there is neither too much change nor a single point of failure. We rank it as medium with respect to performance and scale, since legacy networks have proven that they can handle sizable networks but are showing inability to scale to the size of mega data centers. SDN-via-APIs solutions support neither deep packet inspection nor stateful flow awareness. Nothing in these solutions directly addresses the problems of MAC forwarding table or VLAN ID exhaustion.

So, in summary, SDN via APIs provides a mechanism for moving toward a controller-based networking model. Additionally, these APIs can help alleviate some of the issues raised by SDN around the need to have a better means of automating the changing of network configurations in order to attempt to keep pace with what is possible in virtualized server environments. However, SDN via APIs falls short of meeting many other goals of SDN.

6.3 SDN via Hypervisor-Based Overlays

In Chapter 4 we introduced SDN via hypervisor-based overlay networks. This hypervisor-based overlay technology creates a completely new virtual network infrastructure that runs independently on top of the underlying physical network, as depicted in Figure 4.8. In that diagram we saw that the overlay networks exist in an abstract form *above* the actual physical network below. The overlay networks can be created without requiring reconfiguration of the underlying physical network, which is independent of the overlay virtual topology.

With these overlays it is possible to create networks that are separate and independent of each other, even when VMs are running on the same server. As shown in Figure 4.8, a single physical server hosting multiple VMs can have each VM be a member of a separate virtual network. This VM can communicate with other VMs in its virtual network, but it usually does not cross boundaries and talk to VMs that are part of a different virtual network. When it is desirable to have VMs in different networks communicate, they do so using a virtual router between them.

It is important to notice in the diagram that traffic moves from VM to VM without any dependence on the underlying physical network other than its basic operation. The physical network can use any technology and can be either a layer two or a layer three network. The only matter of importance is that the virtual switches associated with the hypervisor at each endpoint can talk to each other.

What is actually happening is that the virtual switches establish communication tunnels among themselves using general IP addressing. As packets cross the physical infrastructure, as far as the virtual network is concerned they are passed directly from one virtual switch to another via virtual links. These virtual links are the tunnels we just described. The virtual ports of these virtual switches correspond to the *virtual tunnel endpoints* (VTEPs) defined in Section 4.6.2. The actual payload of the packets between these vSwitches is the original layer two frame being sent between the VMs. In Chapter 4 we defined this as *MAC-in-IP* encapsulation, which was depicted graphically in Figure 4.9. We provide a detailed discussion about MAC-in-IP encapsulation in Chapter 7.

Much as we did for SDN via APIs, we restrict our definition of SDN via hypervisor-based overlays to those solutions that utilize a centralized controller. We acknowledge that there are some SDN overlay solutions that do not use a centralized controller (e.g., MidoNet, which has MidoNet agents located in individual hypervisors), but we consider those to be exceptions that cloud the argument that this alternative is, generally speaking, a controller-based approach.

6.3.1 Overlay Controller

By our definition, SDN via hypervisor-based overlays utilizes a central controller, as do the other SDN alternatives discussed thus far. The central controller keeps track of hosts and edge switches. The overlay controller has knowledge of all hosts in its domain, along with networking information about each of those hosts, specifically their IP and MAC addresses. These hosts will likely be virtual machines in data center networks. The controller must also be aware of the edge switches that are responsible for each host. Thus, there will be a mapping between each host and its adjacent edge switch. These edge switches will most often be virtual switches associated with a hypervisor resident in a physical server and attaching the physical server's virtual machines to the network.

In an overlay environment, these edge switches act as the endpoints for the tunnels that carry the traffic across the top of the physical network. These endpoints are the VTEPs mentioned above. The hosts themselves are unaware that anything other than normal layer two Ethernet forwarding is being used to transport packets throughout the network. They are unaware of the tunnels and operate just as though there were no overlays involved whatsoever. It is the responsibility of the overlay controller to keep track of all hosts and their connecting VTEPs so that the hosts need not concern themselves with network details.

6.3.2 Overlay Operation

The general sequence of events involved with sending a packet from host A to a remote host B in an overlay network is as follows:

1. Host A wants to send payload P to host B.
2. Host A discovers the IP address and MAC address for host B using normal methods (DNS and ARP). Host A uses its ARP broadcast mechanism to resolve host B's MAC address, but the means by which this layer two broadcast is translated to the tunneling system varies. The original VXLAN tries to map layer two broadcasts directly to the overlay model by using IP multicast to flood the layer two broadcasts. This allows the use of Ethernet-style *MAC address learning* to map between virtual network MAC addresses and the virtual network IP addresses. There are also proprietary control plane implementations whereby the tunnel endpoints exchange the VM-MAC address to VTEP-IP address mapping information. Another approach is to use MP-BGP to pass MPLS VPN membership information between controllers where MPLS is used as the tunneling mechanism. In general, this learning of the virtual MAC addresses across the virtual network is the area where virtual network overlay solutions differ most. We provide this sampling of some alternative approaches only to highlight these differences and do not attempt an authoritative review of the different approaches.
3. Host A constructs the appropriate packet, including MAC and IP addresses, and passes it upstream to the local switch for forwarding.
4. The local switch takes the incoming packet from host A, looks up the destination VTEP to which host B is attached, and constructs the encapsulating packet as follows:
   - The outer destination IP address is the destination VTEP and the outer source IP address is the local VTEP.
   - The entire layer two frame that originated from host A is encapsulated within the IP payload of the new packet.
   - The encapsulating packet is sent to the destination VTEP.

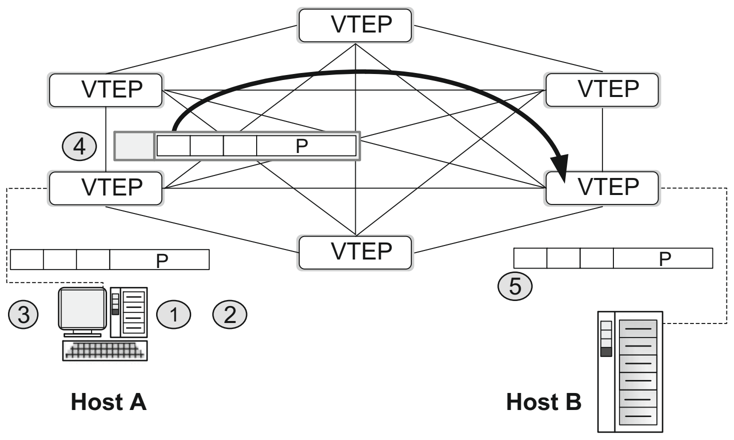

FIGURE 6.11

Overlay operation.

5. The destination VTEP receives the packet, strips off the outer encapsulation information, and forwards the original frame to Host B.

Figure 6.11 shows these five steps. From the perspective of the two hosts, the frame transmitted from one to the other is the original frame constructed by the originating host. However, as the packet traverses the physical network, it has been encapsulated by the VTEP and passed directly from one VTEP to the other, where it is finally decapsulated and presented to the destination host.

6.3.3 Examples of SDN via Hypervisor-Based Overlays

Prior to Nicira's acquisition by VMware in 2012, Nicira's *Network Virtualization Platform* (NVP) was a very popular SDN via hypervisor-based overlays offering in the market. NVP has been widely deployed in large data centers. It permits virtual networks and services to be created independently of the physical network infrastructure below. Since the acquisition, NVP is now bundled with VMware's *vCloud Network and Security* (vCNS) and is marketed as VMware NSX [10],[4] where it continues to enjoy considerable market success. The NVP system includes an open source virtual switch, *Open vSwitch*, (OVS) that works with the hypervisors in the NVP architecture. Open vSwitch has become the most popular open source virtual switch implementation, even outside of NVP implementations. Interestingly, NVP uses OpenFlow as the southbound interface from its controller to the OVS switches that form its overlay network. Thus, it is both an SDN via hypervisor-based overlays *and* an Open SDN implementation.

Juniper's JunosV Contrail [17], however, does not use OpenFlow. Contrail is a controller-based virtual overlay approach to network virtualization. It uses the *Extensible Messaging and Presence Protocol* (XMPP) to program the control plane of the virtual switches in its overlay networks. Because

[4]Since we focus on the NVP component in this work, we usually refer to that component by its original, pre-acquisition name rather than NSX.

the southbound API is XMPP, it is indeed a standards-based approach. Since the control plane still resides on those virtual switches and XMPP is not directly programming the data plane, we do not consider it Open SDN. It is clearly an SDN via hypervisor-based overlays approach, though.

Another important SDN via hypervisor-based overlays offering is IBM's *Software Defined Network for Virtual Environments* (SDN VE) [11]. SDN VE is based on IBM's established overlay technology, *Distributed Overlay Virtual Ethernet* (DOVE).

6.3.4 Ranking SDN via Hypervisor-Based Overlays

In Table 6.1 we include our ranking of SDN via hypervisor-based overlays versus the alternatives. SDN via hypervisor-based overlays is founded on the concepts of network automation and virtualization and, thus, scores high in those categories. With respect to openness, this depends very much on the particular implementation. NVP, for example, uses OpenFlow, OVSDB, and OVS, all open technologies. Other vendors use proprietary controller-based strategies. There are both closed and open subcategories within this alternative, so we have to rate it with a dual ranking $[Medium, High]$ against this criterion, depending on whether or not the overlay is implemented using Open SDN. Because we restricted our definition of SDN via hypervisor-based overlays to controller-based strategies, it ranks high in this category. With respect to device simplification, the overlay strategy does allow the physical network to be implemented by relatively simple IP layer three switches, so we give this a ranking of medium with respect to device simplification.

A sidenote is called for here. The SDN via hypervisor-based overlays is often touted as delivering network virtualization on top of the existing physical network. Taken at face value, this does not force any device simplification, it merely allows it. A customer may be very pleased to be able to keep his very complex switches if the alternative means discarding that investment in order to migrate to simple switches. This actually holds true for OpenFlow as well. Many vendors are adding the OpenFlow protocol to existing complex switches, resulting in hybrid switches. Thus, OpenFlow itself does not mandate simple devices either, but it does allow them.

Plane separation is more complicated. In the sense that the topology of the physical network has been abstracted away and the virtual network simply layers tunnels on top of this abstraction, the control plane for the virtual network has indeed been separated. The physical switches themselves, however, still implement locally their traditional physical network control plane, so we give this a score of medium.

The criterion of too much change is also difficult to rank here. On one hand, the same physical network infrastructure can usually remain in use, so no forklift upgrade of equipment is necessarily required. On the other hand, converting the network administrators' mindset to thinking of virtual networks is a major transformation, and it would be wrong to underestimate the amount of change this represents. Thus, we rank this medium.

Since SDN via hypervisor-based overlays may or may not be based on the centralized controller concept, it gets an N/A ranking in the single-point-of-failure category. Similarly, we feel that it deserves the same medium ranking in performance and scale as Open SDN. It fares better than Open SDN with respect to deep packet inspection and stateful flow awareness because the virtual switches are free to implement it under the overlay paradigm. They may map different flows between the same hosts to different tunnels. This freedom derives from the fact that these virtual switches may implement any propriety feature, such as deep packet inspection, and are not restricted by a standard like OpenFlow.