CHAPTER

SDN IN THE DATA CENTER

8

More than most mainstream deployments, the modern data center has stretched traditional networking to the breaking point. Previous chapters in this book have highlighted the importance of data center technology in accelerating the urgency and relevance of SDN. In its various incarnations discussed thus far, SDN has evolved specifically to address the limitations of traditional networking that have presented themselves most dramatically in today's mega-data center. In this chapter we delve more deeply into data center networking technology, both as it exists today and as it will adapt in the future through the benefit of SDN. Specifically, in this chapter we examine:

- **Data Center Needs**. What are the specific shortcomings that exist in data center networks today?
- **Data Center Technologies**. What technologies are employed in the data center, both now and with the advent of SDN?
- **Data Center Use Cases**. What are some specific applications of SDN in the data center?

In answering these questions, we hope to illuminate the close relationship that exists between SDN's technical strengths and the modern data center.

8.1 DATA CENTER DEFINITION

Let us define what exactly is a data center in order to better understand its needs and the technologies that have arisen to address those needs. The idea of a data center is not new. Densely packed racks of high-powered computers and storage have been around for decades, originally providing environments for mainframes. These eventually gave way to minicomputers and then to the servers that we know and deal with today. Over time the density of those servers and the arrays of storage that served them has made it possible to host a tremendous amount of compute power in a single room.

Today's data centers hold thousands, even tens of thousands, of physical servers. These data centers can be segregated into the following three categories:

- **Private Single-Tenant**. Individual organizations which maintain their own data centers belong in this category. The data center is for the private use of the organization and there is only the one organization or *tenant* using the data center.
- **Private Multitenant**. Organizations which provide *specialized* data center services on behalf of other client organizations belong in this category. IBM and EDS (now HP) are examples of companies which host such data centers. They are built and maintained by the organization providing the service and there are multiple clients whose data is stored there, suggesting

the term *multitenant*. These data centers are private because they offer their services contractually to specific clients.

- **Public Multitenant**. Organizations which provide *generalized* data center services to any individual or organization belong in this category. Examples of companies which provide these services include Google and Amazon. These data centers offer their services to the public. Anybody who wishes to use them may access them via the world-wide web, be they individuals or large organizations.

In the past, these data centers were often hosted such that they could be reached only through private communication channels. However, in recent years, these data centers have begun to be designed to be accessible through the Internet. These types of data centers are often referred to as residing in *the cloud*. Three subcategories of clouds are in common use, *public* cloud, *private* cloud, and *hybrid* cloud. In a public cloud, a service provider or other large entity makes services available to the general public over the Internet. Such services may include a wide variety of applications or storage services, among other things. Examples of a public cloud offering include *Microsoft Azure Services Platform* and *Amazon Elastic Compute Cloud*. In a private cloud, a set of server and network resources is assigned to one tenant exclusively and protected behind a firewall specific to that tenant. The physical resources of the cloud are owned and maintained by the cloud provider, which may be a distinct entity from the tenant. The physical infrastructure may be managed using a product such as VMware's vCloud. Amazon Web Services is an example of how a private cloud may also be hosted by a third party (i.e., Amazon). An increasingly popular model is the hybrid cloud, where part of the cloud runs on resources dedicated to a single tenant but other parts reside on resources that are shared with other tenants. This is particularly beneficial when the shared resources may be acquired and released dynamically as demand grows and declines. Verizon's *cloud bursting* that we present in Section 9.2.3 provides an example of how a hybrid cloud might work. Note that the ability to dynamically assign resources between tenants and to the general public is a major driver behind the interest in network virtualization and SDN.

Combining the increasing density of servers and storage and the rise in networking speed and bandwidth, the trend has been to host more and more information in ever-larger data centers. Add to that the desire of enterprises to reduce operational costs and we find that merging many data centers into a larger single data center is the natural result.

Concomitant with this increased physical density is a movement toward virtualization. Physical servers now commonly run virtualization software which allows a single server to actually run a number of virtual machines. This move toward virtualization in the compute space reaps a number of benefits, from decreasing power requirements to more efficient use of hardware and the ability to quickly create, remove, and grow or shrink applications and services.

One of the side effects of the migration to cloud computing is that it is usually associated with a usage-sensitive payment system. The three main business models of cloud computing are *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and *Software as a Service* (SaaS). These three are all provided and sold on an on-demand, usage-sensitive basis, shifting CAPEX costs to OPEX. The popularity of this new business model has been a major driver for the explosion in the size and number of data centers. We discuss the business ramifications of this new model on SDN in Chapter 14.

Thus, the three trends toward cloud, increased density and virtualization have placed demands on data center networking that did not exist before. We examine those needs in the next section.

8.2 DATA CENTER DEMANDS

As a result of the evolutionary trends identified in the previous section, there have arisen a number of critical networking needs that must be met in order for data center technology to thrive. We have mentioned some of these needs in previous chapters. We examine them in greater detail here.

8.2.1 OVERCOMING CURRENT NETWORK LIMITATIONS

The potential to easily ignite and tear down VMs is a radical departure from the physical world that network managers have traditionally faced. Networking protocols were coupled with physical ports which is *not* the case moving forward. The dynamic nature and sheer number of VMs in the data center have placed demands on the capacity of network components that were earlier thought to be safe from such pressures. In particular, these areas include *MAC Address Table Size*, *Number of VLANs*, and *Spanning Tree*.

MAC address explosion

In switches and routers, the MAC Address Table is used by the device to quickly determine the port or interface out of which the device should forward the packet. For speed, this table is implemented in hardware. As such, it has a physical limit to its size. Naturally, device vendors will determine the maximum number of entries to hold in the MAC table based on controlling their own costs, while at the same time providing an adequate number of entries for the demands of the network. If the table is too large, the vendor will have spent too much money on creating the device. If the table is too small, then the customer will experience the problems we describe below.

Networks in the past had manageable limits on the maximum number of MAC addresses that would need to be in the MAC Address Table at any given time. This was partly attributed to physical limitations of data centers and the number of servers that would be part of a single layer two network. It is also affected by the physical layout of the network. In the past, physically separate layer two networks remained logically separate. Now, with network virtualization and the use of Ethernet technology across WANs, the layer two networks are being stretched geographically as never before. With server virtualization, the number of servers possible in a single layer two network has increased dramatically. With numerous virtual NICs on each virtual server, this problem of a skyrocketing number of MAC addresses is exacerbated. Layer two switches are designed to handle the case of a MAC table miss by flooding the frame out all ports except the one on which it arrived, as shown in Fig. 8.1. This has the benefit of ensuring that the frame reaches its destination, if that destination exists on the layer two network. Under normal circumstances, when the destination receives that initial frame, this will prompt a response from the destination. Upon receipt of the response, the switch is able to learn the port on which that MAC address is located and populates its MAC table accordingly. This works well unless the MAC table is full, in which case it cannot learn the address. Thus, frames sent to that destination continue to be flooded. This is a very inefficient use of bandwidth and can have significant negative performance impact. This problem is exacerbated in the core of the data center network. When the traditional nonvirtualized network design is used, where all the MAC addresses are visible throughout the topology, the pressure on the MAC address tables is intense.

Since VLANs are used extensively in modern layer two networks, let us take a closer VLAN-centric look at Fig. 8.1. When a VLAN-tagged packet fails its match it is flooded out only to all ports on that

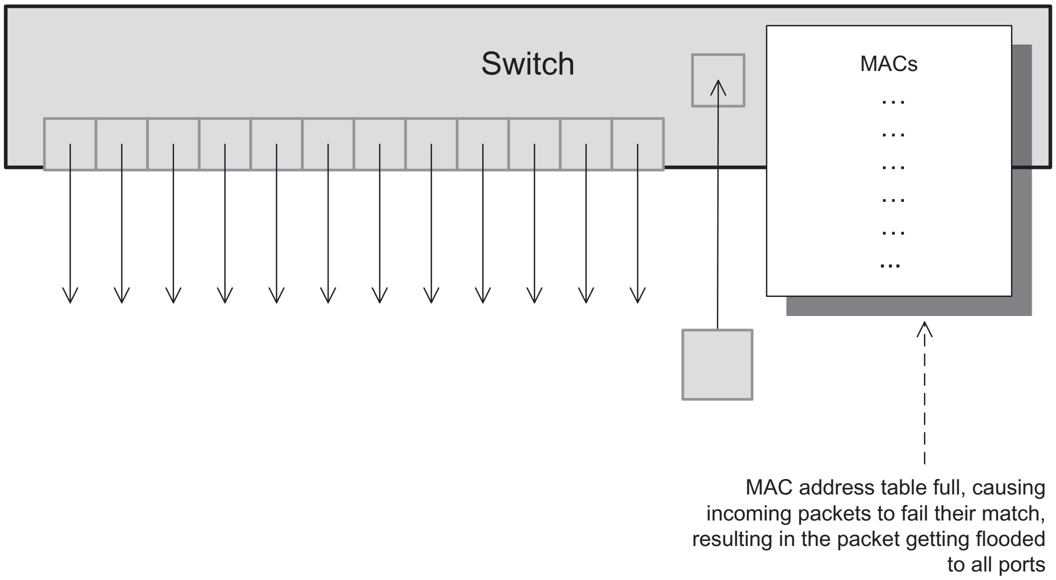
Switch
MACs
...
MAC address table full, causing incoming packets to fail their match, resulting in the packet getting flooded to all ports

FIG. 8.1

MAC address table overflow.

VLAN, somewhat mitigating the flooding inefficiency. Conversely, a host may be a member of multiple VLANs, in which case it will occupy one MAC address table entry per VLAN, further exacerbating the problem of MAC address table overflow.

Number of VLANs

When the IEEE 802.1 working group created the 802.1Q extension to the definition of local area networks, they did not anticipate that there would be a need for more than twelve bits to hold potential VLAN IDs. The IEEE 802.1Q Tag for VLANs is shown in Fig. 8.2. The tag depicted in Fig. 8.2 supports $2^{12} - 2$ (4094) VLANs (the subtraction of two is due to the fact that all zeros and all ones are reserved). When these tags were introduced in the mid-to-late 1990s, networks were smaller and there was very limited need for multiple virtual domains within a single physical network.

The introduction of data centers created the need to segregate traffic so as to ensure security and separation of the various tenant networks' traffic. The technology responsible for providing this separation is VLANs. As long as data centers remained single-tenant networks, the maximum number of 4094 VLANs seemed more than sufficient. To have expanded this twelve-bit field unnecessarily was not wise, as different tables in memory and in the ASICs had to be large enough to accommodate the maximum number. Thus, to have made the maximum larger than 4094 had a definite downside at the time the design work was performed.

When data centers continued to expand, however, especially with multitenancy and server virtualization, that number of VLANs required could easily exceed 4094. When there are no more available VLANs, the ability to share the physical resources amongst the tenants quickly becomes complex.

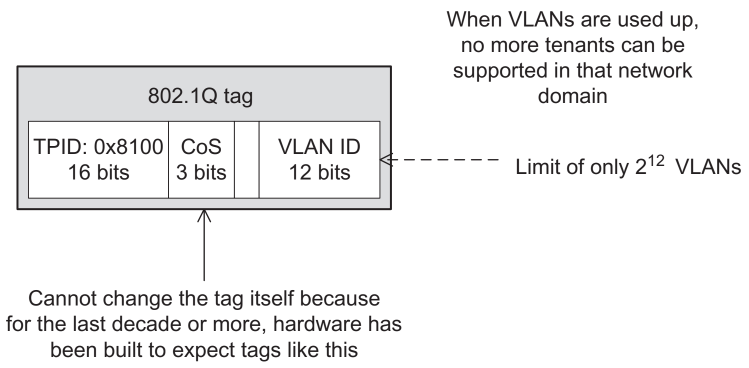
When VLANs are used up, no more tenants can be supported in that network domain
802.1Q tag
TPID: 0x8100 16 bits
CoS 3 bits
VLAN ID 12 bits
Limit of only 2^{12} VLANs
Cannot change the tag itself because for the last decade or more, hardware has been built to expect tags like this

FIG. 8.2

VLAN exhaustion.

Since the number of bits allocated to hold the VLAN is fixed in size, and hardware has been built for years which depends on that specific size, it is nontrivial to expand the size of this field to accommodate more VLANs. Thus, some other solution is needed to overcome this limitation.

An upshot of the limit of 4094 VLANs has been an increase in the use of MPLS. MPLS does not suffer the same limitation in the number of MPLS tags as exists with VLAN IDs, and they need not be isolated to a single layer two network. Precisely because it is a layer three technology, with the correspondingly more complex control plane, MPLS has primarily been deployed in the WAN. It is likely, though, that MPLS will see more use in the data center [1]. One example of this is Juniper's Contrail product, which we presented in Section 6.2.5. We explore the use case of MPLS with SDN in the WAN in Section 9.1.1.

Spanning tree

In the early years of Ethernet, bridges were built as *transparent* devices capable of forwarding packets from one segment to another without explicit configuration of forwarding tables by an operator. Forwarding tables were learned by the bridges by observing the traffic being forwarded through them. Distributed intelligence in the devices was capable of collectively determining if there were loops in the network and truncating those loops so as to prohibit issues such as broadcast storms from bringing down the network.

This was accomplished by the bridges or switches collectively communicating with one another to create a *spanning tree* which enforces a loop-free hierarchical structure on the network in situations where physical loops do exist. This spanning tree was then calculated using the *Spanning Tree Protocol* (STP) we briefly reviewed in Section 1.5.1. The process of determining this spanning tree is called convergence, and in the early days it would take some time (dozens of seconds) for the spanning tree to converge after a physical change to the networking devices and their links had taken place.

Over time, through improvements to STP, the process of converging the spanning tree increased in speed. For example, with the improvements in recent versions of the standard, convergence times for conventionally large networks have decreased dramatically.[1] Despite these improvements, STP still leaves perfectly functional links unused and with frames being forwarded to the root of the spanning tree, which is certainly not universally the optimal path. Data centers need more *cross-sectional* bandwidth. By this we mean using the most efficient path between any two nodes without imposing an artificial hierarchy in the traffic patterns.

The fact that the fluidity of data center virtualization has increased the frequency of changes and disruptions, thus requiring re-convergence to occur more often, has only added to the inefficiency of STP in the data center. STP was simply not built for the modern data center world of virtual switches and ephemeral MAC addresses.

DISCUSSION QUESTION:

We've listed three major network limitations in the data center. Can you think of others that would cause data center administrators to look for answers via SDN or other technologies?

8.2.2 ADDING, MOVING, AND DELETING RESOURCES

Today's virtualized data centers are able to make changes much quicker than in the past. Networks need to adapt in order to keep pace with the virtualization capabilities of servers and storage. Speed and automation are of critical importance when it comes to handling the rapid changes demanded by virtualized servers and storage.

It is clearly important that changes to networking infrastructure take place at the same rate as is possible with servers and storage. Legacy networks require days or weeks for significant changes to VLANs and other network plumbing needs. A large part of the reason for this is that the repercussions of a mistaken network change can easily impact all the data center resources, including not only networking but also its compute and storage components. Coordinating changes with all of these disparate departments is unavoidably complex. These changes need to have the ability to be automated so that changes that must happen immediately can take place without human intervention. Legacy protocols were designed to react *after* the newly ignited service comes online. With SDN one can use the foreknowledge that a new service is about to be initiated and proactively allocate the network capacity it will require.

8.2.3 FAILURE RECOVERY

The size and scale of data centers today makes recovery from failure a complex task, and the ramifications of poor recovery decisions are only magnified as scale grows. Determinism, predictability, and optimal reconfiguration are among the most important considerations related to this area.

With the distributed intelligence of today's networks, recovery from failures may result in unpredictable behavior. It is desirable that the network move to a known state given a particular failure.

[1]In practice, including link failure detection time, convergence times have been reduced to the order of hundreds of milliseconds.

Distributed protocols render this difficult. It requires a complete view of the network in order to make the recovery process yield the best result.

8.2.4 MULTITENANCY

Data center consolidation has resulted in more and more clients occupying the same set of servers, storage and network. The challenge is to keep those individual clients separated and insulated from each other and to utilize network bandwidth efficiently.

In a large multitenant environment, it is necessary to keep separate the resources belonging to each client. For servers this may mean not mixing clients' virtual machines on the same physical server. For networks, this may mean segregation of traffic using a technology which ensures that packets from two distinct tenants remain insulated from one another. This is needed not only for the obvious security reasons, but also for QoS and other service guarantees.

With a small number of tenants in a data center, it was possible to arrange traffic and bandwidth allocation using scripts or some other rudimentary techniques. For example, if it was known that backups would be run at a certain time, arrangements could be made to route traffic appropriately around that heavy load. However, in an environment with hundreds (large data center) or thousands (cloud) of tenants with varying needs and schedules, a more fluid and robust mechanism is required to manage traffic loads. This topic is examined in more detail in the next section.

8.2.5 TRAFFIC ENGINEERING AND PATH EFFICIENCY

As data centers have grown, so has the need to make better use of the resources being shared by all of the applications and processes using the physical infrastructure. In the past this was less of an issue because overprovisioning could make up for inefficiencies. But with the current scale of data centers, it is imperative to optimally utilize the capacity of the network. This entails proper use of monitoring and measurement tools for more informed calculation of the paths that network traffic takes as it makes its way through the physical network.

In order to understand traffic loads and take the appropriate action, the traffic data must be monitored and measured. Gathering traffic intelligence has often been a luxury for networks. But with the need to make the most efficient use of the links and bandwidth available, this task has become an imperative.

State-of-the-art methods of calculating routes use link-state technology, which provides the network topology for the surrounding network devices which are part of the same Autonomous System (AS). This allows path computation to determine the *shortest path*. However, the shortest path as defined by traditional technology is not always the optimal path, since it does not take into consideration dynamic factors such as traffic load.

One of the reasons for the increasing attention on traffic engineering and path efficiency in the data center has been the rise of *East-West* traffic relative to *North-South* traffic. In Section 1.3 we defined these traffic types as follows: *East-West traffic is composed of packets sent by one host in a data center to another host in that same data center. Analogously,* North-South *traffic is traffic entering (leaving) the data center from (to) the outside world.*

Facebook provides a good example of what is driving the growth in East-West traffic. When you bring up your newsfeed page on Facebook, it has to pull in a multitude of data about different people and events in order to build the webpage and send it to you. That data resides throughout the data center.

The server that is building and sending you your Facebook newsfeed page has to pull data from servers in other parts of the data center. That East-West traffic is at least as large as the North-South data (i.e., the final page it sends you) and in fact can be larger if you account for the fact that they are moving data around even when no one is requesting it (i.e., replication, staging data in different data centers, etc.).

In the older data center model, when most traffic was North-South, traffic engineering inside the data center was mostly a question of ensuring that congestion and loss would not occur as the data moved from the servers back toward the outside world. With so much East-West traffic, traffic engineering in the data center has become much more complex. These new traffic patterns are a large part of the reason for the interest in the *Ethernet fabrics* we discuss in Section 8.5. An example of this problem is how to distribute East-West elephant flows across multiple paths in the data center. For instance, if internal servers have 10Gbps uplinks and the data center has a 40Gbps core with multiple equal cost paths, it is very likely that without explicit traffic engineering the full-speed 10Gbps flows will experience congestion at some point(s) in the core.

We have now described some of the most critical needs that exist in data center networks as a result of server virtualization and the massive scale that occurs due to data center consolidation and the advent of the cloud. In response to these needs, the networking industry has developed new technologies to mitigate some of these issues. Some of these technologies are directly related to SDN and some are not. We examine a number of these technologies in the sections that follow.

DISCUSSION QUESTION:

We have listed adds, moves, deletes, failure recovery, multitenancy, and traffic engineering as issues in data centers today. Which of these do you think has been magnified by the emergence of the cloud? Which of these is critical because of virtualization of servers and storage in data centers? Which of these is critical because of applications such as video and voice?

8.3 TUNNELING TECHNOLOGIES FOR THE DATA CENTER

Previously in this chapter we have noted the impact of server virtualization. One of the responses to server virtualization has been the birth of the concept of *network virtualization*. We defined network virtualization in Section 3.7. We also discussed one of the alternative definitions of SDN, which we referred to as *SDN via Hypervisor-Based Overlays*. At that time we showed how hypervisor-based tunneling technologies are employed to achieve this virtualization of the network. There are a number of tunneling technologies used in the data center, and some of these data center-oriented tunneling technologies predate SDN. There are three main technologies being used today to address many of the data center needs we presented in Section 8.2. These tunneling methods are *Virtual eXtensible Local Area Network* (VXLAN) [2], *Network Virtualization using Generic Routing Encapsulation* (NVGRE) [3], and *Stateless Transport Tunneling* (STT) [4].

All of these tunneling protocols are based on the notion of encapsulating an entire layer two MAC frame inside an IP packet. This is known as *MAC-in-IP tunneling*. The hosts involved in communicating with each other are unaware that there is anything other than a traditional physical network between them. The hosts construct and send packets in exactly the same manner as they would had there been no network virtualization involved. In this way, network virtualization resembles server virtualization, where hosts are unaware that they are actually running in a virtual machine environment. Admittedly, the fact that the packets are encapsulated as they transit the physical network does reduce the *maximum transmission unit* (MTU) size, and the host software must be reconfigured to account for that detail.

All three of these tunneling methods may be used to form the tunnels that knit together the *Virtual Tunnel End Points* (VTEPs) in an SDN via Hypervisor-Based Overlays system.[2] Given this background and common behavior of these tunneling technologies, the next sections take a deeper look at these technologies and examine how they differ from one another. These three technologies compete for market share, and while adoption rate statistics are elusive, we presume that:

- VXLAN will benefit from its considerable support by networking heavyweight Cisco.
- NVGRE will be adopted widely in Microsoft's popular Azure data centers.
- STT may struggle for adoption due to its relative newness, and the fact that VMware was a primary contributor and promoter of VXLAN, and hence has a significant installed base.

8.3.1 VIRTUAL EXTENSIBLE LOCAL AREA NETWORK

The VXLAN technology was developed primarily by VMware and Cisco, as a means to mitigate the inflexibility and limitations of networking technology. Some of the main characteristics of VXLAN are:

- VXLAN utilizes MAC-in-IP tunneling.
- Each virtual network or *overlay* is called a VXLAN segment.
- VXLAN segments are identified by a 24 bit segment ID, allowing for up to 2^{24} (approximately 16 million) segments.
- VXLAN tunnels are stateless.
- VXLAN segment end points are the switches that perform the encapsulation and are called VTEPs.
- VXLAN packets are unicast between the two VTEPs and use UDP over IP packet formats.
- It is UDP-based. The UDP port number for VXLAN is 4789.

Fig. 8.3 illustrates the format of a VXLAN packet. As you can see in the diagram, the outer header contains the MAC and IP addresses appropriate for sending a unicast packet to the destination switch, acting as a virtual tunnel end point. The VXLAN header follows the outer header and contains a VXLAN Network Identifier of 24 bits in length, sufficient for about 16 million networks.

The proponents of VXLAN technology argue that it assists load balancing within the network. Much of the load balancing within the data center network is based on the values of various packet fields including the destination and source port numbers. The fact that the source port of the UDP session is derived by hashing other fields of the flow results in a smooth distribution function for load balancing with the network.

[2]These are sometimes called *VXLAN Tunnel Endpoints*, which essentially means the same thing as the more generic and commonly used *Virtual Tunnel End Point*.

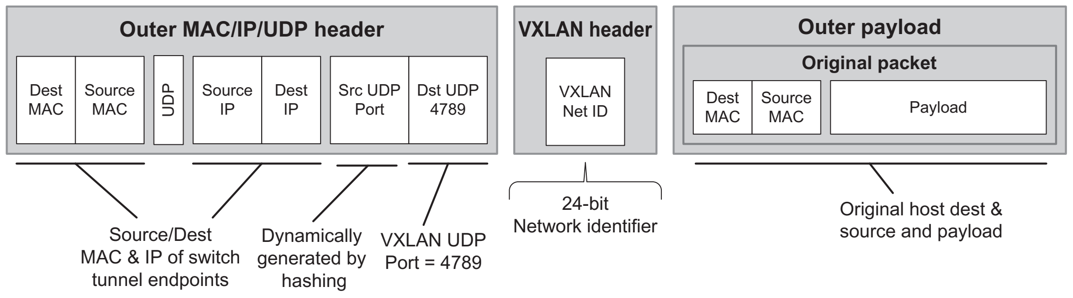

FIG. 8.3

VXLAN packet format.

8.3.2 NETWORK VIRTUALIZATION USING GRE

The NVGRE technology was developed primarily by Microsoft, with other contributors including Intel, Dell, and Hewlett-Packard. Some of the main characteristics of NVGRE are:

- NVGRE utilizes MAC-in-IP tunneling.
- Each virtual network or *overlay* is called a virtual layer two network.
- NVGRE virtual networks are identified by a 24 bit Virtual Subnet Identifier, allowing for up to 2^{24} (16 million) networks.
- NVGRE tunnels, like GRE tunnels, are stateless.
- NVGRE packets are unicast between the two NVGRE end points, each running on a switch. NVGRE utilizes the header format specified by the GRE standard [5,6].

Fig. 8.4 shows the format of an NVGRE packet. The outer header contains the MAC and IP addresses appropriate for sending a unicast packet to the destination switch, acting as a virtual tunnel end point, just like VXLAN. Recall that for VXLAN the IP protocol value was UDP. For NVGRE, the IP protocol value is 0x2F, which means GRE. GRE is a separate and independent IP protocol in

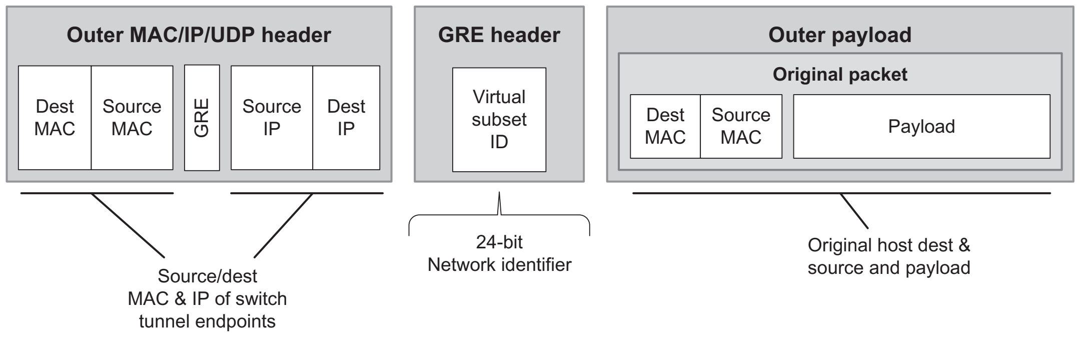

FIG. 8.4

NVGRE packet format.

the same class as TCP or UDP. Consequently, as you can see in the diagram, there are no source and destination TCP or UDP ports. The NVGRE header follows the outer header and contains an NVGRE Subnet Identifier of 24 bits in length, sufficient for about 16 million networks.

8.3.3 STATELESS TRANSPORT TUNNELING

Stateless Transport Tunneling (STT) is a recent entry into the field of tunneling technologies used for network virtualization. Its major sponsor was originally Nicira. Some of the main characteristics of STT are:

- STT utilizes MAC-in-IP tunneling.
- The general idea of a virtual network exists in STT, but is enclosed in a more general identifier called a Context ID.
- STT context IDs are 64 bits, allowing for a much larger number of virtual networks and a broader range of service models.
- STT attempts to achieve performance gains over NVGRE and VXLAN by leveraging the *TCP Segmentation Offload* (TSO) found in the *Network Interface Cards* (NICs) of many servers. TSO is a mechanism implemented on server NICs which allows large packets of data to be sent from the server to the NIC in a single send request, thus reducing the overhead associated with multiple smaller requests.
- STT, as the name implies, is stateless.
- STT packets are unicast between tunnel end points, utilizing TCP in the stateless manner associated with TSO. This means that it does not use the typical TCP windowing scheme, which requires state for TCP synchronization and flow control.

Fig. 8.5 shows the format of an STT packet. Like the other tunneling protocols discussed here, the outer header contains the MAC and IP addresses appropriate for sending a unicast packet to the destination switch, acting as a VTEP. For VXLAN, the IP protocol value was UDP and for NVGRE the IP protocol value was GRE. For STT, the IP protocol is TCP. The TCP port for STT has yet to be ratified, but, tentatively, the value 7471 has been used. The STT header follows the outer header and

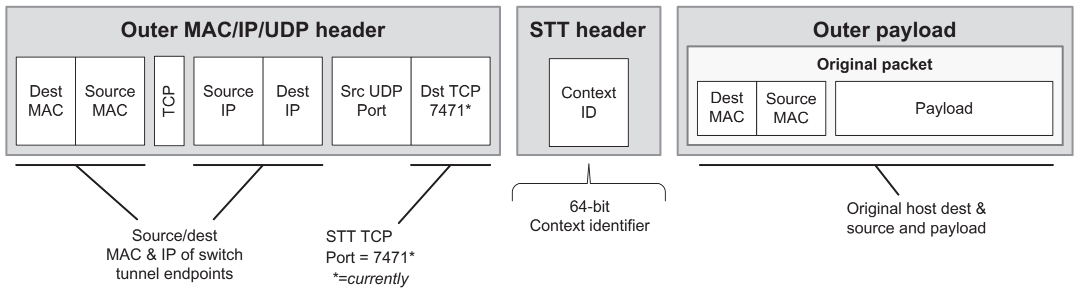

FIG. 8.5

STT packet format.

contains an STT Context Identifier of 64 bits in length, which can be subdivided and used for multiple purposes; however that is done, there is ample space for as many virtual networks as required.

8.4 PATH TECHNOLOGIES IN THE DATA CENTER

One of the critical issues in current data centers is the efficient use of the physical network links that connect network devices forming the data center's network infrastructure. With the size and demands of data centers, it is imperative that all links be utilized to their full capacity.

As we explained in Section 8.2.1, one of the shortcomings of layer two network technologies such as spanning tree is that it will intentionally block certain links in order to ensure a hierarchical network without loops. Indeed, in the data center, this drawback has become more important than the problem of slow convergence times which formed the bulk of the early critiques of STP.

Layer three networks also require intelligent routing of packets as they traverse the physical network. This section explores some path-related technologies that are intended to provide some of the intelligence required to make the most efficient use of the network and its interconnecting links.

8.4.1 GENERAL MULTIPATH ROUTING ISSUES

In a typical network of layer three subnets, there will be multiple routers connected in some manner as to provide redundant links for failover. These multiple links have the potential to be used for load balancing. Multipath routing is the general category for techniques which make use of multiple routes in order to balance traffic across a number of potential paths. Issues which must be considered in any multipath scheme are:

- The potential for *out-of-order delivery* (OOOD) of packets, which take different paths and must be reordered by the recipient, and
- The potential for maximum packet size differences on links within the different paths, causing issues for certain protocols, such as TCP and its path MTU discovery.

While these are general multipath issues, they are largely avoidable in a properly configured modern data center. Provided that the network performs per-flow load balancing and the core MTUs are much higher than the access MTUs these problems should be avoidable.

8.4.2 MULTIPLE SPANNING TREE PROTOCOL

The *Multiple Spanning Tree Protocol* (MSTP) was introduced to achieve better network link utilization with spanning tree technology when there are multiple VLANs present. Each VLAN would operate under its own spanning tree. The improved use of the links was to have one VLAN's spanning tree use unused links from another VLANs when reasonable to do so. MSTP was originally introduced as IEEE 802.1s. It is now part of IEEE 802.1Q-2005 [7]. It was necessary to have a large number of VLANs in order to achieve a well distributed utilization level across the network links as the only distribution was at VLAN-granularity. MSTP predates the Shortest Path Bridging protocol discussed below, which does not suffer this limitation.

8.4.3 SHORTEST PATH BRIDGING

IEEE 802.1aq is the *Shortest Path Bridging* (SPB) standard and its goal is to enable the use of multiple paths within a layer two network. Thus, SPB allows all links in the layer two domain to be active.

SPB is a link state protocol, which means that devices have awareness of the topology around them and are able to make forwarding decisions by calculating the best path to the destination. It uses the *Intermediate System to Intermediate System* (IS-IS) routing protocol [8] to discover and advertise network topology and to determine the paths to all other bridges in its domain.

SPB accomplishes its goals by using encapsulation at the edge of the network. This encapsulation can be either MAC-in-MAC (IEEE 802.1ah) or Q-in-Q (IEEE 802.1ad). We discussed Q-in-Q encapsulation briefly in Section 5.5.5.

Fig. 8.6 shows the frame format for Q-in-Q. As can be seen in the figure, Q-in-Q means pushing VLAN tags into the packet such that the inner tag becomes the original VLAN, also called the *Customer VLAN ID* (C-VLAN) or *C-VID*. The outer tag has multiple names, depending on the context of its use. They are known as *Metro Tag* for its use in Metro Ethernet backbones, and *Provider Edge VLAN* (PE-VLAN), *Service Provider VLAN ID* (S-VLAN) or *S-VID* for its use in provider VLANs. The *C-VID* represents the original customer VLAN tag.

When the packet arrives at the edge of the provider's interior network, a new VLAN tag is pushed into the packet identifying the VLAN on which the packet will travel when it is inside the provider's network. When it exits, that VLAN tag is removed.

Fig. 8.7 shows the frame format for MAC-in-MAC. The new MAC header includes the backbone's source (SA) and destination (DA) addresses. Also part of the header is the VLAN identifier for the backbone provider's network (B-VID), as well as the backbone's service instance ID (I-SID) which allows the provider to organize customers into specific services and provide appropriate levels of QoS and throughput to each service. When a packet arrives at the provider edge, the new header gets attached to the beginning of the packet and is sent through the provider's network. When it leaves the provider's domain, the header is stripped.

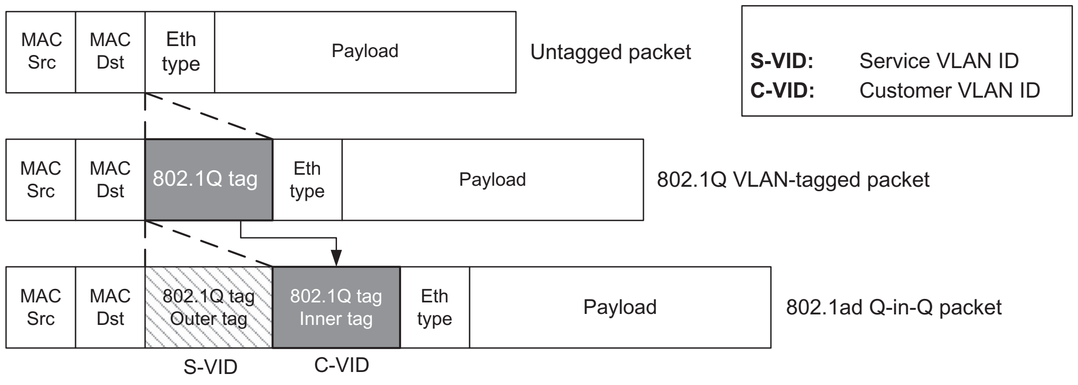

FIG. 8.6

Shortest path bridging—Q-in-Q (VLAN).

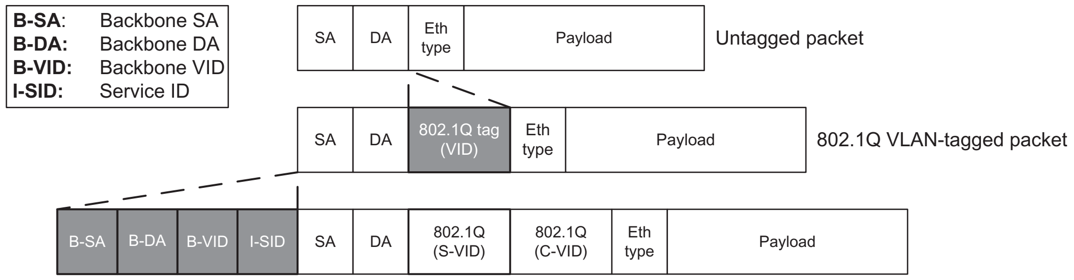

FIG. 8.7

Shortest path bridging—MAC-in-MAC.

8.4.4 EQUAL-COST MULTIPATH

One of the most important operations that must be undertaken in large networks is optimal path computation. Merely getting a packet to its destination is far simpler than getting it there in the most efficient way possible. In Section 1.5.2 we introduced OSPF and IS-IS as modern link-state protocols used for calculating routes in complex layer three networks. We just explained in Section 8.4.3 that IS-IS is used as the basis for calculating routes in the layer two SPB protocol. There is a general routing strategy called *Equal-cost multipath* (ECMP) that is applicable in the data center. Multipath routing is a feature that is explicitly allowed in both OSPF and IS-IS. The notion is that when more than one equal-cost path exists to a destination, these multiple paths can be computed by a shortest path algorithm and exposed to the packet forwarding logic. At that point some load balancing scheme must be used to choose between the multiple available paths. As several routing protocols can derive the multiple paths and there are many ways in which to load balance across the multiple paths, ECMP is more of a routing strategy than a specific technology. A discussion of some of the issues with ECMP can be found in [9]. An analysis of a specific ECMP algorithm is found in [10].

Another sophisticated path computation technology is the *Path Computation Element (PCE)-based Architecture*. As this has been used primarily in the WAN and not seen much use in the data center, we will defer deeper treatment of this topic to Section 9.1.1. It is likely that PCE will be applied in the data center for directing elephant flows.

8.4.5 SDN AND SHORTEST-PATH COMPLEXITY

Throughout this chapter we have made frequent reference to the importance of calculating optimal paths. We have implied that a centralized controller can perform this task better than the traditional distributed algorithm approach because: (1) it has a more stable, global view of the network, (2) it can take more factors into consideration than current distributed approaches do, including current bandwidth loads, and (3) the computation can be performed on the higher capacity memory and processor of a server rather than the more limited memory and processor available on the network devices. While all this is true, we should point out that the fundamental computational complexity of shortest-path calculation is not changed by any of these factors. Shortest-path remains a fundamentally hard problem which grows harder very fast as the number of switches and links scales. The well-known

Dijkstra's algorithm [11] for shortest path remains in wide use in *Interior Gateway Protocols* (IGPs) (it is used in both OSPF and IS-IS) and SDN is unlikely to alter that. Having a faster machine and a more stable data base of the network graph helps compute optimal paths more quickly; they do not affect the complexity of the fundamental problem.

It is worthwhile noting that some *Massively Scaled Data Centers* (MSDCs) are now abandoning an IGP altogether and instead use the *External Border Gateway Protocol* (EBGP) for all routing. This implies the use of a path-vector routing protocol instead of one based on the Dijkstra algorithm. This should scale better computationally as the network grows while maintaining awareness of distant link failures since a failed link will result in paths being removed from the routing table.

8.5 ETHERNET FABRICS IN THE DATA CENTER

Traffic engineering in the data center is challenging using traditional Ethernet switches arranged in a typical hierarchy of increasingly powerful switches as one moves up the hierarchy closer to the core. While the interconnecting links increase in bandwidth as one moves closer to the core, these links are normally heavily oversubscribed and blocking can easily occur. By oversubscription we mean that the aggregate potential bandwidth entering one tier of the hierarchy is greater than the aggregate bandwidth going to the next tier.

An alternative to a network of these traditional switches is the Ethernet fabric [12]. Ethernet fabrics are based on a nonblocking architecture where every tier is connected to the next tier with equal or higher aggregate bandwidth. This is facilitated by a topology referred to as *fat tree* topology. We depict such a topology in Fig. 8.8. In a fat tree topology, the number of links entering one switch in a given tier of the topology is equal to the number of links leaving that switch toward the next tier. This implies that as we approach the core, the number of links entering the switches is much greater than those toward the leaves of the tree. The root of the tree will have more links than any switch lower than it. This topology is also often referred to as a *Clos* architecture. Clos switching architecture dates from the early days of crossbar telephony and is an effective nonblocking switching architecture. Note that the Clos architecture is not only used in constructing the network topology but may also be used in the internal architecture of the Ethernet switches themselves.

For Ethernet fabrics the key characteristic is that the aggregate bandwidth not decrease from one tier to the next as we approach the core. Whether this is achieved by a larger number of low bandwidth links or a lower number of high bandwidth links does not fundamentally alter the fat tree premise.

As we approach the core of the data center network, these interconnecting links are increasingly built from 100Gb links and it is costly to simply try to overprovision the network with extra links. Thus, part of the Ethernet fabric solution is to combine it with ECMP technology so that *all* of the potential bandwidth connecting one tier to another is available to the fabric.

DISCUSSION QUESTION:

We have discussed a number of tunneling and path technologies as potential solutions to issues faced by networking administrators in the data center today. Of these, which do you believe is best suited to solving these issues, and why? Might there be solutions consisting of a combination of these technologies?

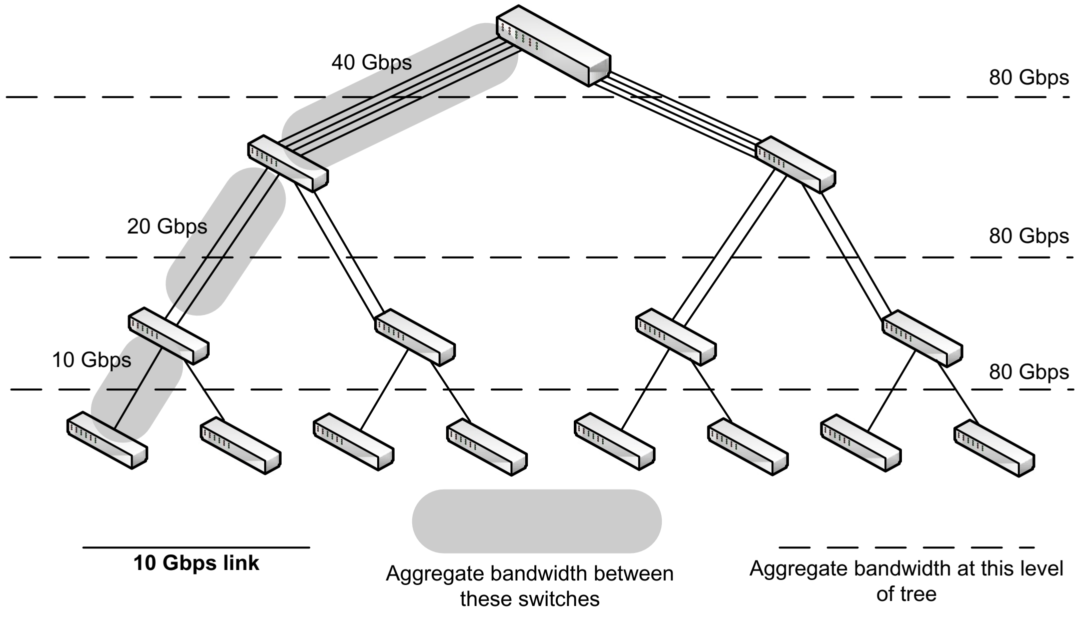
40 Gbps
80 Gbps
20 Gbps
80 Gbps
10 Gbps
80 Gbps
10 Gbps link
Aggregate bandwidth between these switches
Aggregate bandwidth at this level of tree

FIG. 8.8

Fat tree topology.

8.6 SDN USE CASES IN THE DATA CENTER

We have described several definitions of software defined networking: the original SDN which we call Open SDN, SDN via APIs, and SDN via Hypervisor-Based Overlays. We additionally described SDN via Opening Up the Device, but here we will focus on the first three, as they are the only ones to have gained significant commercial traction as of this writing. How do these versions of SDN address the needs of today's data center as we have described them in this chapter? The following sections answer that question.

For each data center requirement we presented in Section 8.2, we consider each of the three SDN technologies. We present them in the order SDN via Overlays first, followed by Open SDN, and, lastly, SDN via APIs. The reader should note that in this book when we use the expression SDN via Overlays we mean SDN via *Hypervisor-Based* Overlays. This distinction is important as there are other overlay mechanisms that are not hypervisor-based. The reason for this ordering is because SDN via Overlays is a technology specifically addressing the data center and so one would expect the Overlay solution to directly address most of these issues.

Note that Table 8.1 briefly summarizes the ability of the various SDN definitions to address the data center needs that we have mentioned in this chapter. We discuss aspects of this table in the sections that follow.

Table 8.1 SDN Alternatives' Support of Data Center Needs

Need	SDN via Overlays	Open SDN	SDN via APIs
Overcoming Network Limitations	Inherent	Feasible	Implementation-dependent
Adds, Moves, Deletes	Inherent	Inherent	Implementation-dependent
Failure Recovery	Unaddressed	Inherent	Implementation-dependent
Multitenancy	Inherent	Inherent	Implementation-dependent
Traffic Engineering and Path Efficiency	Unaddressed	Inherent	Implementation-dependent

8.6.1 OVERCOMING CURRENT NETWORK LIMITATIONS

In this section we examine how SDN via Overlays, Open SDN and SDN via APIs address the limitations of MAC address table size and maximum number of VLANs, and other aspects related to Table 8.1.

SDN via Overlays

The simplest and most readily available solution to these problems involves tunneling, so *SDN via Overlays* is an obvious choice here. The only MAC addresses visible through the physical network are the MAC addresses of the tunnel end points, which are at the hypervisors. As a trivial example, if there are eight VMs per hypervisor, then you have reduced the total number of MAC addresses by a factor of eight. If the tunnel end points are further upstream or the number of VMs per hypervisor is higher, the MAC address savings are even greater.

For the issue of VLAN exhaustion, that is, exceeding the limit of 4094, this solution is superior, as the new mechanism for multitenancy is tunnels, not VLANs. As we explained in Section 8.3, the number of tunneled networks or *segments* can be 16 million or greater using VXLAN, NVGRE, or STT tunneling technologies.

As for the issue of spanning tree convergence and using all links in the physical infrastructure, SDN via Overlays does not address issues related to the physical infrastructure. The network designer would have to use current non-SDN technologies to mitigate these types of physical limitations.

When we say that SDN via Overlays leaves unaddressed the issues of Failure Recovery and Traffic Engineering, it is important for the reader to understand that this technology does not prevent solutions to these issues to be implemented in the underlay network. We simply mean that SDN via Overlays does not *itself* provide the solution.

Open SDN

Open SDN has the capability of addressing these network limitations as well. It does not, however, *inherently* resolve these limitations in the same way as does SDN via Overlays due to that alternative's basic nature of using tunnels. Moving control functionality off of the device to a centralized controller does not directly address limitations such as MAC address table size and the maximum number of VLANs.

However, Open SDN is well-suited to creating a solution that is an instance of SDN via Overlays. The SDN controller can create tunnels as required at what will become the tunnel end points, and then OpenFlow rules are used to push traffic from hosts into the appropriate tunnel. Since hardware exists that has built-in tunneling support, SDN devices can be built that derive these benefits from tunneling, but with the performance gain of hardware. Thus, Open SDN can solve these network limitations similarly to the Overlay alternative.

SDN via APIs

Adding SDN APIs to networking devices does not directly address network limitations as we have discussed them in this chapter. However, some data center SDN via APIs solutions have been created using existing or new protocols. In particular:

- Cisco's *APIC-Data Center* (APIC-DC) utilizes a new protocol called *OpFlex*. This protocol is designed to carry policy-level information from an the APIC-DC controller, to intelligent devices capable of rendering the policies received.
- Juniper's *Contrail* and its open-source variant *OpenContrail*, utilize existing protocols such as NETCONF, MPLS, and XMPP combined with APIs existing in devices today to create a virtualized network targeted at the data center.

So while it is true that SDN via APIs inherently does not address network limitations, these creative uses of existing protocols and APIs can yield improvements in networking behavior.

8.6.2 ADDING, MOVING, AND CHANGING RESOURCES

Agility, automation, fast and efficient adds, moves, and changes—this type of functionality is critical in data centers in order to keep pace with speeds brought about by automation of servers and storage. We now examine the ways in which SDN technology can address these issues.

SDN via Overlays

The primary attribute of SDN via Overlays as it addresses adds, moves, and changes is that the technology revolves around virtualization. It does not deal with the physical infrastructure at all. The networking devices that it manipulates are most often the virtual switches that run in the hypervisors. Furthermore, the network changes required to accomplish the task are simple and confined to the construction and deletion of virtual networks, which are carried within tunnels that are created expressly for that purpose. These are easily manipulated via software.

Consequently, the task of making adds, moves, deletes, and changes in an overlay network is quite straightforward and is easily automated. Because the task is isolated and constrained to tunnels, problems of complexity are less prevalent compared to what would be the case if the changes needed to be applied and replicated on all the physical devices in the network. Thus, many would argue that overlays are the simplest way to provide the automation and agility required to support frequent adds, moves, deletes, and changes.

A downside of this agility is that, since the virtual networks are not tightly coupled with the physical network, it is easy to make these adds, moves, and changes without being certain that the underlying physical network has the capacity to handle them. The obvious solution is to overengineer the physical network with great excess capacity, but this is not an efficient solution to the problem.

Open SDN

As we discussed in the previous section, if Open SDN is being used to create tunnels and virtual networks, then it is straightforward for Open SDN to achieve the same results as discussed earlier. The task is to create the overlay tunnels as required and to use OpenFlow rules to push packets into the appropriate tunnels.

In addition to the advantages of virtual networks via tunnels, Open SDN offers the ability to change the configuration and operation of the physical network below—what some refer to as the *underlay*. The real advantage of this capability is described in Section 8.6.5.

SDN via APIs

APIs provide a programmatic framework for automating tasks that would otherwise require manual intervention. Having a controller aware of server virtualization changes that can respond with changes to the network is a definite advantage. Good examples include complete API-based solutions in the data center such as Cisco's APIC-DC and Juniper's Contrail. Both of these solutions use existing or new proprietary APIs provided on their respective devices in order to foster the agility required in data centers. A possible limitation of the API solution is that, if using legacy protocols, the controller may be constrained by the limitations of the devices themselves (a limitation which is not true for Open SDN or Overlays). However, remember that the main facet of supporting adds, moves, and changes, is the ability to do so in an automated and agile manner. From this perspective, with a centralized controller managing the network, SDN via APIs should provide significant improvement over current legacy networks.

8.6.3 FAILURE RECOVERY

Data centers today have mechanisms for achieving high availability, redundancy, and recovery in the case of failure. However, as mentioned earlier, these mechanisms have deficiencies in the predictability and optimization of those recovery methods. Consequently, we would expect SDN to provide some assistance in meeting those specific needs.

Note that failures on the compute side, i.e., physical servers or virtual machines, are really just a subset of the previous section on *adds, moves, and changes* and have been addressed in that context.

SDN via Overlays

Because it does not deal at all with the physical network below it, overlay technology offers little in terms of improving the failure recovery methods in the data center. If there are failures in the physical infrastructure, those must be dealt with via the mechanisms already in place, apart from overlays. In addition, the interplay between the virtual and physical topologies can sometimes be difficult to diagnose when there are problems.

Open SDN

One of the stated benefits of Open SDN is that with a centralized controller the whole network topology is known and routing (or, in this case, rerouting) decisions can be made which are consistent and predictable. Furthermore, those decisions can incorporate other sources of data related to the routing decisions, such as traffic loads, time of day, even scheduled or observed loads over time. Creating a solution such as this is not trivial, but the SDN application responsible for such failure recovery functionality can leverage existing and emerging technologies, such as IS-IS and PCE.

SDN via APIs

Versions of SDN via APIs such as Cisco's APIC-DC and Juniper's Contrail can utilize knowledge about the network in a similar manner to Open SDN, and can make use of those existing legacy APIs on devices to achieve superior availability and recovery from failures. This is in fact the direction that API-based solutions have been headed since 2013.

8.6.4 MULTITENANCY

With large data centers and cloud computing, it is frequently the case that more and more tenants are passing traffic along the same physical network infrastructure and therefore sharing the same physical communications links. The traditional way of achieving the separation required by this sharing of network bandwidth has been through the use of VLANs. We have shown how the maximum number of VLANs is no longer adequate for today's data centers.

SDN via Overlays

Overlay technology resolves the multitenancy issue by its very nature through the creation of virtual networks that run on top of the physical network. These substitute for VLANs as the means of providing traffic separation and isolation. In overlay technologies, VLANs are only relevant within a single tenant. For each tenant, there is still the 4094 VLAN limit, but that seems to currently suffice for a single tenants's traffic.

Open SDN

Open SDN can implement network virtualization using layer three tunnel-based overlays in a manner very similar to SDN via Overlays. Open SDN offers other alternatives as well, however. Other types of encapsulation (e.g., MAC-in-MAC, Q-in-Q) can also be employed to provide layer two tunneling, which can provide multiplicative increases in the number of tenants. For example, using Q-in-Q can theoretically result in 4094 times 4094 VLANs, or roughly 16 million, the same value as is possible with the Overlay solutions utilizing VXLAN or NVGRE.

SDN via APIs

Many legacy networks depend on VLANs to provide network virtualization, and those VLANs are established and configured via APIs. As such, SDN via APIs can play a role in addressing multitenancy issues. Moreover, as technologies such as VXLAN become more prevalent in network devices, it becomes possible to utilize these capabilities using APIs on devices. While traditionally used in service provider networks, protocols such as MPLS are now being applied in the data center (e.g., Contrail), using these legacy APIs to provide a level of multitenancy well beyond what was possible with VLANs. For these reasons, SDN via APIs can enhance multitenancy support in data center environments.

8.6.5 TRAFFIC ENGINEERING AND PATH EFFICIENCY

In the past, networking hardware and software vendors created applications which measured traffic loads on various links in the network infrastructure. Some even recommended changes to the infrastructure in order to make more efficient use of network bandwidth. As good as these applications may have been, the harsh reality was that the easiest way to ensure optimal response times and minimal latency was to overprovision the network. In other words, the simplest and cheapest solution was almost always to purchase more and newer networking gear to make certain that links were uncongested and throughput was maximized.

However, with the massive expansion of data centers due to consolidation and virtualization, it has become imperative to make the most efficient use of networking bandwidth. *Service level agreements* (SLAs) with clients and tenants must be met, while at the same time costs need to be cut in order for the data center service to be competitive. This has placed an added importance on traffic load measurements and using that information to make critical path decisions for packets as they traverse the network.

SDN via Overlays

In a similar manner to the issue of failure recovery SDN via Overlays does not have much to contribute in this area because of the fact that it does not attempt to affect the physical network infrastructure. Traffic loads as they pass from link to link across the network are not a part of the discussion concerning traffic which is tunneled across the top of the physical devices and interfaces. Thus, SDN via Overlays is dependent on existing underlying network technology to address these types of issues.

Open SDN

Open SDN has major advantages here in the areas of centralized control and having complete control of the network devices. Open SDN has centralized control with a view of the full network and can make decisions predictably and optimally. It also controls the individual forwarding tables in the devices and, as such, maintains direct control over routing and forwarding decisions.

Two other major ways that Open SDN can impact the traffic moving through the network are: (1) path selection and (2) bandwidth management on a per-flow basis. Open SDN can make optimal path decisions for each flow. It can also prioritize traffic down to the flow level, allowing for granular control of queuing, using methods such as IEEE 802.1 *Class of Service* (CoS), IP-level *Type of Service* (ToS), and *Differentiated Services Code Point* (DSCP).

Fig. 8.9 provides an illustration of the difference between shortest path via the current node-by-node technology and optimal path, which is possible with an SDN controller which can take into consideration more network parameters such as traffic data, in making its decision.

Open SDN is explicitly designed to be able to directly control network traffic on the switching hardware down to the flow level. We contrast this with the overlay alternative where network traffic does exist at the flow level in the virtual links, but not natively in the switching hardware. Therefore, Open SDN is particularly well-suited to addressing traffic engineering and path efficiency needs.

SDN via APIs

The core feature of SDN via APIs is to set configuration of networking devices in a simpler and more efficient manner. Since this is done from a centralized controller, a solution using APIs would enhance automation, and will yield a greater degree of agility. While legacy APIs were generally not designed to handle the time-critical, highly granular needs inherent in traffic engineering, *extended* APIs can improve traffic engineering even in legacy devices with local control planes. This is rendered feasible through the use of SDN applications running on an API-based controller such as OpenDaylight,

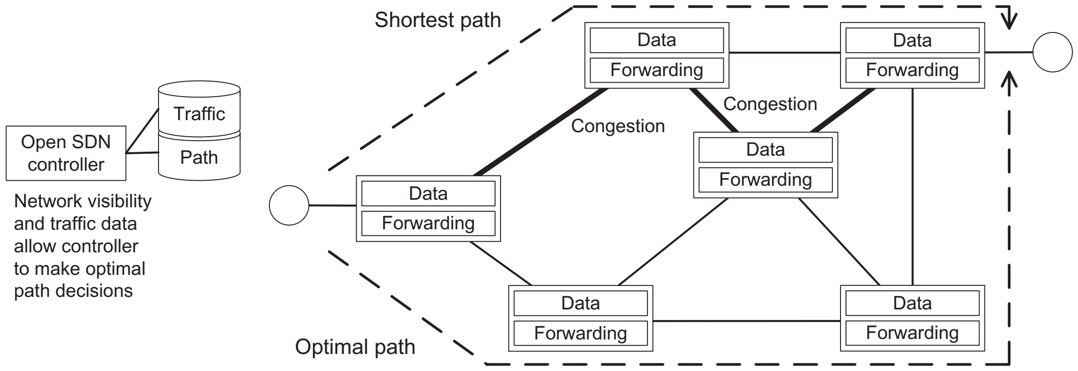

FIG. 8.9

Optimal path selection by SDN controller.

Contrail, or APIC-DC. In fact, though Fig. 8.9 illustrates the capabilities of OpenFlow for optimal path selection, a similar solution is provided via OpenDaylight's BGP-LS/PCE-P plugin.

It is worth noting that *policy-based routing* (PBR) has the ability to direct packets across paths at a fairly granular level. In theory one could combine current traffic monitoring tools with PBR and use current SNMP or CLI APIs to accomplish the sort of traffic engineering we discuss here. RSVP and MPLS-TE are examples of traffic engineering protocols that may be configured via API calls to the device's control plane.

However, those APIs and PBR did not traditionally enable frequent and dynamic changes to paths and cannot react as quickly in the face of change as can OpenFlow. However, with new solutions such as APIC-DC and Contrail, utilizing policy-based and path-based protocols, vendors are thrusting SDN APIs into a more agile SDN-based future.

8.7 COMPARISON OF OPEN SDN, OVERLAYS, AND APIs

We see in Table 8.1 a comparison of the different types of SDN and their contribution to alleviating problems with network in the data center. We take a closer look in the following sections.

SDN via Overlays

- The technology is designed for solving data center issues.
- The technology typically utilizes contemporary data center technologies, specifically VXLAN and NVGRE, and, thus, the technology is not entirely new.
- The technology creates a virtual overlay network and thus does a very good job with overcoming networking limitations and improving agility.
- The technology does not address any issues related to improving the behavior of the physical infrastructure. This has the advantage, though, of incurring little or no costs related to new networking equipment.

- It is sometimes difficult to diagnose problems and determine if they relate to the virtual network or the physical one.

Open SDN

- The technology is designed for solving networking issues in a wide range of domains, not just data centers.
- Open SDN can be implemented using networking devices designed for Open SDN. Legacy switching hardware can often be enhanced by adding OpenFlow to function as Open SDN devices.
- The technology, used in conjunction with virtual networks (e.g., VXLAN and NVGRE), does a very good job addressing all the networking issues we have discussed.
- The technology is broader in scope; hence, it is more disruptive and may pose more transitional issues than does SDN via Overlays.
- Although both Open SDN and SDN via Overlays allow significant innovation, the Open SDN use cases discussed in Chapter 9 will show how implementers have *radically* redefined the network. In so doing, they abandon legacy constructs and structure the network to behave in exact alignment with application requirements.

SDN via APIs

- The technology can utilize existing network devices with minimal upgrades.
- Some solutions (e.g., APIC-DC and OpFlex) are natively policy-based solutions, whereas others are added onto an existing solution.
- Other solutions translate existing successful architectures from one domain into the data center domain. This is the case for Contrail which uses MPLS technology in the data center, though MPLS has traditionally been used in the WAN. The effectiveness of this translation has yet to be proven, but it certainly holds promise.
- There is major vendor support from Juniper and Cisco, each for their own data center solution, which can prove to be significant in determining the success or failure of a new technology.

These points indicate that SDN via Overlays is a good solution and may be an easier transitional phase in the near term, while Open SDN holds the most promise in a broader and more comprehensive sense. While Table 8.1 illustrated that SDN via APIs may not inherently be an optimal solution in the data center due to dealing with legacy issues, we have pointed out that a number of data center problems may be addressed by particular SDN via APIs solutions. The more simple SDN via APIs solutions may in fact represent the easiest transition plan for network programmers in the short term as less radical change is needed than that required moving to either the Overlay or Open SDN approach.

DISCUSSION QUESTION:

Which of the types of SDN (Open, APIs, Overlays) is your favorite for resolving immediate issues in the data center? Which do you consider to be the best long-term strategy? Do you believe it is wise to favor near-term pragmatism, or long-term viability, for solving these issues?

8.8 REAL-WORLD DATA CENTER IMPLEMENTATIONS

In this chapter we have spent time defining the needs of modern data centers, and we have investigated the ways that each SDN technology has addressed those needs. One may wonder if anybody is actually implementing these SDN technologies in the data center. The answer is that, although it is still a work in progress, there are organizations which are running SDN pilots and in some cases placing it into production. In Table 8.2 we list a number of well-known enterprises and their early adopter implementations of SDN in their data centers. The table shows a mix of Open SDN and SDN via Overlays, but no examples of SDN via APIs. The bulk of the API-oriented efforts are not really what we have defined as SDN. They use technologies similar to what we described in Sections 3.2.3 and 3.2.4, that is, orchestration tools and VMware plugins to manipulate networks using SNMP and CLI. Some experimentation with newer SDN APIs is presumably underway, but as of this writing production data center SDN networks tend more to be hypervisor-based overlays with a few examples using Open SDN. It is worth pointing out that Juniper's Contrail is somewhat of an outlier here. We have defined Contrail as an example of both SDN via APIs (Section 6.2.5) *and* an SDN via Overlays technology (Section 6.3.3). Contrail has been deployed in a commercial data center environment [13] and thus might be considered as an example of a production SDN via APIs deployment. In any event, between Open SDN and the overlay approach, our observation is that the extremely large greenfield data center examples gravitate toward Open SDN, whereas hypervisor-based overlays have seen greater adoption in the more typical-scale data centers.

8.9 CONCLUSION

This chapter has described the many significant and urgent needs that have caused data center networking to become a stumbling block to technological advancement in that domain. We examined

Table 8.2 Data Center SDN Implementations

Enterprise	SDN Type	Description
Google	Open SDN	Has implemented lightweight OpenFlow switches, an OpenFlow Controller and SDN Applications for managing the WAN connections between their data centers. They are moving that Open SDN technology into the data centers.
Microsoft Azure	Overlays	Implementing Overlay technology with NVGRE in vSwitches, communicating via enhanced OpenFlow, creating tens of thousands of virtual networks.
Ebay	Overlays	Creating public cloud virtual networks, using VMware's Nicira solution.
Rackspace	Overlays	Creating large multitenant public cloud using VMware's Nicira solution.

new technologies and standards that attempt to address those needs. Then we presented how the three main SDN alternatives made use of those technologies, as well as new ideas introduced by SDN to address those needs. Lastly, we listed how some major enterprises are using SDN today in their data center environments.

In Chapter 9 we will consider other use cases for SDN, such as carrier and enterprise networks.

REFERENCES

[1] Kompella K. New take on SDN: does MPLS make sense in cloud data centers? SDN Central, December 11, 2012. Retrieved from: http://www.sdncentral.com/use-cases/does-mpls-make-sense-in-cloud-data-centers/2012/12/ .

[2] Mahalingam M, Dutt D, Duda K, Agarwal P, Kreeger L, Sridhar T, et al. VXLAN: a framework for overlaying virtualized layer 2 networks over layer 3 networks. Internet Draft, Internet Engineering Task Force, August 26, 2011.

[3] Sridharan M, et al. NVGRE: network virtualization using generic routing encapsulation. Internet Draft, Internet Engineering Task Force, September 2011.

[4] Davie B, Gross J. STT: a stateless transport tunneling protocol for network virtualization (STT). Internet Draft, Internet Engineering Task Force, March 2012.

[5] Farinacci D, Li T, Hanks S, D M, Traina P. Generic Routing Encapsulation (GRE). RFC 2784, Internet Engineering Task Force, March 2000.

[6] Dommety G. Key and sequence number extensions to GRE. RFC 2890, Internet Engineering Task Force, September 2000.

[7] IEEE standards for local and metropolitan area networks—virtual bridged local area networks. IEEE 802.1Q-2005. New York: IEEE; 2005.

[8] Information technology—telecommunications and information exchange between systems—intermediate system to intermediate system intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473). ISO/IEC 10589:2002(E), Switzerland, November 2002.

[9] Thaler D, Hopps C. Multipath issues in unicast and multicast next-hop selection. RFC 2991, Internet Engineering Task Force, November 2000.

[10] Hopps C. Analysis of an equal-cost multi-path algorithm. RFC 2992, Internet Engineering Task Force, November 2000.

[11] Dijkstra EW. A note on two problems in connection with graphs. Numer Math 1959;1:269–71.

[12] Ammirato J. Ethernet fabric switching for next-generation data centers. Network World, November 03, 2008. Retrieved from: http://www.networkworld.com/news/tech/2008/110308-tech-update.html .

[13] Moriarty E. Juniper Contrail gets deployed by TCP cloud. NetworkStatic, September 22, 2014. Retrieved from: https://www.sdxcentral.com/articles/news/juniper-contrail-sdn-controller-tcp-cloud/2014/09/ .