Virtual Networks

Interestingly, data center network designers face a challenge of balancing between two conflicting goals:

- Universal connectivity
- Safe, isolated communication

Universal connectivity: It should be obvious that systems across the entire data center must be able to communicate with one another.

When they run, apps communicate with databases, the apps running on other servers, storage facilities, and, possibly, with computers on the global Internet.

Furthermore, to give a provider freedom to place VMs and containers on arbitrary physical servers, the network must guarantee *universal connectivity* — a pair of apps must be able to communicate independent of the physical servers on which they run.

The leaf-spine network architecture in each server connects to a leaf switch (i.e., a Top-of-Rack switch), and the leaf switches connect to spines. Clearly, the architecture provides universal connectivity. In fact, multiple physical paths exist between any two servers.

- *Safe, isolated communication*. Although universal connectivity across a data center
  is required, a customer of a multi-tenant data center also demands a guarantee that outsiders will not have access to the customer's systems or communication. That is, a customer needs assurance that their computing systems are isolated from other tenants and
  their communication is safe. Ideally, each customer would like a separate network that
  only connects the customer's VMs and containers.

Virtual Networks, Overlays, And Underlays

A cloud service with thousands of tenants makes separate physical networks impractical.

The answer lies in an approach known as *network virtualization*.

The basic idea is straightforward: build a network with universal connectivity, and then configure the network switches to act as if they control a set of independent networks.

That is, a provider imagines a set of independent networks, and configures each switch to keep traffic sent across one of the imagined networks separate from traffic on other imagined networks.

To emphasize that each of the imagined networks is only achieved by restricting the set of VMs and containers that receive a given packet, we say that the configuration creates a set of *virtual networks*.

Conceptually, each virtual network links a tenant's virtual machines and containers. Figure 7.1 illustrates the idea by showing two virtual networks that each link four VMs.
Keep in mind that the two networks in the figure are fiction.
In reality, each VM runs on a server in a rack, and racks are connected by spine switches.
Suppose, for example,
that VM 2 and VM 5 run on servers in separate racks.
The network path between them
includes two leaf switches and at least a spine switch.
We use the term *overlay network* to refer to a virtual network that does not actually exist but which in effect has been created by configuring switches to restrict communication.
We use the term *underlay network* to refer to the underlying physical network that provides connections among entities in a virtual network.

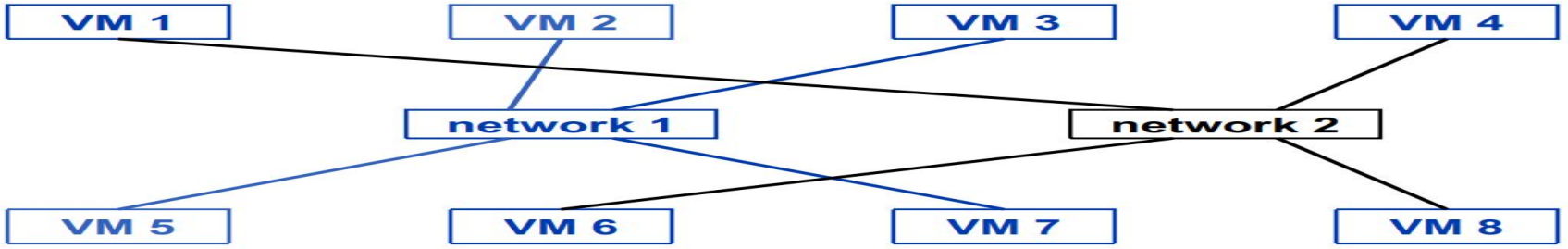

Figure 7.1 Illustration of two virtual networks that each connect four VMs. Each virtual network forms an *overlay*.

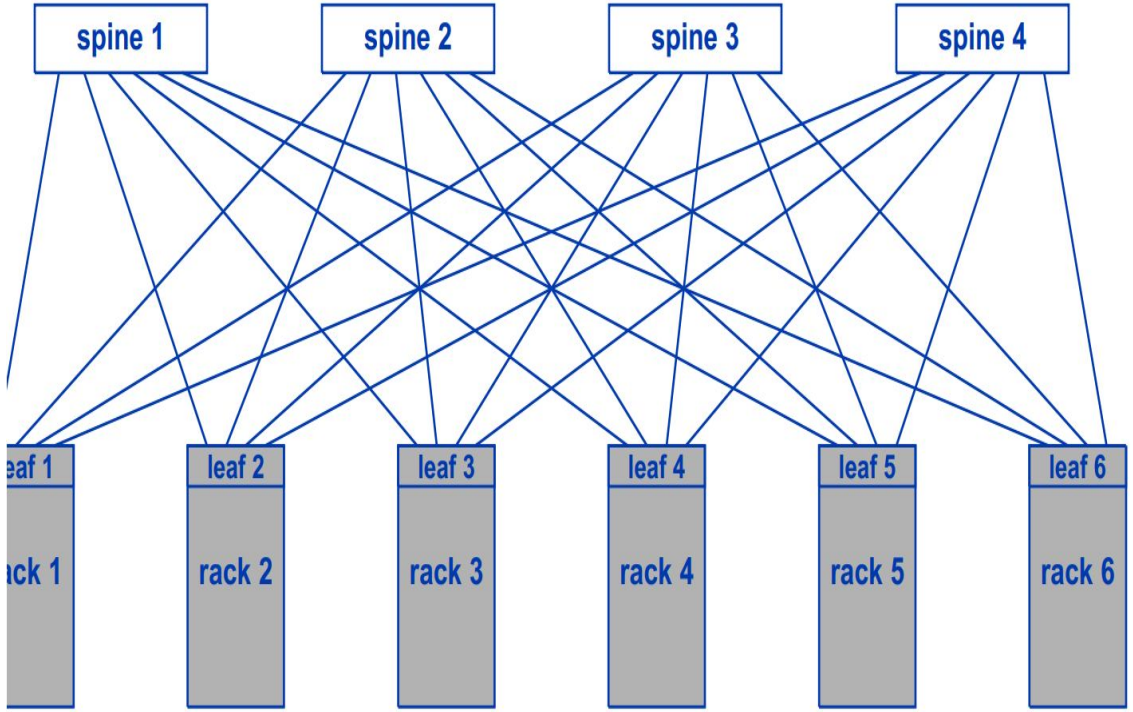

Figure 4.5 An example leaf-spine network with a leaf (i.e., a ToR switch) in each rack connecting to four spine switches.

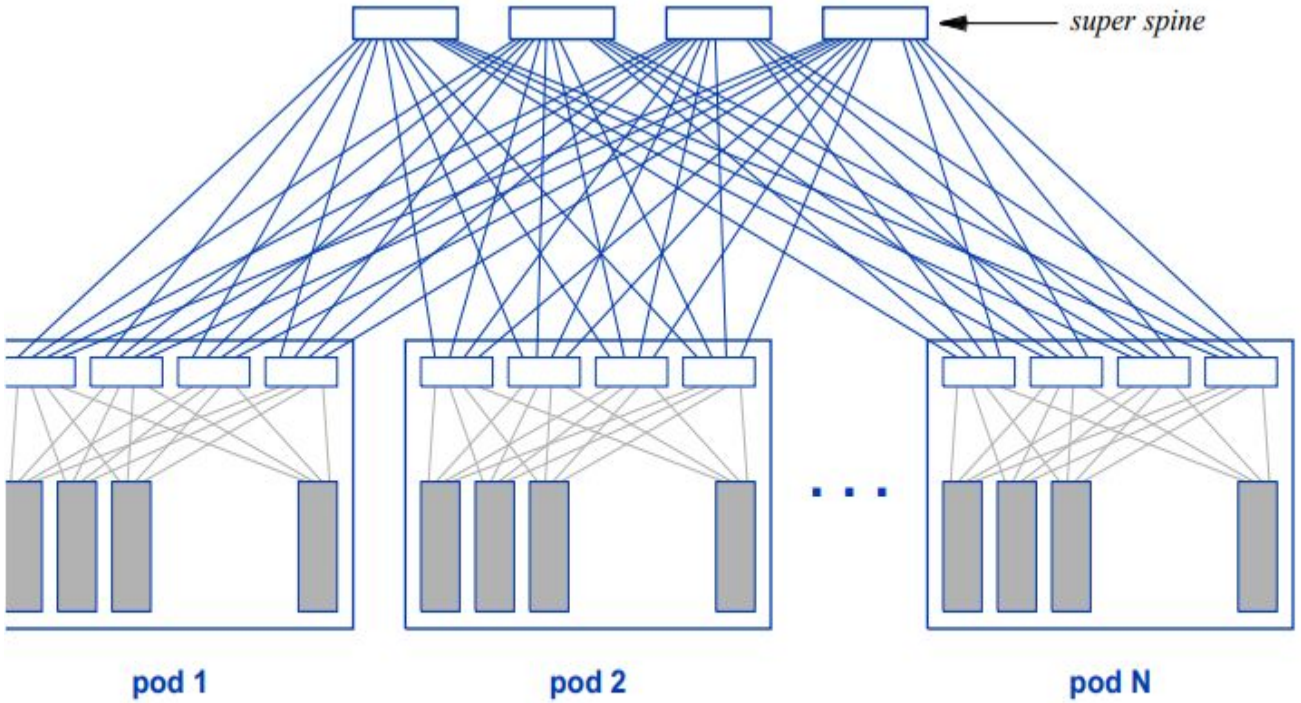

Figure 4.7 Illustration of a super spine configuration in which each pod has a leaf-spine network, and each spine switch connects to each of the super spine switches.

Virtual Local Area Networks (VLANs)

Cloud providers have used a variety of network virtualization technologies.

One of the earliest technologies is known by the name *Virtual Local Area Network* (*VLAN*). VLANs are part of the Ethernet standard, and the Ethernet switches used in data centers support the use of VLANs.
A traditional switch without VLANs forms a single network.

A set of computers connect to ports on the switch, and all the computers can communicate.

When a network administrator uses a VLAN switch, the administrator assigns each port on the switch a small integer known as the port's *VLAN tag*.

If a port has VLAN tag X, we say that the computer connected to the port is "on VLAN X."

VLAN technology makes the switch hardware behave like a set of smaller switches.

Without VLAN technology, any time a computer broadcasts a packet, all computers attached to the switch receive a copy.

With VLAN technology enabled, when a computer on VLAN X broadcasts a packet, only computers on VLAN X receive a copy.
An administrator can configure multiple switches that all use VLAN tags consistently.

Virtual Local Area Networks (VLANs)

When a packet arrives, hardware adds the port's VLAN tag to the packet, and the tag remains in place when the packet is sent to another switch.

It may seem that VLAN technology suffices for a data center.

Indeed, smaller data centers have used VLANs.

However, two limits prevent the technology from handling large cloud data centers.

First, each VLAN tag consists of twelve bits, which limits the technology to 4096 VLANs.

Over four thousand VLANs is sufficient for a single organization because each department can be assigned a separate VLAN number.

In a large data center, however, if each tenant is assigned a dozen VLAN tags, the data center is limited to 341 possible tenants.

Furthermore, the VLAN scheme assigns VLAN tags to ports on switches, not to VMs or containers.

Finally, when they use VLANs, switches run a *Spanning Tree Protocol* that does not scale to large data centers

Scaling VLANs to a Data Center with VXLAN

Network switches are designed to send arbitrary packets among computers, including Internet packets and other types.

The ability to communicate over the Internet has become so important that data centers now use Internet packets almost exclusively.

The addresses used on Internet packets are known as *IP addresses* (*Internet Protocol Addresses*).

Switches throughout each data center are configured to examine IP addresses and forward packets to their correct destination, either within the data center or on the global Internet.

Engineers observed that Internet packets could be used to extend VLAN technology to a scale sufficient for the largest data centers.

They devised a technology known as *Virtual Extensible LAN* (*VXLAN*) that many data centers use.

The technology requires each switch to include VXLAN software, and requires the network administrator to configure special routing protocols so the VXLAN system can learn the locations of computers in the data center.

Once it has been configured, VXLAN can provide the equivalent of more than sixteen million virtual networks — enough for even the largest cloud providers.

VXLAN uses multicast technology to make delivery of packets efficient.

Scaling VLANs to a Data Center with VXLAN

To understand how it works, imagine a dozen VMs all on the same VLAN.

Suppose three VMs are running on servers on one side of the data center and nine are running on servers on the other side of the data center.

If one of the three broadcasts a packet, multicast sends copies to each of them locally.

Instead of sending nine individual copies across the data center to each of the other nine VMs, however, VXLAN sends a single copy across the data center which is then distributed to the nine recipients.

The use of multicast reduces the amount of traffic VXLAN imposes on the data center network, thereby allowing the technology to scale.
The VXLAN technology reverses the usual network paradigm.

A conventional network uses Ethernet packets to carry Internet packets.

That is, each Internet packet is placed inside an Ethernet packet for transfer from a computer to a switch or from one switch to another.

We use the term *encapsulation* to refer to the idea.

VXLAN adds an interesting twist: once an Ethernet packet has been created, VXLAN places the entire Ethernet packet inside an Internet packet for transfer.

The surprise is that to send the resulting Internet packet, VXLAN places it in an “outer” Ethernet packet .

The extra encapsulation may seem unnecessary, but the effect is that VXLAN can scale VLAN technology to large data centers

A Virtual Network Switch within a Server

The use of VMs and containers complicates data center networking in two ways.
First, unlike a traditional network that assigns an IP address to each physical computer,
a data center usually assigns a separate IP address to each virtual machine.

Thus, when multiple VMs run on the same physical server, multiple addresses will be assigned to the server.

Second, if two VMs in the same server communicate, packets must be forwarded from one to the other (provided the forwarding rules permit such communication).

How can a hypervisor forward packets among the VMs it has created? One answer
lies in a *virtual network switch*, such as *Open vSwitch*.

The idea is straightforward: use a piece of software that acts like a conventional network switch. Use the physical network connection on the server to connect the virtual switch to the data center network.

Connect each VM to the virtual switch, and allow the VM to send and receive packets,
just as if the VM connected to a real network switch.

More important, arrange for the virtual switch to use standard configuration software, and configure the virtual switch to follow the same rules for packet forwarding as other data center switches (e.g., prevent packets from a VM owned by one tenant from flowing to a VM owned by another).

Network Address Translation (NAT)

- Recall that container technology allows an app to run in a separate environment, isolated and protected from other containers.
- The question arises, does each container have its own IP address, or do containers all share the IP address of the host OS?
- The answer is that container technology supports three possibilities:
- A container can clone the host's IP address
- A container can receive a new IP address
- A container can use address translation

A container can clone the host's IP address.

If a container uses the same IP address as the host OS, we say that the container has *cloned* the address.

Using the same address as the host OS means the container's network use may conflict with the use by
normal apps or other containers that have cloned the address.

For example, if two containers clone the host's IP address and both attempt to run a web server on port 80, only
the first to request the port will succeed. Consequently, cloning is seldom used in a cloud data center.

A container can receive a new IP address.

Each container can be assigned a unique IP address, and the host operating system can use a virtual switch to provide
connectivity.

A container can use address translation.

Address translation was invented and widely deployed before cloud computing arose.

In fact, many individuals have used NAT technology either from a *wireless router* in their residence or a Wi-Fi hot spot, such as those in coffee shops and hotels.

Known by the acronym *NAT* (*Network Address Translation*), the technology allows multiple computers to share a single IP address.

When used with containers, NAT software runs in the host operating system. When a container that uses NAT begins execution, the container requests an IP address, and the NAT software responds to the request by assigning an IP address from a set of reserved, *private* IP addresses that cannot be used on the Internet.

How can a container with a private address communicate with computers on the Internet?

The NAT software handles communication by intercepting each outgoing packet and replacing the private IP address with the IP address of the host OS. NAT then sends the packet. When a reply comes back, NAT replaces the host's address with the private address assigned to the container, and forwards the reply to the container.

NAT allows multiple containers to share the host's IP address safely analogous to the way a wireless router or Wi-Fi hot spot allows multiple devices to share an IP address.

Managing Virtualization and Mobility

Managing a conventional network is relatively straightforward because the network and the devices connected to the network remain relatively stable. Of course, the owner must configure each switch by specifying a set of rules the switch will use to process packets.

Once a configuration has been entered, however, the network will perform correctly until a piece of equipment fails.

Routing software can handle some failures automatically by changing the rules in switches to send packets along alternative paths that avoid the equipment that has failed.

Configuring and managing the network in a cloud data center poses a complex challenge for three reasons:
Complex interconnections among switches
Multiple levels of virtualization
Arbitrary placement of addressable entities and migration

Complex interconnections among switches. To understand the complexity of a physical network. Imagine configuring each switch to have correct forwarding rules for destinations on the global Internet as well as for each possible destination in the data center.

Consider that the configuration in a switch must specify all equal-cost paths for a destination, allowing ECMP hardware to balance the load across all the paths.

Multiple levels of virtualization.

Consider the configuration needed for VXLAN technology.

Each switch in the data center must be configured with an IP address, and IP forwarding must be working correctly before VXLAN can be added.

Furthermore, IP multicast must also be configured, and operating correctly because VXLAN uses IP multicast.

In addition, VXLAN requires a manager to configure routing protocols that propagate address information across the entire data center.

Arbitrary placement of addressable entities and migration.

A provider can place addressable entities — VMs and containers — on arbitrary physical servers.

As a qconsequence, the IP addresses that belong to a given tenant may be spread across the data center.

More important, whatever system is used to connect all the virtual entities owned by a tenant must accommodate VM migration, and must update forwarding over the network to allow them to reach one another after a migration.

Automated Network Configuration And Operation

- How can thousands of switches arranged in a complex interconnection be configured correctly and efficiently? How can the forwarding information be updated when a
VM moves? The answers lie in technologies that automate the configuration and operation of networks. Examples include:
d Spanning Tree Protocol
d Standard routing protocols
Spanning Tree Protocol. The Ethernet network technology used in data centers allows a sender to broadcast a packet. When a set of switches are connected in a cycle,
broadcast causes a potential problem because one switch forwards a copy to another,

- which forwards to another, and eventually the original switch receives a copy, causing the cycle to repeat forever. Unfortunately, the leaf-spine network architecture used in data centers contains many cycles. The *Spanning Tree Protocol* (*STP*), which runs when a switch boots, solves the problem by detecting cycles and setting up rules to prevent packets from cycling forever.
Standard routing protocols. Like most networks, data center networks employ standard routing protocols that propagate routing information automatically. The protocols, including *OSPF* (*Open Shortest Path First*) and *BGP* (*Border Gateway Protocol*),
learn about possible destinations inside the data center and on the global Internet, compute a shortest path to each destination, and install forwarding rules in switches to send
packets along the shortest paths. More important, routing protocols monitor networks,
and when they detect that a switch or link has failed, automatically change forwarding
to route packets around the failure.

Software Defined Networking

- A technology for automated network mangement known as *Software Defined Networking* (*SDN*) stands out as especially important for data centers. SDN allows a manager to specify high-level policies†, and uses a computer program to configure and monitor network switches according to the policies. That is, instead of relying on humans, a data center owner can use software to master the complexity and handle the necessary low-level details. Software can accommodate a large data center, can handle multiple levels of virtualization, and can update forwarding rules when VMs migrate. The SDN approach uses a dedicated computer to run SDN software. The computer runs a conventional operating system, typically Linux, an *SDN controller* app, and a management app (or apps). A management app can use policies to choose how to forward packets. The controller forms a logical connection to a set of switches and communicates appropriate forwarding rules to each switch. Figure 7.2 illustrates an SDN controller communicating with six switches.

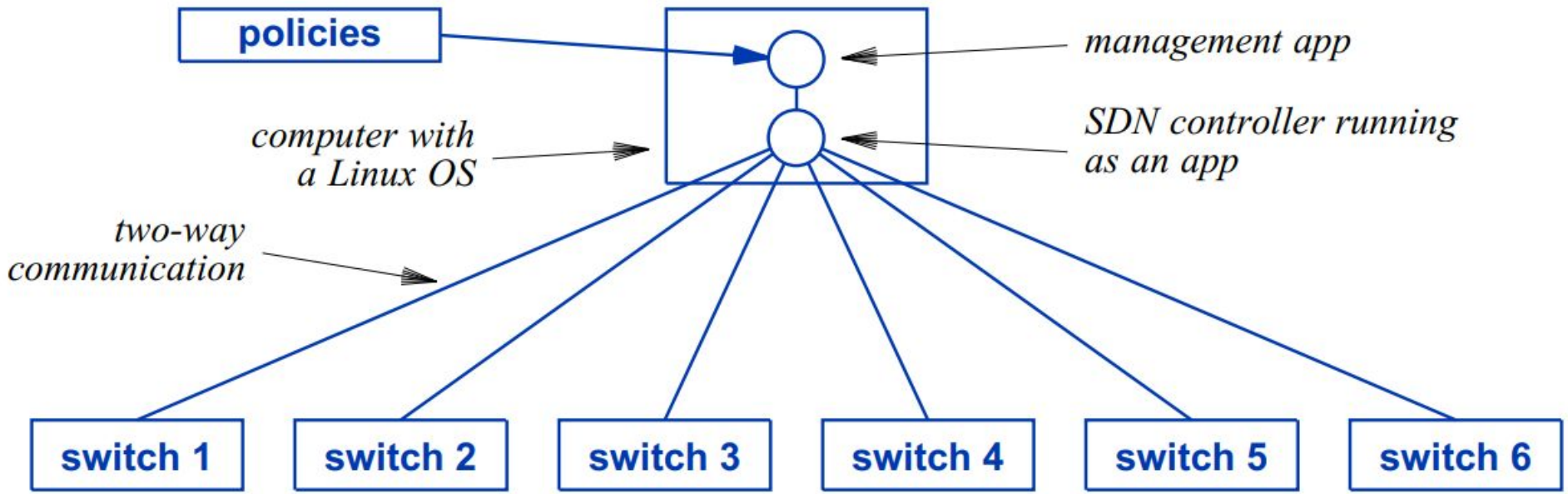

Figure 7.2 An illustration of the communication between a management app, an SDN controller, and a set of network switches being managed.

- As the figure shows, the logical connections between an SDN controller and each switch employ bidirectional communication that allows data to flow in either direction. In addition to the controller sending configuration to the switch, the controller can monitor the status of the switch itself and the links to other switches. For example, SDN can configure a switch to inform the controller when events occur, such as link failure or recovery. Furthermore, a switch can send specified packets to the controller. Typically, a controller configures each switch to send the controller any packet that cannot be handled by its forwarding rules. Thus, the controller will be informed of exceptions, which can include packets from a new application or packets sent as an attack on the data center.

The OpenFlow Protocol

- A key component of SDN technology, the *OpenFlow* protocol standard defines the communication available to an SDN controller. That is, OpenFlow specifies both the form and meaning of messages that can pass between a controller and a switch. To use SDN, a switch needs an OpenFlow module; the switches used in data centers include such support.
  How does OpenFlow control packet forwarding? The basic idea is straightforward: a controller installs a set of forwarding rules in each switch. Each rule describes a particular type of packet and specifies the output port over which the packet should be sent. When a packet arrives at the switch, the switch hardware checks each of the rules, finds one that matches the packet, and sends the packet over the specified port.
  A forwarding rule uses items in a packet header to decide where the packet should be sent. For example, a rule can examine fields that specify the application being used, directing packets that carry World Wide Web traffic out one port, packets that carry database traffic out another port, and all other packets out a third port. Alternatively a forwarding rule can examine the packet's destination, directing packets destined for the global Internet out one port and packets destined for servers inside the data center out another port. A forwarding rule can use the sender's address to direct traffic from some senders along one path and traffic from other senders along another path (e.g., to give some customers priority). Figure 7.3 lists a few of the header fields that a switch can examine when making a forwarding decision.

Field	Meaning
VLAN tag	The VLAN to which the packet belongs
IP source	The sending computer
IP destination	The ultimate destination
TCP source	The type of the sending application
TCP destination	The type of the receiving application

Figure 7.3 Examples of header fields that a controller can use in a forwarding rule.

Programmable Networks

- As described above, the first generation of SDN software uses static forwarding rules, where each rule specifies a set of header fields in a packet and an output port for packets that match the rule. The chief limitation of static forwarding rules lies in the overhead needed to handle exceptions: a packet that does not match any rule must be sent to the controller†. When it receives an exception, the controller examines the policies, chooses an action to use (which may include dropping the packet), installs new forwarding rules in the switches, and then sends the packet back to the switch for processing according to the new rules.
  A second generation of SDN has been designed to reduce the need for a controller to handle each exception by placing a computer program in each switch; we use the term *programmable network* to describe the idea. Instead of a conventional language, the second generation uses a special programming language named *P4*. The point is: