CHAPTER

Network Virtualization 7

Network virtualization is similar to server virtualization but instead of dividing up a physical server among several virtual machines, physical network resources are divided up among multiple virtual networks. This is useful in applications such as multi-tenant data center environments where each tenant can be allocated multiple virtual machines as well as a virtual network connecting these virtual machines. One way to do this is to provide a special label within each frame that identifies the virtual network that it belongs to. Labeling frames and forwarding data based on these labels is sometimes called network tunneling. Tunneling has been used in telecom networks for some time, but these types of tunnels have limitations that make them less useful in virtualized data center networks. This chapter will provide a background on multi-tenant data center requirements along with a background on tunneling techniques used in telecom networks. We will then describe two new industry standards for virtualized data center networks along with example use cases. At the end of the chapter, we will discuss the various tunneling locations that can be used at the edge of the network along with how loads can be distributed through these tunnels.

MULTI-TENANT ENVIRONMENTS

The goal of a multi-tenant public cloud data center is to make the customer experience much the same as if they were using their own private data center. An additional goal of the cloud service provider is to have a flexible allocation of resources so they can quickly adapt to changing needs in order to reduce both capital expense and operating expense. For several years now, virtual machines have been used along with virtual switches to improve server utilization as we described in the last chapter. Attention is now turning to the virtualization of networking resources in a way that maintains a private network experience. Figure 7.1 shows a simplified view of a virtualized cloud data center including the physical servers and physical network components.

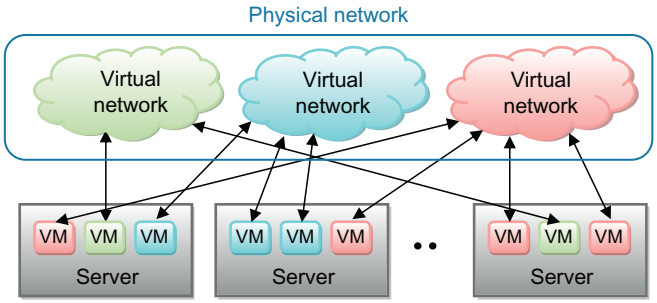

FIGURE 7.1

Logical diagram of multitenant data center.

In this example, the physical resources are divided up among three different tenants. The server administrator allocates virtual machine resources to each tenant that may reside on different servers across the network. The network administrator also allocates virtual network resources to each tenant that interconnects the given tenant's virtual machines. The virtual machine network connections and virtual networks are isolated from each other using special headers that will be described further in this chapter.

Network requirements

Virtualized networking places several requirements on the physical data center network. In many cases, the virtual network is emulating a layer 2 Ethernet network, and all traffic sent or received from the VMs are layer 2 frames that must be tunneled through the data center network without the knowledge of the VMs. To do this, some entity in the server or attached to the server must encapsulate the layer 2 frame with a tunnel header on the transmit side and de-encapsulate the tunnel header on the receive side. The tunnel label, along with other header information, is used to forward the packet through the layer 3 physical network. By using a unique tunnel value (or tag) for each virtual network, separation and security can be maintained between tenants. The number of tunnel values increases dramatically for large data centers which may host thousands of tenants.

One might ask the question, Why not simply use the IEEE virtual local area network (VLAN) tag to identify unique tenants just as it is used to identify different virtual local area networks within a corporate network? The reason for not using this tag is that corporate clients leasing public cloud services want to maintain their own VLAN information to separate different departments just as they would in their own data center. So the tunnel labels cannot share this tag due to conflicts with tenant VLAN information. In addition, the VLAN tag only supports 4096 total values, restricting the maximum number of tenants that a cloud data center can support.

MAC address learning

Because most tunneling endpoints in the network appear as layer 2 ports to the host virtual machines, media access control (MAC) address learning must also be supported through these tunnels. Before we get into the details of these tunneling protocols, let's discuss how traditional layer 2 MAC address learning is accomplished.

When a frame comes into a layer 2 switch, it contains both a destination MAC (DMAC) address and a source MAC (SMAC) address and, in most cases, also includes a VLAN tag. The switch first compares the SMAC/VLAN pair against information in its local MAC address table. If no match is found, it adds this information to the MAC table along with the switch port on which the frame arrived. In other words, the switch has just learned the direction (port number) of a given MAC address based on the received SMAC. Now any frames received on other ports with this MAC address and VLAN will be forwarded to this switch port. This process is called MAC address learning.

But what happens if a frame arrives and has a DMAC/VLAN pair that is not currently in the MAC address table (called an unknown unicast address)? In this case, the frame is flooded (broadcast) to all egress ports except the port it arrived on. In this way, the frame should eventually arrive at its destination through the network. Although flooding ties up network bandwidth, it is considered a rare enough event that it will not impact overall network performance. The assumption is that the device with the unknown destination address will eventually receive the frame and then send a packet back through the network from which its source address can be learned and added to the various switch address tables throughout the network. This can happen fairly rapidly as many network transactions require some sort of response and protocols such as transmission control protocol (TCP) generate acknowledgment packets. These responses and acknowledgments can be sent back to the originator by using the SMAC in the received flooded frames.

TRADITIONAL NETWORK TUNNELING PROTOCOLS

Various tunneling protocols have been used for years in telecom networks. In some cases, this has been driven by the need to isolate different customers as data is transported across Internet service provider networks. In some cases, this has been driven by the need to simplify the forwarding process as data is transported across IP networks. In this section, we will discuss both the Q-in-Q and multiprotocol label switching (MPLS) tunneling protocols that are used in carrier networks. For completeness, we will also describe the VN-Tag that we introduced in the last chapter and how it could be used as a tunnel tag. For each tunneling protocol, we will describe some properties that limit its usefulness for virtualized networking in large cloud data center networks.

Q-in-Q

Within a metropolitan area network, a given service provider may carry traffic for a corporate client that is routed between different facilities in different parts of the city. These corporate clients want their networks to appear as one large corporate network even though they span multiple locations. In order to tunnel client data through their networks, service providers can add a second, outer, VLAN tag as shown in Figure 7.2. This was standardized by the IEEE as 802.1ad and is used in many provider networks today (sometimes called Provider Bridging). Because this is an extension to the 802.1Q standard, it has become more commonly known as Q-in-Q.

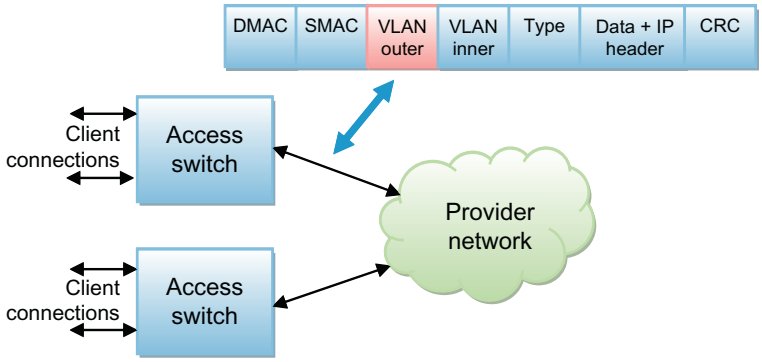

FIGURE 7.2

Q-in-Q forwarding and frame format.

In order to protect the inner VLAN tag for customer use, an outer VLAN tag is added where the 12-bit VLAN ID field is used to identify a given customer. This tag is added and removed by the access switch and is used to isolate traffic in the provider network. All forwarding and learning in the provider network can be done with the MAC addresses and outer VLAN tag. In some cases, the VLAN 3-bit priority field is also used to assign class of service in the provider network. The inner VLAN remains untouched and can be used by the customer for their own purposes. One problem with this technique is that the VLAN ID is limited to 12-bits which can contain only up to 4096 customer IDs. Because of this, it is not very useful for tunneling in large cloud data center networks. In fact, many large telecom service providers use techniques such as MPLS Transport Profile (MPLS-TP) instead; this will be described in more detail below.

MPLS

We briefly described MPLS in Chapter 2 as a way to reduce the frame processing requirements when forwarding frames through a TCP/IP network. In the core of a TCP/IP network, a given switch/router must deal with multiple high bandwidth streams, and matching multiple frame header fields at these high rates can tax the

available processing resources. Frame process rates are lower at the network edge where incoming traffic operates at lower overall bandwidth. To take advantage of these conditions, MPLS was first proposed by Ipsilon Networks and then handed over to the Internet Engineering Task Force (IETF) in the late 1990s. Figure 7.3 shows the MPLS label location in the frame header along with the switch and router components in an MPLS network.

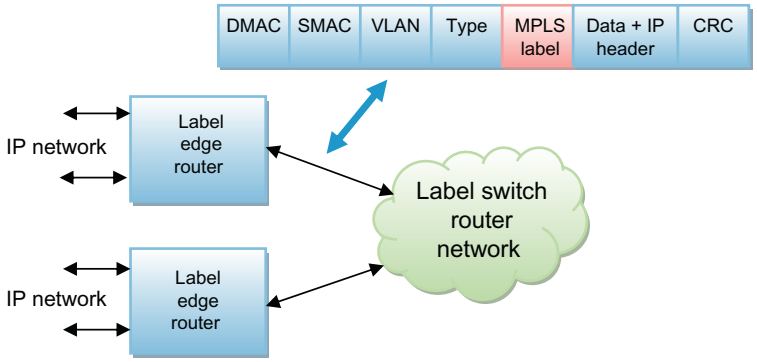

FIGURE 7.3

MPLS forwarding and frame format.

The MPLS label edge router (LER) does the frame match heavy lifting and then adds an MPLS label to the frame as shown in Figure 7.3. This is known as a label 'push' operation. To simplify forwarding in the network core, the MPLS label switch routers (LSRs) only need to examine the MPLS label in order to forward the frame. At the MPLS network egress, the label is removed in what is referred to as a 'pop' operation. It's possible to have a hierarchy of MPLS networks where one network is tunneled through another network by adding additional MPLS labels. In these cases, a LSR may need to examine multiple labels and potentially perform multiple push and pop operations on a given frame,

Since its introduction in the late 1990s, MPLS has found success in many areas. It is used in both IPv4 and IPv6 networks along with Virtual Private Networks (VPNs) and in carrier Ethernet networks using the MPLS-TP. While many people have proposed using MPLS to provide network virtualization and tunneling in the cloud data center, it has not caught on yet and the industry seems to be instead migrating toward protocols such as Virtual Extensible LAN (VXLAN) and Network Virtualization Generic Routing Encapsulation (NVGRE) which are described in the next sections. There are several reasons for this. Some people feel that MPLS is too complex and expensive to implement at the edge of a data center network. In many cases, tunneling occurs within the vSwitch and hypervisor which both consume server processor cycles. Any tunneling protocol implemented here must be as simple as possible in order to minimize CPU overhead. In addition, multicast was not an important requirement for the original MPLS standard. As we will describe in the VXLAN section of this chapter, multicast is an important feature for MAC learning in these layer 2 tunneled

virtual networks. In any case, MPLS seems to be as tenacious as Ethernet, and we may find its adoption in some virtual data center networks despite these shortcomings.

VN-Tags

Cisco and other companies originally proposed the VN-Tag protocol which became the basis for the Bridge Port Extension standard called IEEE 802.1qbh as we described in the last chapter. Figure 7.4 shows the VN-Tag location in a standard Ethernet frame and where it is used in the data center network. The VN-Tag contains a virtual interface (VIF) identifier which can be used to forward frames to any type of VIF such as a virtual machine within a server. A single physical network connection to the server using a VN-Tag capable network interface controller (NIC) may, therefore, utilize tags for multiple VIFs.

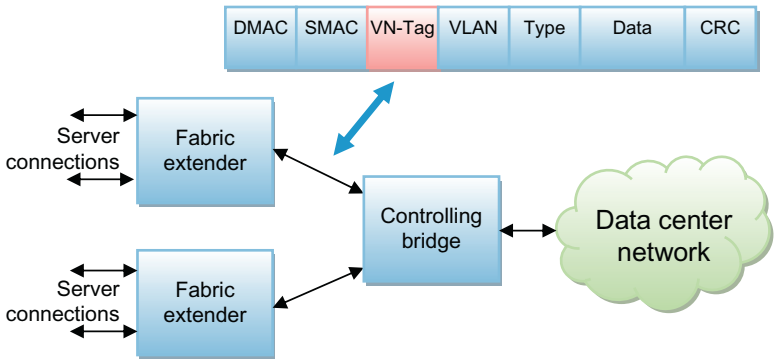

FIGURE 7.4

VN-Tag forwarding and frame format.

Some people may mistake the VN-Tag for a tunneling tag. In a Cisco style network, the controlling bridge at the end of a row is the master of the fabric extenders that sit within the server racks. The controlling bridge makes all of the VIF assignments and the NIC can use these tags to pass the frames to the correct VMs. Because VN-tagging forms a frame that is not recognized by all different types of networking equipment, it is used locally within the controlling bridge domain and may need to be encapsulated further to pass it on through the data center network. Although these VN-tags provide virtual networking features, they are used to identify VMs instead of tenants. Because of this and other reasons, Cisco was a main contributor in the development of the VXLAN standard which will be described in the next section.

VXLAN

The VXLAN specification was originally proposed by Cisco and VMware and now has support from several other leading companies. It's a technique for tunneling virtual layer 2 networks through layer 3 physical networks in large data centers that

need to support multiple tenants. In order to tunnel these packets, they are encapsulated using special VXLAN tags. Keep in mind that MPLS is a layer 3 protocol and VXLAN frames may be tunneled through a larger MPLS data center network. The IETF formed a special working group that is finalizing this standard. In this section, we will provide more information on this standard including the frame format and how VXLAN encapsulated frames are forwarding through the network. In the next section, we will discuss a competing standard called NVGRE. As of this writing, both the VXLAN and NVGRE specifications are in draft form in the IETF, and the NVGRE specification is less mature. In this chapter, we will provide information as it is available today. A third tunneling protocol called stateless transport tunneling (STT) has been proposed by Nicira who was recently acquired by VMware. We will not cover STT in this chapter and instead focus on the two main protocols backed by major OEMs.

Frame format

Servers within a cloud data center typically present TCP/IP frames, which are encapsulated with layer 2 header information, to the attached network. In many cases, this header also contains a VLAN tag that the tenant may be using to identify different departments within its organization. From the server's or VM's point of view, it is sending and receiving these types of frames through an attached layer 2 network. The trick here is how to support multiple tenants like this in a large L3 data center network. This is where tunneling protocols like VXLAN come into play.

The VXLAN protocol provides these features by encapsulating these frames with a VXLAN tag along with a User Datagram Protocol (UDP) header as shown in Figure 7.5. Here, the original frame containing MAC addresses and a VLAN tag are encapsulated using a VXLAN header and a UDP header. This entire combination is then routed through the layer 3 network using IP addresses. By using unique VXLAN tags for each tenant, each tenant can use its own pool of MAC addresses while the VXLAN protocol provides isolation between these different virtual layer 2 network domains.

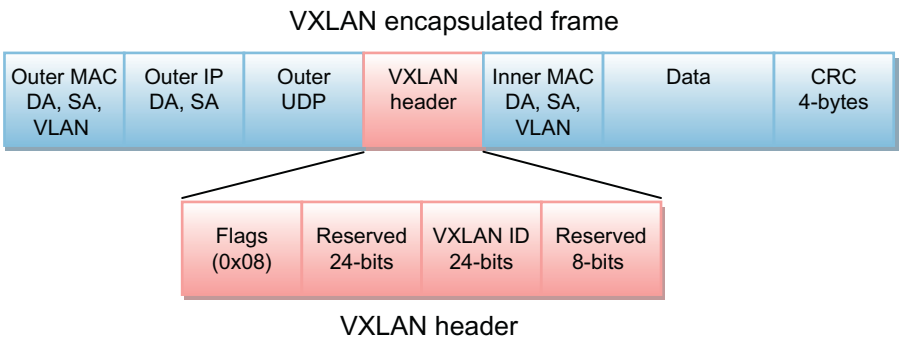

FIGURE 7.5

VXLAN frame format.

UDP is a member of the set of protocols used in the Internet that include TCP. Unlike TCP, which requires acknowledgment frames and can provide error correction, UDP is considered a best-effort protocol with no guarantee of delivery. TCP was designed for the larger internet where connections may be disrupted while packets are being transmitted across long distance telecommunication networks. Within the data center, connections are in a controlled environment, meaning that the likelihood of a dropped packet is much lower. In addition, the original inner TCP/IP header (part of "Data" in Figure 7.5) can provide reliable transmission through error detection and retransmission if needed. Because of this, it doesn't make sense to burden the VXLAN frame with the extra reliability requirements and overhead that is used in TCP. The UDP protocol uses the concept of port numbers. For the VXLAN frame, the source port is provided by the source tunnel endpoint and the destination port is set to a well-known UDP port number.

The point in the network where frame encapsulation and de-encapsulation occurs is known as the VXLAN Tunnel End Point (VTEP). The outer IP addresses are used to forward data through the data center layer 3 network between these VTEPs. These endpoints can be inside the hypervisor, the NIC, or a physical switch, which we will discuss later in this chapter. The VXLAN Network ID (VNI) is a 24-bit value that is inserted in the VXLAN header and can identify up to 16 million unique virtual networks. We will next provide more information on the encapsulation and de-encapsulation process performed by the VTEP.

VTEP encapsulation

The VXLAN Tunnel End Point (VTEP) is the VXLAN encapsulation point and is connected to a traffic source which may be a stand-alone server or virtual machine. For example, the VTEP could be part of the hypervisor in a server platform, part of the network interface device in the server, or part of the attached top of rack (ToR) switch. Figure 7.6 will be used as an example of how a layer 2 unicast frame is encapsulated when sending it from VM2 to VM3 through a VXLAN tunnel.

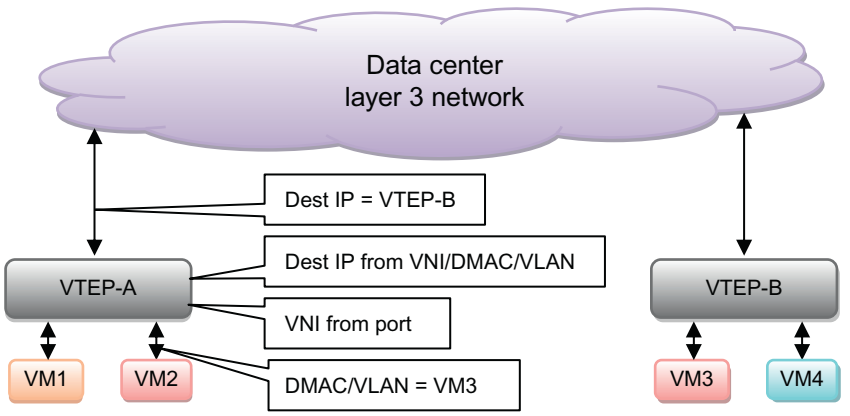

FIGURE 7.6

VTEP encapsulation example.

At the ingress to the network, VM2 sends a frame that may contain a TCP/IP address along with a layer 2 header containing a SMAC address, a DMAC address, and a VLAN tag. The DMAC address is the layer 2 address of VM3 which is part of the same tenant VLAN as VM2. As far as VM2 in concerned, VM3 is part of its local layer 2 network, when in fact it could be in another VM on the other side of the data center.

When the frame is delivered to VTEP-A, the VNI is determined based on information such as which virtual machine the data is coming from. It is assumed that each VM in the network will be assigned to a single VNI and that a given VNI will have all of the tenant's virtual machines associated with it. Once the VNI has been identified, VTEP-A will also examine the inner DMAC/VLAN address and use this along with the VNI to determine that the destination is VTEP-B. The frame is then encapsulated with the VXLAN header containing the VNI, the UDP header, the destination IP address of VTEP-B, and the source IP address of VTEP-A.

If the inner DMAC is an unknown address within this VNI, a MAC address learning process is used similar to what is used within a layer 2 network. To do this, IP multicast addresses are used. Every VTEP associated with a given VNI will join the same IP multicast group. Unknown addresses are flooded to all other associated VTEPs using this multicast IP address in the outer IP destination address field. When a response is received from a destination VTEP, the SMAC from this response frame is used to update the VTEP-A forwarding table, just as it is done in a L2 network. In this way, the environment that the VM is exposed to behaves just like a layer 2 network including address learning.

VTEP de-encapsulation

When the frame arrives at the destination VM, it should appear that the frame arrived from a standard layer 2 network. As shown in Figure 7.7, when the frame arrives at VTEP-B it contains the source IP address from VTEP-A. The correct

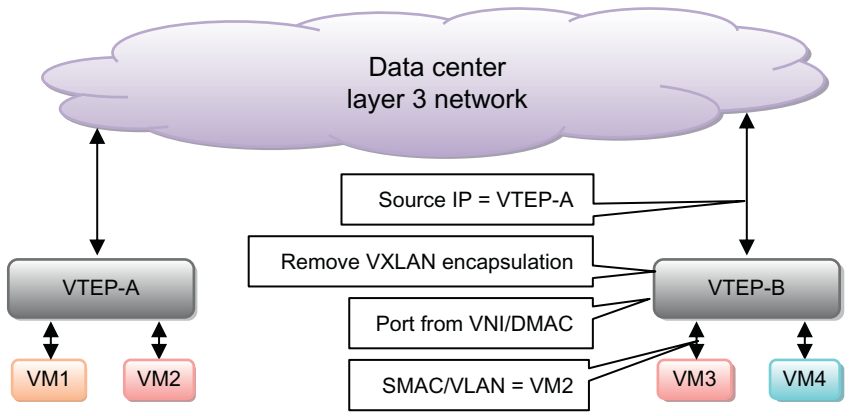

FIGURE 7.7

VTEP de-encapsulation example.

destination VM (VM3 in this example) is determined by examining the VNI and the inner DMAC address. The DMAC is required because there could be two VMs for a given tenant associated with a VTEP endpoint. The VXLAN header, UDP header, and outer IP and L2 fields are then removed from the frame before presenting it to the VM.

For layer 2 MAC address learning, the egress VTEP needs to perform several tasks. When a frame is received, VTEP-B will learn the inner MAC source address and associated source IP address so that it doesn't need to flood response packets later on. In order to receive flooded frames from other VTEPs that have unknown addresses, VTEP-B needs to make sure it has joined the IP multicast group associated with a given VNI. Because a given VTEP may be supporting multiple VMs, each associated with different VNIs, it may need to join several IP multicast groups.

NVGRE

The NVGRE protocol is very similar to VXLAN and was created to solve a similar set of problems that were described in the last section. It is interesting that our industry continues to come up with two competing solutions to the same problem. For example, TRILL versus SPB and iWarp versus RoCE that were described in Chapter 5. Or VEPA versus VN-Tag that was described in Chapter 6. In any case, NVGRE is another IETF standard that is being backed by Microsoft, Intel, HP, and Dell. In this section, we will provide information on the parent Generic Routing Encapsulation (GRE) standard along with an overview of the NVGRE frame format. We will also provide some examples of how frames are tunneled through the network while highlighting the differences between VXLAN and NVGRE. A draft was submitted to the IETF a few weeks before this writing that is a proposal for a tunneling protocol that is a superset of VXLAN and NVGRE called Generic Network Virtualization Encapsulation (GENEVE). It will be interesting to see how this progresses over the next few years.

Generic routing encapsulation

GRE was developed by Cisco as a way to encapsulate a wide variety of different network layer protocols within a generic header that can provide point-to-point links over an IP network. An example application is the secure VPN links that you may use when working at remote locations from your office. When developing a new standard to tunnel layer 2 frames through layer 3 data center networks, Microsoft and others decided to reuse this existing method instead of creating a new header as was done with the VXLAN standard. Although this works well for its intended purpose, there are some limitations which will be described below.

Frame format

The NVGRE frame format shown in Figure 7.8 is very similar to the VXLAN frame format that we described in the last section. Instead of using a UDP header and a VXLAN header, only a GRE header is used, reducing the frame size by a few bytes. The GRE header contains the unique protocol ID (0x6558) for NVGRE frames as well as a 24-bit virtual segment identifier (VSID), that, like VXLAN, can support up to 16M unique tenant subnets. Another difference is that the inner layer 2 header does not contain a VLAN tag, and if one exists, it is removed before the frame is encapsulated. So in this case, the VSID can also be used to segregate multiple virtual network segments for a given tenant.

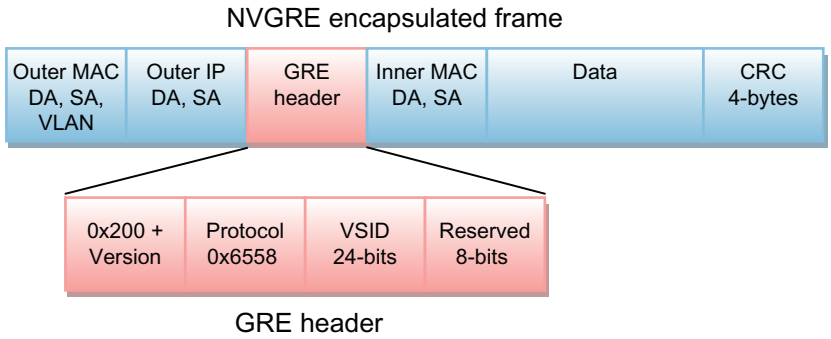

FIGURE 7.8

NVGRE frame format.

NVE encapsulation

The Network Virtual Endpoint (NVE) defined in the NVGRE specification is very similar to the VTEP defined in VXLAN. It is part of the Hyper-V hypervisor from Microsoft and NVGRE is sometimes referred to as Hyper-V virtual switching. Figure 7.9 shows how a layer 2 frame generated from a VM is encapsulated and tunneled through a NVGRE virtual network. In this example, a frame from VM2 has a DMAC/VLAN with the destination address of VM3. As with VXLAN, the VM2 has no idea that the frame is being tunneled, and it thinks that VM3 is attached to its local layer 2 network. Because the NVGRE requires that the NVE remove the inner VLAN tag, the NVE uses the inner VLAN ID plus knowledge of which virtual network the tenant VM belongs to in order to assign a unique VSID. In order to support tenants that want to use multiple VLANs, multiple VSIDs can be supported for a given tenant.

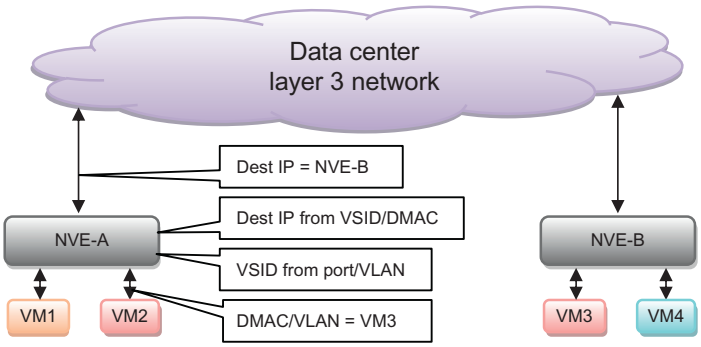

FIGURE 7.9

NVE encapsulation example.

Once the VSID is identified, this plus the inner DMAC address can be used to identify the destination IP address of the NVE associated with VM3. The frame is then encapsulated with a GRE header and the appropriate L2/L3 routing headers in order to tunnel the frame between NVEs. The current version of the NVGRE specification does not define how unicast or multicast address information is disseminated to the NVEs. The specification currently states: *Address acquisition is beyond the scope of this document and can be obtained statically, dynamically, or using stateless address auto-configuration*. This is different from the VXLAN specification that defines IP multicast as the flooding method for MAC address learning, but it doesn't preclude NVGRE from using a similar technique.

NVE de-encapsulation

When a tunneled frame arrives at its destination across the IP network, it is de-encapsulated by another NVE. In the example shown in Figure 7.10, this frame will arrive at NVE-B containing the source address of NVE-A. At this point, NVE-B will

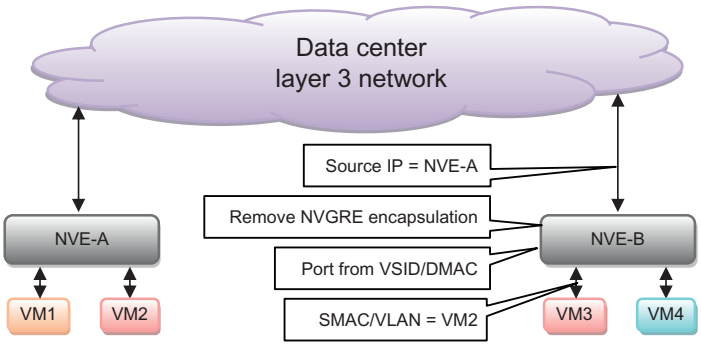

FIGURE 7.10

NVE de-encapsulation example.

use the VSID and DMAC information to determine the destination VM. The DMAC is required because there could be two VMs for a given tenant associated with a NVE. The GRE header along with the outer IP header and outer L2 header are then removed before presenting the original layer 2 frame to VM3.

Although the NVGRE spec requires that the VLAN tag be removed from the original frame, it does not specify how the tenant VLAN information can be tunneled through the layer 3 network. It is assumed that the VSID assigned to a particular tenant VLAN at the tunnel ingress can be used to recreate the VLAN ID and the tunnel egress. It does not suggest how the VLAN priority field is passed through. Maybe this will be covered in a later version of the specification.

TUNNEL LOCATIONS

In the examples above, we made the assumption that the VTEP or NVE tunnel endpoints existed next to the virtual machines in the hypervisor/vSwitch. These are the most common proposals for these new tunneling protocols and it makes sense given that one of the lead authors of the VXLAN spec is VMware and one of the lead authors of the NVGRE spec is Microsoft. But network virtualization tunneling endpoints can also exist at other locations near the edge of the network as shown in Figure 7.11. In this section we will describe three tunneling endpoint locations including the vSwitch, the network interface card, and the top of rack switch.

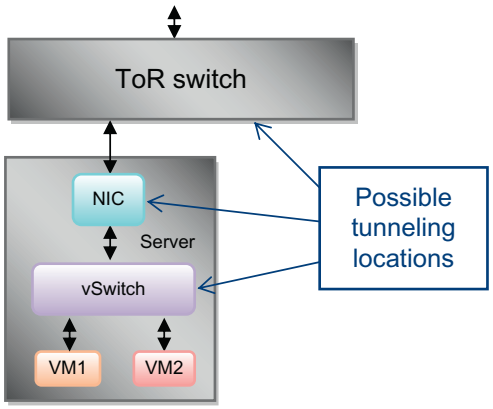

FIGURE 7.11

Tunneling locations in the network edge.

vSwitch

It makes sense to move the tunnel endpoints as close as possible to the virtual machines as this makes tenant identification easier. Today, VXLAN encapsulation and de-encapsulations functions are available in VMware's vSphere hypervisor and NSX software-defined networking products. In addition, Microsoft offers NVGRE encapsulation and de-encapsulation functions in Windows Server 2012. Keep in mind that this type of tunneling consumes CPU resources that must be taken into account when implementing these services.

Another factor that must be considered is the effect tunneling may have on the TCP/IP offload resources in the attached NIC. Today, many NICs have offload features such as large segment offload and TCP checksum offloads that are designed to operate with non-encapsulated packets. For example, large segment offload requires that each new segment generated by the NIC get the same header information. If a NIC is not designed to comprehend the VXLAN or NVGRE headers, it won't know how to replicate them properly. To allow tunneling to work in the hypervisor, these features may need to be turned off, further taxing the CPU resources. Some vendors such as Intel have added large segment offload hardware assist for these tunneling protocols, both improving performance and reducing the CPU workload. There are also cases where technologies like SR-IOV are used to bypass the hypervisor. In these cases, tunneling must be performed in the NIC or ToR switch.

Network interface card

As mentioned above, advanced NIC features such as TCP offloads and SR-IOV can interfere with the ability to provide VXLAN or NVGRE tunneling in the hypervisor. To help this situation, the NIC may provide advanced features that allow them to maintain offload capabilities while at the same time support VXLAN and/or NVGRE tunneling. One example of this is large segment offloads. Many Ethernet networks do not support jumbo frames and have a maximum frame size of around 1500 bytes. If an application wants to send larger chunks of data (for example a video stream), the TCP/IP layer must first divide it up into smaller 1500B frames. This can tax CPU resources and is typically offloaded to the NIC.

The problem is that most NICs cannot deal with a tunnel header added by the hypervisor. They are designed to simply replicate the layer 2 header on each frame segment they create. To solve this problem, NIC vendors have upgraded their large segment offload hardware to replicate the tunnel header along with the layer 2 header so that all segments of a large data flow are tunneled properly through the network. An alternative solution would be to add encapsulation and de-encapsulation resources after the TCP/IP offload functions in the NIC. In addition, if the frame encapsulation and de-encapsulation resources are after the SR-IOV block, frames could be tunneled before reaching the network. But today, few NICs offer this feature, so any data flows that bypass the hypervisor by using SR-IOV must be passed on to the top of rack switch before tunneling into the network.

Top of rack switch

When no hypervisor is used or the hypervisor is bypassed, and the NIC cannot perform tunneling after the SR-IOV block, the ToR switch can act as the tunneling endpoint. Unlike the hypervisor that knows the source VM, in order for the switch to identify the proper VNI or VSID for the tunnel, the switch must use information from the received frame such as the MAC source address from the inner header in order to identify the associated VM. One advantage of using the ToR switch for tunnel endpoints is that traffic intended for other servers within the same rack do not need to be tunneled and can be simply forwarded using layer 2 information, reducing the table size requirements when a large number of VMs for one tenant are in the same rack.

LOAD BALANCING

In a previous chapter, we mentioned that Ethernet uses the Spanning Tree Protocol in order to avoid loops in the layer 2 network. This can leave many unused links in the network, wasting this potential bandwidth. An advantage of using virtual networks is that tunnels use layer 3 forwarding mechanisms that do not have this restriction. In this section, we will describe the technique of using hash-based algorithms for tunnel selection and also a common load distribution mechanism, called equal cost multipath routing (ECMP), used in IP networks. In addition, we will describe how VXLAN and NVGRE can take advantage of these mechanisms along with any limitations they impose.

Hash-based algorithms

Hash algorithms have been around for decades and are used for applications such as table lookups. For example, you can use a person's name and address as a hash key used by a hash algorithm. The output of the hash algorithm will be a pointer into a table where the person's information will be stored. Later, when you want to retrieve a given persons information, you don't need to scan the entire table, but simply use this same hash key and algorithm to obtain a pointer to the person's information in the table. This can greatly simplify the lookup process.

In a switch or router, a similar technique can be used to select a path for a packet to take based on information in the packet header as shown in Figure 7.12. Let's assume that we have a simple switch with one input and four outputs as shown in the figure. For each frame coming into the switch, a header field or multiple header fields can be chosen to form a hash key. Instead of pointing to an entry in a table, the hash algorithm is used to select one of four possible output ports. If a proper hash algorithm is chosen, and frames arrive with fairly random header fields, traffic will be uniformly distributed across the output ports. Another nice feature is that frames belonging to the same flow (for example the same video stream) will always be hashed to the same output, maintaining in-order delivery. The key in networking applications is to find a hash algorithm that provides uniform distribution for common traffic patterns found in the network. As traffic patterns change over time, the load distribution may become less uniform causing congestion points in the network.

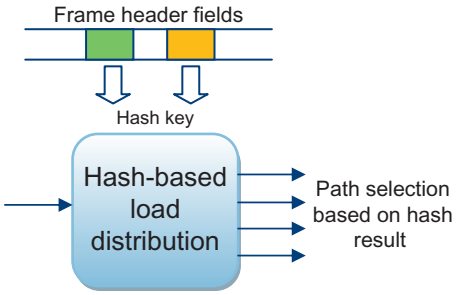

FIGURE 7.12

Load distribution using hashing.

Equal cost multipath routing

ECMP routing is a technique used in IP networks to increase the number of paths and overall bandwidth through the network. In multitenant environments where virtual networks are used, techniques like ECMP can be used to improve tunnel bandwidth and throughput. An example ECMP distribution is shown in Figure 7.13 where the source VTEP or NVE adds the tunneling tag which has information used to help distribute the tunnel traffic across multiple paths in the IP network. What is not shown in Figure 7.13 is that each IP router may further distribute the traffic across additional paths by using the tunneling tag as part of a hash key.

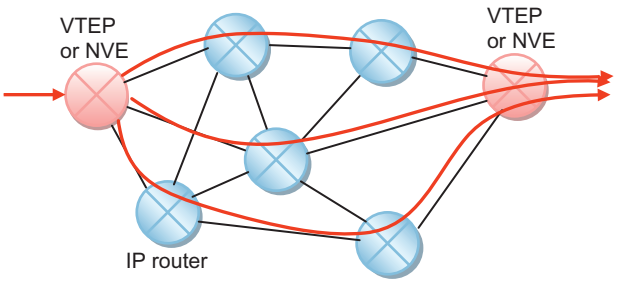

FIGURE 7.13

Equal cost multipath routing from a VTEP or NVE.

When using VXLAN tunneling, the VTEP can use the inner MAC header fields to form a hash key which the hash algorithm then uses to select one of several ECMP paths through the IP network. Typically, the hash key will be used as a random UDP source port number, which is inserted into the frame by the VTEP encapsulation function. Switches and routers in the data center network are designed to use various header fields such as the UDP source port number to form ECMP hash keys. Therefore, the UDP source port field can be used by these switches and routers to select ECMP paths through the network.

As we mentioned earlier, the NVGRE spec is less mature than the VXLAN spec and one issue that has been identified by some people in the industry is that NVGRE does not have a TCP or UDP header. This means that traditional IP routers do not have this information for their ECMP hash key. Some newer IP routers can use the GRE header as a source for their hash key allowing the ability to improve ECMP hashing across the network. The NVGRE draft specification recommends using the full 32-bit GRE field for ECMP hashing. One way to do this is to have the ingress NVE place a random 8-bit value in the reserved field next to the 24-bit VSID field to improve distribution.

REVIEW

In this chapter, we described multitenant cloud data centers and how they can take advantage of virtualized networks. We described traditional methods of tunneling data through carrier networks and challenges that they present in virtualized data center networks. Next, we provided some details behind VXLAN and NVGRE, the two most popular emerging standards for network virtualization in multitenant data centers. In addition, we provided an overview of tunneling locations and also provided details on how traffic distribution can be accomplished using these tunneling protocols. In the next chapter we will provide information on storage networks and how storage traffic is forwarded in cloud data center networks.