allow software developers to reuse code provided by those common frameworks, and, therefore to concentrate on solving domain-specific problems. Without the ability to leverage software functionality in this manner, each vendor has to develop, test, and maintain large amounts of redundant code, which is not the case in an open software environment. With the closed networking environment that is prevalent today, little such leverage is available, and, consequently, each vendor must implement all of the common functionality required by their devices. Common network functionality and protocols must be developed by every device vendor. This clearly increases the costs attributable to software development.

In recent years, silicon vendors have been producing *common off-the-shelf* (COTS) ASICs which are capable of speeds and functionality that rivals or surpasses the proprietary versions developed by networking hardware vendors. However, given the limited software leverage mentioned above, vendors are often unable to efficiently make use of these *merchant silicon* chips, since software must be re-engineered for each product line. So while their products may be quite profitable, NEMs must write and support larger amounts of software than would otherwise be necessary if networking devices were developed in truly open environments. The fact that such a large body of software must run on each and every network device serves to further increase this cost. There is additional overhead resulting from the requirement to support multiple versions of legacy protocols as well as keeping up with the latest protocols being defined by standards bodies.

DISCUSSION QUESTION:

Even with a wholesale migration to SDN there would still be complex control plane software needed to program the data plane. Surely, there would be improvements in this software over time that would result in multiple versions of this software. Why, then, would this represent less complexity than the current paradigm?

2.2.2 CLOSED ENVIRONMENTS ENCOURAGE VENDOR LOCK-IN

It is true that over the years standards have been developed in the networking space for most relevant protocols and data that are used by switches and routers. For the most part, vendors do their best to implement these standards in a manner that allows heterogeneous networks of devices from multiple vendors to co-exist with one another.

However, in spite of good intentions by vendors, enhancements are often added to these standard implementations, which attempt to allow a vendor’s product to outperform its competition. With many vendors adding such enhancements, the end result is that each vendor product will have difficulty interoperating smoothly with products from another vendor. Adherence to standards helps alleviate the issues associated with attempting to support multiple vendor types in a network, but problems with interoperability and management often far outweigh the advantages that might be gained by choosing another vendor. As a result, customers frequently become effectively married to a vendor they chose years or even decades before. This sort of vendor lock-in alleviates downward pressure on cost as the vendor is largely safe from competition and can thus preserve high profit margins.

2.2.3 COMPLEXITY AND RESISTANCE TO CHANGE

Quite often in networking we arrive at a point of having made the network operational and the normal impulse from that point on is to just leave things as they are, to not disturb it lest it break and we must start all over again. Others may have been burned by believing the latest vendor who proposed a new solution and, when the dust settled, their closed, proprietary, vendor-specific solution was just as complex as that of the previous vendor.

Unfortunately, in spite of efforts at standardization, there is still a strong argument to stay with that single-vendor solution. Often, that closed, complex solution may be easier to deploy precisely because there is only one vendor involved, and that vendor's accountability is not diluted in any way. By adopting and embracing a solution that works, we believe we lower our short-term risk. That resistance to change results in long-term technological stagnation and sluggishness. The ideal would be a simpler, more progressive world of networking, with open, efficient, and less expensive networking devices. This is a goal of SDN.

2.2.4 INCREASED COST OF OPERATING THE NETWORK

As networks become ever-larger and more complex, the *Operational Expense* (OPEX) of the network grows. This component of the overall costs is increasingly seen to be more significant than the corresponding *Capital Expense* (CAPEX) component. SDN has the capacity to accelerate the automation of network management tasks in a multivendor environment [6,7]. This, combined with the fact that SDN will permit faster provisioning of new services and provides the agility to switch equipment between different services [8] should lead to lower OPEX with SDN. In Section 15.3.6, we will examine proposals where SDN may be used in the future to reduce the power consumption of the networking equipment in a data center, which is another major contributor to network OPEX.

2.3 SDN IMPLICATIONS FOR RESEARCH AND INNOVATION

Networking vendors have enjoyed an enviable position for over two decades. They control networking devices from the bottom up: the hardware, the low level firmware, and the software required to produce an intelligent networking device. This platform on which the software runs is closed, and, consequently, only the networking vendors themselves can write the software for their own networking devices.

The reader should contrast this to the world of software and computing, where you can have many different hardware platforms created by multiple vendors, consisting of different and proprietary capabilities. Above that hardware reside multiple layers of software, which ultimately provide a common and open interface to the application layer. The *Java Virtual Machine* and the *Netbeans Integrated Development Environment* provide a good example of cross-platform development methods. Using such tools, one can develop software that may be ported between Windows PC, Linux, or an Apple MAC. Analogously, tablets and smartphones, such as iPhones or Android-based phones, can share application software and can run on different platforms with relative ease. Imagine if only Apple were able to write software for Apple products and only Microsoft was able to write software to run on

PCs or Windows-based servers? Would the technological advances we have enjoyed in the last decade have taken place? Probably not.

In [9] the authors explain that this status quo has negatively impacted innovation in networking. In the next section, we examine this relationship and, how, on the contrary, the emergence of SDN is likely to accelerate such innovation.

2.3.1 STATUS QUO BENEFITS INCUMBENT VENDORS

The collective monopolies extant in the networking world today are advantageous to networking vendors. Their only legitimate competition comes from their fellow established NEMs. Although, periodically, new networking companies emerge to challenge the status quo, that is the exception rather than the rule. The result is that the competitive landscape evolves at a much slower pace than it would in a truly open environment. This is due to limited incentives to invest large amounts of money when the current status quo is generating reasonable profits, due primarily to the lack of real competition and the resulting high margins they are able to charge for their products.

Without a more competitive environment, the market will naturally stagnate to some degree. The incumbent NEMs will continue to behave as they have in the past. The small players will struggle to survive, attempting to chip away at the industry giants, but with limited success, especially since the profit margins of those giants are so large. This competition-poor environment is unhealthy for the market and for the consumers of its products and services. Consider the difference in profit margins for a server vendor versus that for a networking vendor. Servers are sold at margins close to five percent or below. Networking device margins can be as high as thirty percent or more for the established vendors.

2.3.2 SDN PROMOTES RESEARCH AND INNOVATION

Universities and research labs are focal points of innovation. In technology, innovations by academia and other research organizations have accelerated the rate of change in numerous industries. Open software environments such as Linux have helped to promote this rapid pace of advancement. For instance, if researchers are working in the area of operating systems, they can look at Linux and modify its behavior. If they are working in the area of server virtualization or databases, they can look at KVM or Xen and MySQL or Postgres. All of these open-source packages are used in large-scale commercial deployments. There is no equivalent in networking today. Unfortunately, the current closed nature of networking software, network protocols, network security, and network virtualization is such that it has been challenging to experiment, test, research, and innovate in these areas. This, in fact, is one of the primary drivers of SDN [4]. In that case, a number of universities collaborated to propose a new standard for networking called OpenFlow, which would allow for this free and open research to take place. This makes one wonder if SDN will ultimately be to the world of networking what Linux has become to the world of computing.

General innovation, whether from academia or by entrepreneurs, is stifled by the closed nature of networking devices today. How can a creative researcher or entrepreneur develop a novel mechanism for forwarding traffic through the Internet? That would be nearly impossible today. Is it reasonable for a start-up to produce a new way of providing hospitality-based networking access in airports, coffee shops, and malls? Perhaps, but it would be required to run across network vendor equipment such as *wireless access points* (APs) and access switches and routers, that are themselves closed systems. The more that the software and hardware components of networking are commoditized, the lower their cost to customers. SDN promises both the hardware commoditization as well as openness, and both of these factors contribute to innovation.

To be fair, though, one must keep in mind that innovation is also driven by the prospect of generating wealth. It would be naive to imagine that a world of low-cost networking products will have a purely positive impact on the pace of innovation. For some companies, the lower product margins presaged by SDN will reduce their willingness to invest in innovation. We will examine this correlation and other business ramifications of SDN in Chapter 14.

2.4 DATA CENTER INNOVATION

In Section 1.3 we explained that in the last few years, server virtualization has caused both the capacity and the efficiency of data centers to increase exponentially. This unbounded growth has made possible new computing trends such as the cloud, which is capable of holding massive amounts of computing power and storage capacity. The whole landscape of computing has changed as a result of these technological advances in the areas of compute and storage virtualization.

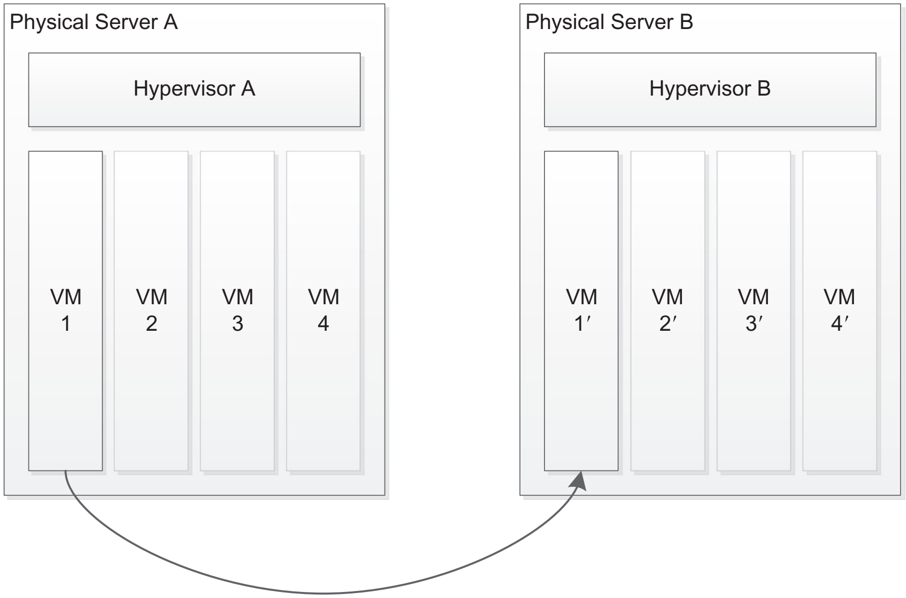
2.4.1 COMPUTE AND STORAGE VIRTUALIZATION

Virtualization technology has been around for decades. The first commercially available VM technology for IBM mainframes was released in 1972 [10], complete with the *hypervisor* and the ability to abstract the hardware below, allowing multiple heterogeneous instances of other operating systems to run above it in their own space. In 1998 VMware was established and began to deliver software for virtualizing desktops as well as servers.

Use of this *compute virtualization* technology did not explode until data centers became prevalent, and the need to dynamically create and tear down servers, as well as moving them from one physical server to another, became important. Once this occurred, however, the state of data center operations immediately changed. Servers could be instantiated with a mouse click, and could be moved without significantly disrupting the operation of the server being moved.

Creating a new VM, or moving a VM from one physical server to another, is straightforward from a server administrator's perspective, and may be accomplished very rapidly. Virtualization software, such as VMware, Hyper-V, KVM, and XenServer, are examples of products that allow server administrators to readily create and move virtual machines. This has reduced the time needed to start up a new instance of a server to a matter of minutes or even seconds. Fig. 2.3 shows the simple creation of a new instance of a virtual machine on a different physical server.

Likewise, *storage virtualization* has existed for quite some time, as has the concept of abstracting storage blocks and allowing them to be separated from the actual physical storage hardware. As with servers, this achieves efficiency in terms of speed (e.g., moving frequently used data to a faster device), as well as in terms of utilization (e.g., allowing multiple servers to share the same physical storage device).



Create another instance of VM1 on Physical Server B: **Elapsed time = MINUTES**

FIG. 2.3

Server virtualization: creating new VM instance.

These technological advancements allow servers and storage to be manipulated quickly and efficiently. While these advances in computer and storage virtualization have been taking place, the same has not been true in the networking domain [11].

2.4.2 INADEQUACIES IN NETWORKS TODAY

In Chapter 1 we discussed the evolution of networks which allowed them to survive catastrophic events such as outages and hardware or software failures. In large part, networks and networking devices have been designed to overcome these rare but severe challenges. However, with the advent of data centers, there is a growing need for networks to not only recover from these types of events, but also to be able to respond quickly to frequent and immediate changes.

While the tasks of creating a new network, moving a new network, and removing a network are similar to those performed for servers and storage, doing so requires work orders, coordination between server and networking administrators, physical or logical coordination of links, *Network Interface Cards* (NICs), and ToR switches, to name a few. These time-consuming tasks are reflected in Fig. 2.4, which illustrates the difference in elapsed time between creating a new instance of a VM, which is in the order of minutes, compared to the multiple days that it may take to create a new instance of a network. This disparity is due to the fact that the servers are virtualized yet the network is still purely a physical network. Even when we are configuring something virtual for the network, such as a VLAN, changes

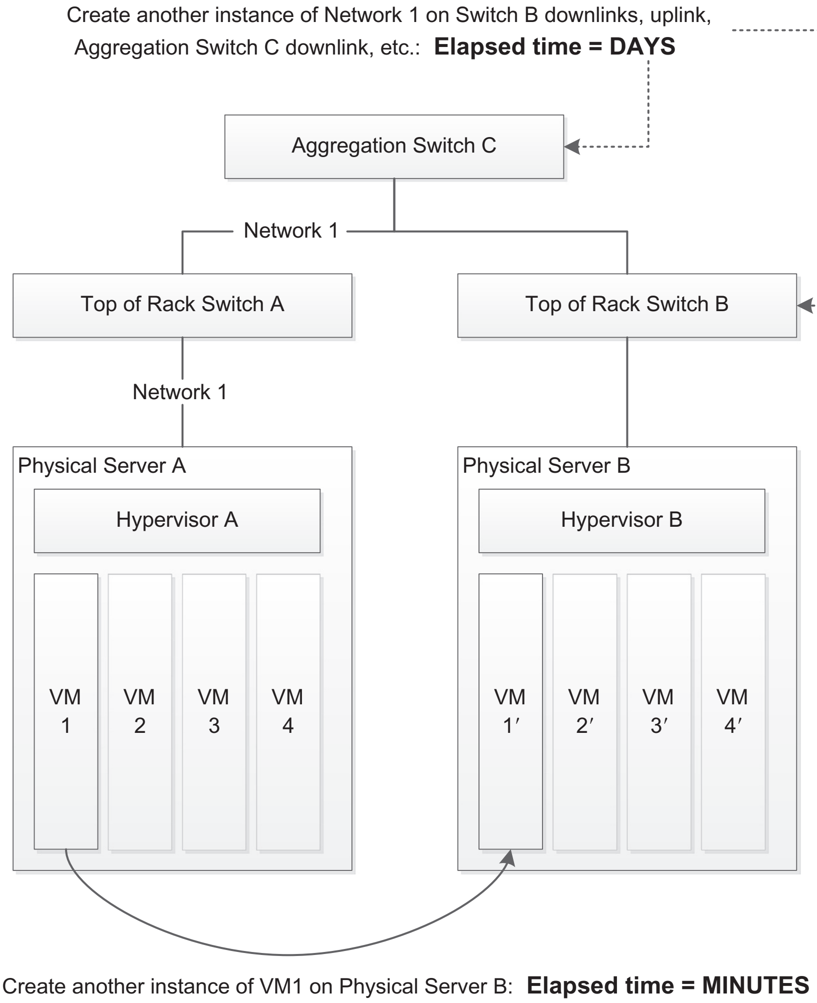

Create another instance of Network 1 on Switch B downlinks, uplink,
Aggregation Switch C downlink, etc.: Elapsed time = DAYS
Aggregation Switch C
Network 1
Top of Rack Switch A
Top of Rack Switch B
Network 1
Physical Server A
Hypervisor A
VM 1
VM 2
VM 3
VM 4
Physical Server B
Hypervisor B
VM 1′
VM 2′
VM 3′
VM 4′
Create another instance of VM1 on Physical Server B: Elapsed time = MINUTES

FIG. 2.4

Creating a new network instance in the old paradigm.

are more cumbersome than in their server counterparts. In Chapter 1 we explained that while the control plane of legacy networks had sophisticated ways of autonomously and dynamically distributing layer two and layer three state, no corresponding protocols exist for distributing the policies that are used in policy-based routing. Thus, configuring security policy, such as ACLs or virtualization policy, such as to which VLAN a host belongs, remains static and manual in traditional networks. Thus, the task of reconfiguring a network in a modern data center does not take minutes, but, rather, days. Such inflexible networks are hindering IT administrators in their attempts to automate and streamline their virtualized data center environments. SDN holds the promise that the time required for such network reconfiguration be reduced to the order of minutes, such as is already the case for reconfiguration of VMs.

DISCUSSION QUESTION:

Discuss why in Fig. 2.4, creating an instance of network 1 on switch B may take days yet creating another instance of VM1 need only take minutes.

2.5 DATA CENTER NEEDS

The explosion of the size and speed of data centers has strained the capabilities of traditional networking technologies. We discuss these needs briefly below and cover them in greater detail in Chapter 8. The sections below serve as an indication of new requirements emerging from the technological advances taking place now in data center environments.

2.5.1 AUTOMATION

Automation allows networks to come and go at will, following the movements of servers and storage as needs change. This characteristic is sometimes referred to as *agility*, the ability to dynamically instantiate networks and to disable them when they are no longer needed. This must happen fast, efficiently, and with a minimum of human intervention. Not only do networks come and go, but they also tend to expand and contract. Supporting such agility is only possible through automation.

2.5.2 SCALABILITY

With data centers and cloud environments, the sheer number of end stations that connect to a single network has grown exponentially. The limitations of MAC address table sizes and number of VLANs have become impediments to network installations and deployments. The large number of physical devices present in the data centers also poses a *broadcast control* problem. The use of tunnels and virtual networks can contain the number of devices in a broadcast domain to a reasonable number.

2.5.3 MULTIPATHING

Accompanying the large demands placed on the network by the scalability requirement stated above, is the need for the network to be efficient and reliable. That is, the network must make optimal use of its resources, and it must be resistant to failures of any kind; and, if failures do occur, the network must be able to recover immediately. Fig. 2.5 shows a simple example of multipath through a network, both for choosing the shortest path as well as for alternate or redundant paths. Legacy layer two control plane software would block some of the alternate and redundant paths shown in the figure in order to eliminate forwarding loops. Because we are living with network technology invented years ago, the network is forced into a hierarchy, which results in links which could have provided shortest-path routes between nodes lying entirely unused and dormant. In cases of failure, the current hierarchy can reconfigure

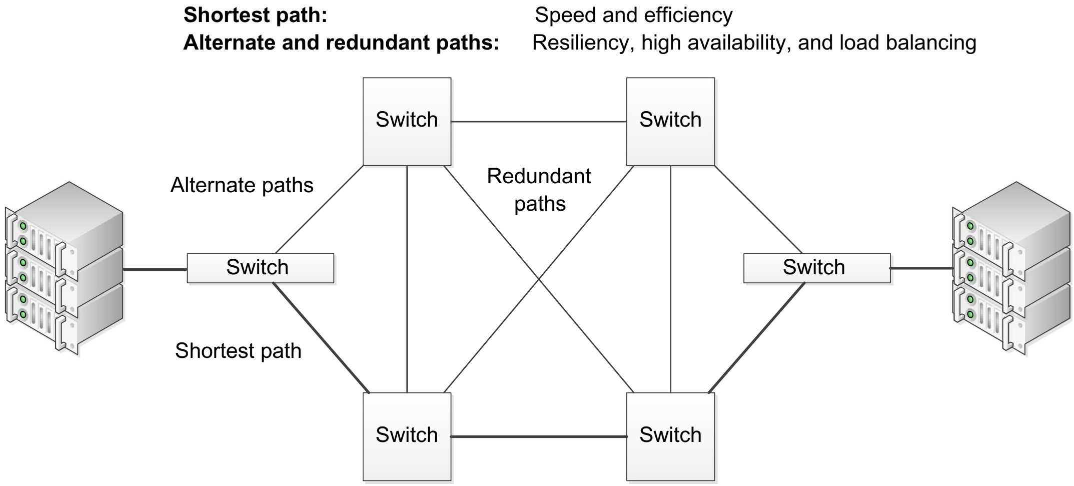

Shortest path: Speed and efficiency
Alternate and redundant paths: Resiliency, high availability, and load balancing
Switch
Switch
Alternate paths
Redundant paths
Switch
Switch
Shortest path
Switch
Switch

FIG. 2.5

Multipath.

itself in a nondeterministic manner and with unacceptable latency. The speed and high-availability requirements of the modern data center mandate that multiple paths not be wasted by being blocked and, instead, be put into use to improve efficiency as well as to achieve resiliency and load-balancing.

2.5.4 MULTITENANCY

With the advances in data center technology described above and the subsequent advent of *cloud computing*, the idea of hosting dozens, or even hundreds or thousands of customers or *tenants* in the same physical data center has become a requirement. One set of physical hardware hosting multiple tenants has been feasible for some time in the server and storage area. Multitenancy implies that the data center has to provide each of its multiple tenants with their own (virtual) network that they can manage in a manner similar to the way that they would manage a physical network.

2.5.5 NETWORK VIRTUALIZATION

The urgency for automation, multitenancy, and multipathing has increased as a result of the scale and fluidity introduced by server and storage virtualization. The general idea of virtualization is that you create a higher-level abstraction that runs on top of the actual physical entity you are abstracting. The growth of compute and storage server virtualization has created demand for network virtualization. This means having a virtual abstraction of a network running on top of the actual physical network. With virtualization the network administrator should be able to create a network anytime and anywhere (s)he chooses, as well as expand and contract networks that are already in existence. Intelligent virtualization software should be capable of this, without requiring the upper virtualized layer to be aware of what is occurring at the physical layer.

Server virtualization has caused the scale of networks to increase as well, and this increased scale has put pressure on layer two and layer three networks as they exist today. Some of these pressures can be alleviated to some degree by tunnels and other type of technologies, but fundamental network issues remain, even in those situations. Consequently, the degree of network virtualization required to keep pace with data center expansion and innovation is not possible with the network technology that is available today.

To summarize, advances in data center technology have caused weaknesses in the current networking technology to become more apparent. This situation has spurred demand for better ways to construct and manage networks [12], and that demand has driven innovation around SDN [13].

DISCUSSION QUESTION:

Explain how network virtualization can support multitenancy.

2.6 CONCLUSION

The issues of reducing cost and the speed of innovation as motivators for SDN will be recurring themes in the balance of this work. The needs of the modern data center are so tightly linked with the demand for SDN that we dedicate all of Chapter 8 to the examination of use cases in the data center that benefit from SDN technology. These needs did not appear overnight nor did SDN simply explode onto the networking scene in 2009, however. There were a number of tentative steps over many years that formed the basis for what ultimately appeared as the SDN revolution. In the next chapter we will review this evolution and examine how it culminated in the birth of what we now call SDN. We will also discuss the context in which SDN has matured and the organizations and forces that continue to mold its future.

REFERENCES

[1] IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges; IEEE 802.1D-2004, New York, June 2004.

[2] Shenker S. The future of networking, and the past of protocols. Open Networking Summit, October 2011. Stanford University; 2011.

[3] Kim H, Feamster N. Improving network management with software defined networking. IEEE Commun Mag 2013;51(2).

[4] McKeown N, Parulkar G, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev Arch 2005;35(3).

[5] Kirkpatrick K. Software-defined networking. Commun ACM 2013;56(9).

[6] Malim G. SDN's value is in operational efficiency, not capex control. Global Telecoms Business; 2013. Retrieved from: http://www.globaltelecomsbusiness.com/article/3225124/SDNs-value-is-in-operational-efficiency-not-capex-control.html.

[7] Wilson C. NTT reaping opex rewards of SDN; 2013. Retrieved from: http://www.lightreading.com/carrier-sdn/ntt-reaping-opex-rewards-of-sdn/d/d-id/705306.

[8] Yegulalp S. Five SDN benefits enterprises should consider. Network Computing; 2013. Retrieved from: http://www.networkcomputing.com/next-generation-data-center/commentary/networking/five-sdn-benefits-enterprises-should-con/240158206.

[9] Greenberg A, Hjalmtysson G, Maltz D, Myers A, Rexford J, Xie G, et al. A clean slate 4D approach to network control and management. ACM SIGCOMM Comput Commun Rev 2005;35(3).

[10] Witner B, Wade B. Basics of z/VM virtualization. IBM. Retrieved from: http://www.vm.ibm.com/devpages/bkw/vmbasics.pdf.

[11] Bari MF, Boutaba R, et al. Data center network virtualization: a survey. IEEE Commun Surv Tutorials 2013;15(2).

[12] Narten T, Sridharan M, et al. Problem statement: overlays for network virtualization. Internet Draft. Internet Engineering Task Force; 2013.

[13] Brodkin J. Data center startups emerging to solve virtualization and cloud problems. Network World; 2011. Retrieved from: http://www.networkworld.com/news/2011/061411-data-center-startups.html.