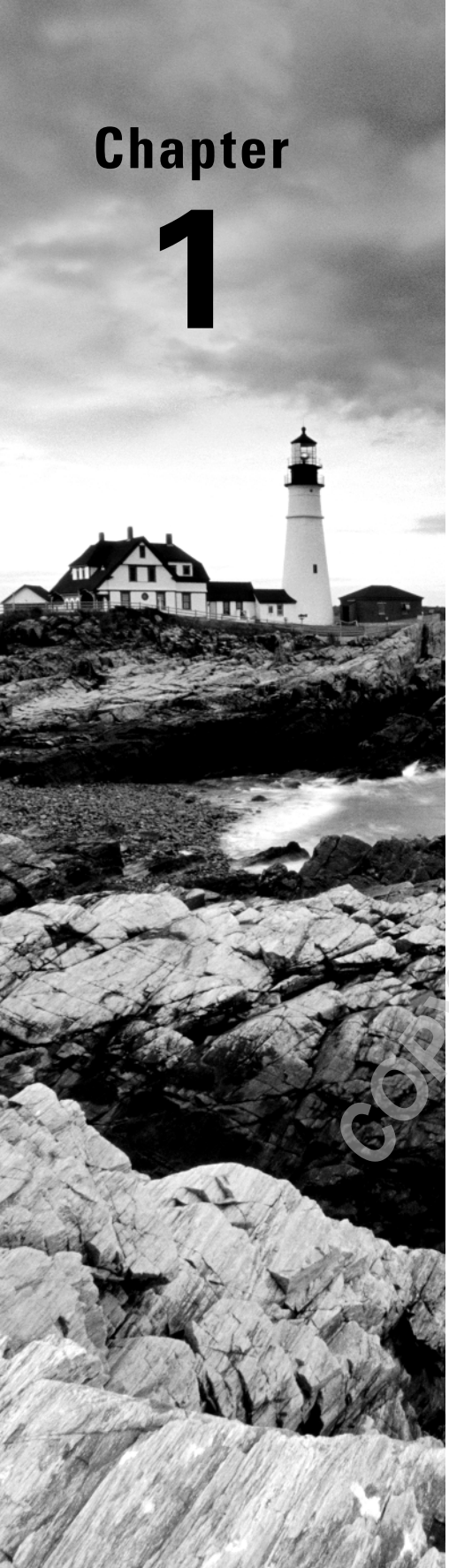


Chapter 1

Networking Fundamentals

THE CCNP ENCOR EXAM OBJECTIVES COVERED IN THIS CHAPTER INCLUDE THE FOLLOWING:

Domain 1.0: Architecture

- ✓ **1.1** Explain the different design principles used in an enterprise network
- ✓ **1.7** Differentiate hardware and software switching mechanisms

Domain 3.0: Infrastructure

- ✓ **3.1** Layer 2
- ✓ **3.2** Layer 3

Forgetting the fundamentals is by far the biggest cause of failures—both network failures and failing Cisco exams. Just visit any networking forum and look at the posts from people who failed an exam by a narrow margin. Almost without exception, they can trace back their failure to misunderstanding or simply failing to learn fundamental networking concepts.

Networking fundamentals can at times seem abstract and even impractical. It's important to remember that networks are both logical and physical, so you need to keep a tight grip on both. If you neglect theory and just focus on typing in commands, you'll end up with a jalopy network. It might work, but not very well, and probably not for long. On the other hand, learning theory that you fail to put into practice leads to being educated but unemployed.

This chapter will give you a solid theoretical foundation on which to build practical skills. Much of the theory should already be familiar to you, and you'll likely have some "I already know this stuff" moments. But more often than not you'll gain new insights on something you already understood.

There's a lot of networking information out there, much of which is poorly explained, if not just plain wrong. Networking myths abound on forums, blogs, and even Wikipedia. Even official Cisco documentation has been known to contain the occasional errata. It's not intentional, of course. Learning networking is no different than learning any other complex topic. Some concepts are easy, whereas others just never quite click. Those harder concepts are fertile breeding ground for misconceptions that eventually get passed around until they become common knowledge, or worse, "best practices." Almost every network professional I've encountered holds at least one glaring misconception about networking that eventually ends up stumping them (sometimes on an exam!). Chances are you, too, have been the unfortunate recipient of such information. The sooner we identify and dispel those myths, the better. That's what this chapter is all about.

The OSI Model

The origin of many networking myths can be traced back to the Open Systems Interconnection (OSI) reference model developed by Charles Bachman of Honeywell and formalized by the International Organization for Standardization (ISO). The ISO intended the OSI model to be a standard framework for data networks. It describes a set of "activities necessary for systems to interwork using communication media" (ISO/IEC 7498-4). The model organizes these activities or functions into the following seven layers:

7. Application
6. Presentation

5. Session
4. Transport
3. Network
2. Data Link
1. Physical

The seven layers are taught zealously in most introductory networking courses. You may have had them permanently drilled into your head with the help of one or two fun little mnemonics! (My favorite is "All people seem to need data processing.") As we discuss the functions of the different layers, keep in mind that the layers of the OSI model are arbitrary. They're not written on stone tablets, nor are they the result of a rigorous scientific process that conclusively proved that the perfect network has these seven layers. The ISO arrived at each layer by attempting to group similar network functions together in a layer and then organizing the layers in a hierarchical fashion so that each layer of functions is dependent on the one below it. This led to impressive results in layers 1–4 (the lower layers) and utter confusion in layers 5–7 (the upper layers).

Table 1.1 shows what common protocols fall into each of the lower layers.

TABLE 1.1 The lower layers and their associated protocols

Layer	Name	Example protocols
1	Physical	Thicknet (10BASE5)
		Thinnet (10BASE2)
		1000BASE-T
		T1/E1
2	Data Link	IEEE 802.3/Ethernet II (DIX)
		Point-to-Point Protocol (PPP)
		High-Level Data Link Control (HDLC)
3	Network	IPv4
		IPv6
4	Transport	TCP
		UDP

The Upper Layers: Application, Presentation, and Session

One thing that has always been clear about the OSI model is that the Application layer includes application data and application protocols. The Hypertext Transfer Protocol (HTTP) is an application protocol that a web browser uses for communicating with web servers. Application data would be an `HTTP GET` request that the browser sends to a web server. Likewise, the web page that the server sends in response would also be application data. In short, application data is whatever the application sends or receives over the network.

Incidentally, an application can use more than one protocol. For example, when a web browser uses the Hypertext Transfer Protocol Secure (HTTPS) protocol to send a request to a web server, it's making use of two protocols: HTTP and Transport Layer Security (TLS). Despite the latter's confusing name, both are application protocols.

For all practical purposes, the upper layers (Session, Presentation, and Application) are one layer: the Application layer. The actual functions of the Session and Presentation layers—things like authentication and negotiating an application protocol—occur in the application anyway. They don't include any network functions and are concerned only with application data and application protocols.

Making Sense of Layers

The ISO never clearly defined what a layer is. The closest they came was a circular definition. But we can infer from the OSI reference model what they had in mind.

For the curious, the ISO defined a layer as a "subdivision of the OSI architecture, constituted by subsystems of the same rank" (ISO/IEC 7498-1). While it's tautological that "subsystems of the same rank" are conceptually in the same layer, it still doesn't tell us what a layer *is*.

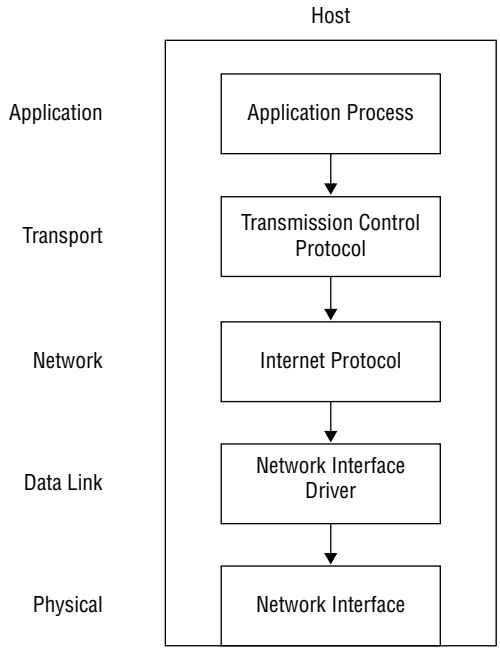
The concept of layering comes straight from software development (many of the OSI folks were operating system developers). The idea was that applications would treat the network as a software abstraction, somewhat like a filesystem. A filesystem acts as a layer that sits between the application and physical storage (e.g., disks). When the application needs to store some data, it just sends that data to the filesystem layer, which in turn takes care of the specifics of writing it to disk.

The OSI folks thought that in the same way that an application can store data on a filesystem without having to know anything about the underlying disks, so could it also send data over a network without requiring any network-specific coding or knowing anything about the network's infrastructure. Each layer would consist of a set of network-related functions implemented by the operating system or some middleware that would sit between the application and the host's physical network interface. Collectively, these layers would handle all the mechanisms of getting the application data onto the network and giving the network enough information to make sure the data got to its destination.

With the exception of the Physical layer, the layers of the OSI model are purely imaginary. Just as a filesystem is a software abstraction that hides the details of physical storage, the layers of the OSI model are just collections of software functions that hide the details of the network from applications and users. You can't see a filesystem with your eyes in the same way that you can see a hard drive, and you can't see the Data Link layer in the same way that you can see a switch. Layers are software abstractions and nothing more.

Figure 1.1 illustrates the concept of how layering might work using the Transmission Control Protocol (TCP) and Internet Protocol (IP), which are both included in the kernels of modern operating systems (Linux, Unix, and Windows). Keep in mind that the only real objects in this figure are the host and the physical network interface.

FIGURE 1.1 How layers abstract the network from an application

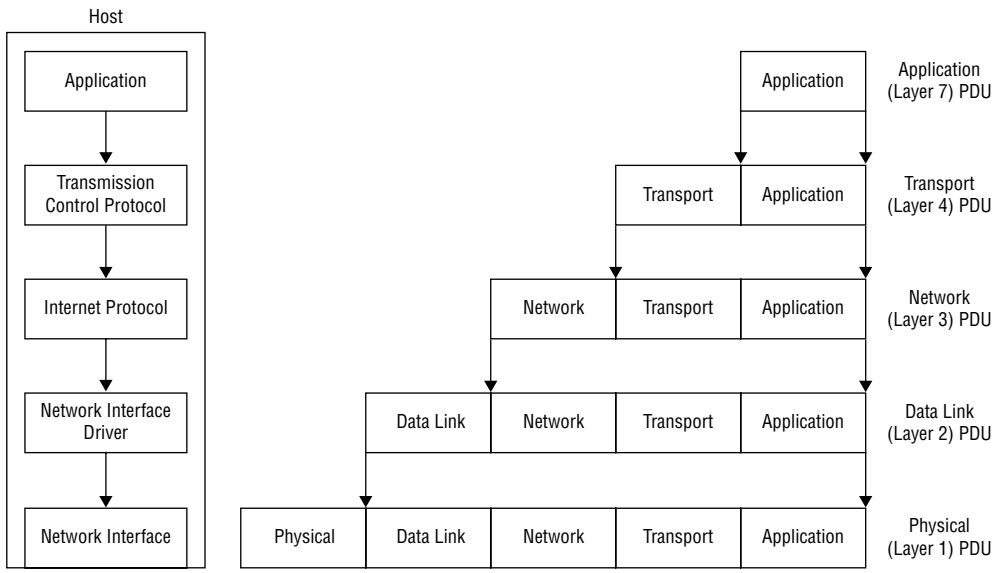


You may see some striking similarities between the layers in Figure 1.1 and the so-called TCP/IP or Internet protocol suite model. It and the OSI model are often juxtaposed as competing models. The fact is that the TCP/IP model is just a specific implementation of the OSI model based on the TCP/IP protocol suite.

In this high-level example, when an application needs to send data it places the data in what the OSI model generically calls an application protocol data unit (PDU). The specifics of the application PDU aren't important and, with the exception of firewalls that do deep

packet inspection, are opaque to the network. The application passes its PDU to a protocol in the layer directly below, as shown in Figure 1.2. The protocol generates a new PDU and tacks the application PDU onto the end of it—a process called *encapsulation*. It then passes this new PDU down to a protocol at the next lower layer, and so on. What ends up on the wire is a giant PDU that contains several smaller PDUs from the protocols operating at the higher layers. Later in the chapter we'll walk through a detailed example of how encapsulation works, but first, we need to talk about what happens at each of these lower layers.

FIGURE 1.2 At each layer, data is encapsulated in a PDU and passed down to the next lower layer.



The Lower Layers: Physical, Data Link, Network, and Transport

The purpose of a network is to allow applications running on different hosts to communicate with one another. Robert Metcalfe, one of the inventors of the original Ethernet, said it succinctly in 1972: "Networking is interprocess communication." Thus, at a minimum, a network needs to perform three basic functions:

Layer 1: Physical Connectivity between Nodes A node can be a workstation, server, router, switch, firewall, or any network-connected device that has a processor and memory.

Layer 2: Node-to-Node Data Transfer Data transfer between two nodes physically connected to a shared medium.

Layer 3: Forwarding/Routing Data transfer between any two nodes, regardless of whether they're physically connected to the same medium.

The OSI model sorts these three functions along with many others into the first four layers of the OSI model, as shown in Table 1.2. Not all protocols that operate in a given layer implement all the functions listed for that layer.

TABLE 1.2 Networking functions provided by each layer

Function	1 Physical	2 Data Link	3 Network	4 Transport
Transmission of bitstreams over physical media	X			
Enabling/disabling physical network interface	X			
Node-to-node data transfer over a shared medium		X		
Forwarding/routing		X	X	
Error control		X	X	X
Flow control		X	X	X
Multiplexing/splitting		X	X	X
Ordering		X	X	X
Fragmentation/reassembly			X	X

The OSI replicates some functions in most layers, blurring the distinction among them. It becomes apparent that what distinguishes the layers isn't what they do but what they *don't* do. Higher layers lack functionality provided by lower layers, something you'd expect given the hierarchical structure of layers. One layer whose functions differ starkly from the others is the Physical layer.

Layer 1: The Physical Layer

The main function of the Physical layer is to convert bits to electromagnetic energy such as light, electrical current, or radio waves, and transmit them over some medium such as fiber-optic or copper cables or the airwaves. Whereas the functions of the other layers are performed in software, this particular function is performed by a node's physical network interface.

A challenge of using electromagnetic energy to send bits is that the physical media can carry only one bitstream at a time. In the early days of networking, two nodes would be connected via a pair of wires. If both simultaneously sent a signal, their signals would interfere with each other and create a collision. Hence, both nodes were in the same collision domain. To avoid this, the nodes had to use half-duplex communication wherein only one node could transmit at a time. Half-duplex wired communication may seem an irrelevant relic from the past, but as you'll learn in a moment, during its heyday half-duplex communication had an unfortunate impact on the Ethernet standards that still haunts us to this day. Broadcast storms and the infamous Spanning Tree Protocols (STPs) can be traced back to the early use of half-duplex communication.

Today, full-duplex communication is the norm in wired networks and something we take for granted. All that's needed for full-duplex communication is for the physical interface to separate the transmit and receive functions. Twisted-pair copper cabling, for example, does this by using two pairs of wires: one pair for transmitting and another pair for receiving. Likewise, fiber-optic cables have separate strands for transmitting and receiving. Wavelength-division multiplexing achieves full-duplex communication on a single fiber strand by using one light frequency for transmitting and another for receiving.

Layer 2: The Data Link Layer

The primary function of the Data Link layer is to facilitate data transfer between two (and only two) nodes that are connected to a shared medium. Some physical media can support only two nodes, as is the case with a crossover cable or point-to-point serial link. Other media, such as wireless, can support more than two nodes.

When only two nodes share the same media, data transfer is easy. As long as both nodes are aware of the point-to-point nature of the link, one node can send the data, and the other node receives it, knowing that it's the intended recipient. The Point-to-Point Protocol (PPP) and High-level Data Link Control (HDLC) are two common layer 2 protocols used on T1 serial links.

But when multiple nodes can share a medium, as they did with early Ethernet, things get tricky. At this point you're rightly thinking that with the exception of wireless, nobody connects nodes to a shared medium anymore. Hubs went out of fashion long ago. Now we connect devices to switches (the marketing term for bridges). However, switches actually *simulate* the behavior of a shared medium. Time for a quick history lesson.

A Brief History of Ethernet

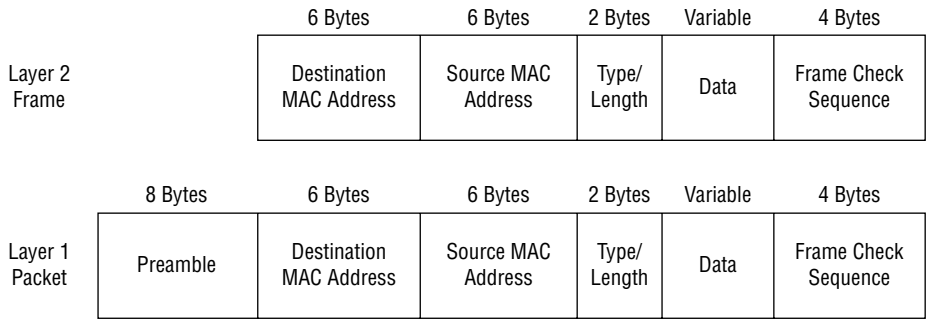
The original Ethernet standards from the 1970s were designed for nodes all connected to a shared electrical bus that often took the form of a thick yellow cable (you may have heard

the term Thicknet). Whenever one node would transmit a signal, all other nodes connected to the cable would receive it. Communication was half-duplex, and all nodes were in the same collision domain. As a way of detecting errors introduced by collisions, the original Ethernet II (DIX) specification got a frame check sequence (FCS, sometimes called a cyclic redundancy check, or CRC) to detect errors. Even today as back then, nodes discard frames that fail the FCS check.

The multi-access nature of Ethernet made it necessary to assign each node's network interface a unique, 48-bit Media Access Control (MAC) address. The sending node would construct a frame that included the destination node's MAC address and the data to send. All nodes would receive the frame, but only the destination node would process it.

Now let's fast-forward to today. We still use MAC addresses, even though the original rationale for using them is long gone. To maintain backward compatibility over the years, we never got rid of them. Figure 1.3 shows the original DIX frame format that we still use today. We're still using a technology designed specifically for devices that were all sharing a thick yellow cable. Today, however, instead of nodes sharing this thick yellow cable, they're connected to a switch.

FIGURE 1.3 Layer 2 frame and layer 1 packet, structurally identical to the revised (1997) IEEE 802.3 format that we use today



You may have seen diagrams that show the Ethernet frame with an 8-byte preamble at the beginning. The preamble is not actually part of the frame but is a series of bits that provide clock synchronization for the Physical layer and signal the start of the frame. The entire collection of bits—including the preamble and frame—compose a layer 1 Ethernet packet. Although most of the time when you hear "packet" it refers to an IP packet (layer 3), "packet" is a generic term for any PDU. To avoid confusion, you can think of the raw bits as a layer 1 Ethernet PDU.

Switches replace the shared cable of the early Ethernet with multiple cables, breaking the inherent broadcast nature of the thick yellow cable. Switches thus have to perform some interesting hackery to maintain backward compatibility with the early Ethernet standards. When a switch receives an Ethernet frame, by default it forwards that frame to all other devices connected to the switch—a process called flooding or broadcasting. This creates the illusion that all nodes are connected to the same thick yellow cable. (In networking parlance, they're all in the same broadcast domain or segment.) This illusion is called transparent bridging (aka switching) because to the nodes, the switch is invisible. Incidentally, you'll recognize this simulated yellow cable by its common name: a local area network (LAN).

The MAC Address Table

Although switches eliminate collision domains by offering full-duplex communication, they still waste bandwidth by flooding traffic to nodes that don't need it. To mitigate flooding, switches implement a form of routing. When a switch receives a frame on an interface, it records the ingress interface and source MAC address in its MAC address table. Subsequently, when a switch receives a frame destined for that same MAC address, it queries the MAC address table, which returns the interface number. The switch then forwards the frame only out of that interface, rather than flooding it.

The MAC address table is stored in a type of memory called content-addressable memory (CAM). CAM is often used as a synonym for the MAC address table. The CAM takes a MAC address and VLAN as input and returns an interface name and number as the output. CAM provides faster read times than RAM.

```
SW3#show mac address-table dynamic
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0c3c.8a00.5e02    DYNAMIC     Gi0/2
   1    0c3c.8ad7.9101    DYNAMIC     Gi0/2
   1    0c3c.8afd.c101    DYNAMIC     Gi0/1
   1    0c3c.8afd.c102    DYNAMIC     Gi0/2
  10    0c3c.8ad7.800a    DYNAMIC     Gi0/0
  20    0c3c.8ad7.8014    DYNAMIC     Gi0/0
Total Mac Addresses for this criterion: 6
```

The use of the MAC address table changes the fundamental nature of MAC addresses. They no longer function as just names for identification, but also as addresses for location.

On the other hand, if a switch receives a frame for a MAC address that doesn't have a mapping in the MAC address table—called an unknown unicast—it reverts to its default behavior and floods the frame out of all other interfaces.

Unknown unicasts are more common than you might think. Entries in the MAC address table don't last forever. By default, a MAC address entry is deleted or ages out 300 seconds (5 minutes) after the switch last sees the traffic from the MAC address. Note that aging time is not based on when the entry was created.

```
SW3#show mac address-table aging-time vlan 1
Global Aging Time:  300
Vlan    Aging Time
----    ----------
   1     300
```

You can adjust the global aging time to between 10 and 1,000,000 seconds or disable aging by setting the aging time to 0.

```
SW3(config)#mac address-table aging-time ?
  <0-0>         Enter 0 to disable aging
  <10-1000000>  Aging time in seconds
```

You can also adjust the aging time on a per-VLAN basis.

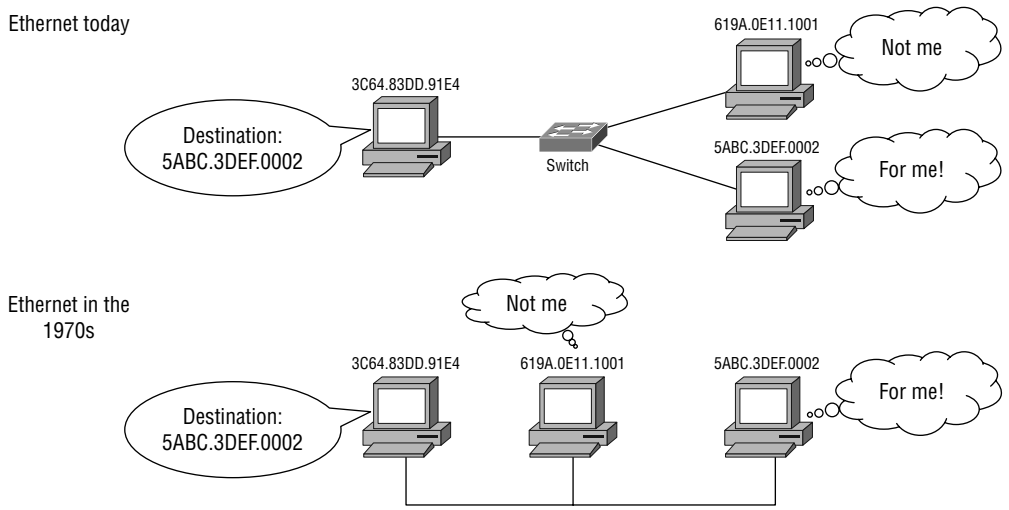
```
SW3(config)#mac address-table aging-time 300 vlan ?
  <1-4094>  VLAN id
```

Disabling aging might sound like a good idea, as it would prevent flooding, right? Not necessarily. The CAM has a finite amount of space, and once the MAC address table is full, the switch will flood traffic to every destination MAC not in the table.

The MAC address table mitigates flooding but doesn't eliminate it. The fundamental flooding behavior of Ethernet remains. To make matters worse, Ethernet implements a special MAC address called a broadcast address (FFFF.FFFF.FFFF). Frames sent to this address are flooded out of all ports. You can imagine the number of major outages that arose from this unwise decision!

The end result is that any node in a broadcast domain can send a frame to another node and the destination node will receive it. We may have added more cables and more devices, but the fundamental behavior of Ethernet hasn't changed in 50 years, as shown in Figure 1.4. When it comes to networking, history has a way of repeating itself.

FIGURE 1.4 Early Ethernet over a shared medium compared to Ethernet using a switch



Maximum Transmission Unit

Another interesting side effect we inherited from the use of the legendary thick yellow cable is that Ethernet had to impose a limit on the maximum frame size to keep a single node from hogging the medium with colossal frames. The Ethernet maximum transmission unit (MTU) defines the maximum size of the Data field in bytes. DIX and IEEE 802.3 support a maximum MTU of 1,500 bytes. Higher-layer protocols trying to send packets larger than the MTU must break apart their packets into fragments that will fit into the frame's Data field. To avoid fragmentation, some interfaces support jumbo frames with an interface MTU of 9,000 bytes to 9,216 bytes.

Subnet Limits

When you think of the term subnet, you probably think of an IP subnet address and mask, such as 192.168.1.0/24. The IP subnet address and mask collectively form a CIDR block, or just CIDR for short. But a subnetwork (subnet) is actually a collection of connected nodes that all use the same Data Link layer protocol. For example, a collection of nodes in the same VLAN is an example of a subnetwork. To avoid confusion, I'll refer to the combination of IP subnet address and mask as either a CIDR or an IP subnet.

The moral of the convoluted story behind Ethernet and bridges is that no matter how many tricks and kluges you invent, you can't extend a subnet beyond a few hundred nodes. Regardless of the protocols used, the number of nodes in a subnet is limited by the underlying physical media. To create large networks that include thousands or millions of nodes, we need to join multiple subnets together. This brings us to the Network layer.

Layer 3: The Network Layer

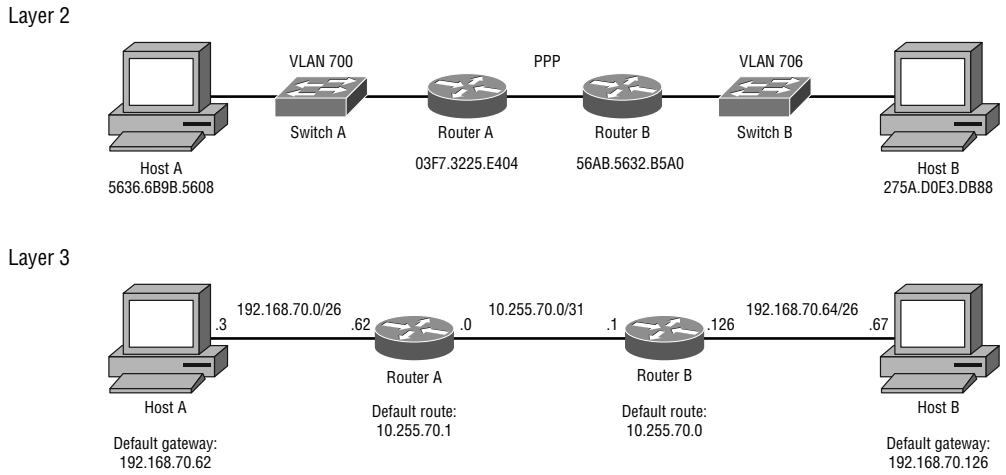
Recalling that a subnet consists of connected nodes running the same Data Link layer protocol, the Network layer's primary function is to enable data transfer between nodes that may or may not be in the same subnet. Hence, Network layer protocols must ensure that two things happen:

- Nodes in different subnets will communicate using a gateway/router.
- Nodes in the same subnet will communicate with one another using the Data Link layer protocol.

It may seem redundant for the Network layer to enable connectivity between nodes in the same subnet, since the Data Link layer already provides this functionality. But the purpose of the Network layer is to abstract the physical and data link characteristics of the network away so that applications don't need to be concerned with them. Instead, the application just deals with Network layer addresses—usually IP addresses.

To see how IP abstracts away the Data Link layer, compare the layer 2 and layer 3 topologies shown in Figure 1.5.

FIGURE 1.5 Simple layer 2 and layer 3 topologies



IP creates an addressing scheme on top of the Data Link layer, giving each subnet a different CIDR—a combination of an IP subnet address and subnet mask:

VLAN 700—192.168.70.0/26

PPP—10.255.70.0/31

VLAN 706—192.168.70.64/26

A CIDR is the name that IP uses to address a subnet. Hence, a CIDR and subnet should always be tightly coupled, but they're not the same thing. The purpose of a CIDR (IP

subnet address and mask) is to help a node determine based on the destination's IP address whether it's in the same subnet or a different subnet. If the destination's IP is in the same CIDR, the node assumes it's in the same subnet and will address the frame to the node's MAC address. Otherwise, the node will assume the destination is in another subnet and will address the frame to the MAC address of the default gateway for the subnet.

The OSI's dream of turning the network into a software abstraction begins to show cracks in the Network layer. Applications do indeed need to have some knowledge of the network, even if it's just IP addresses.

Forwarding within a Subnet

If the destination is in the same subnet, the node will simply communicate with the destination at the Data Link layer. For example, if Host A (192.168.70.3) attempts to ping Router A (192.168.70.62), the following will happen:

1. Host A will note that Router A's IP is in the same subnet.
2. Host A will send an Address Resolution Protocol (ARP) request to the broadcast destination address, asking who has 192.168.70.62.
3. Switch A will flood the ARP request and Router A will receive it.
4. Router A will send an ARP reply to Host A's MAC address. The reply will contain Router A's IP address (192.168.70.62) and MAC address.
5. Switch A will forward the ARP reply to Host A.
6. Host A will encapsulate the IP packet inside an Ethernet frame addressed to Router A's MAC address. Host A will set the Type field in the frame to `0x0800`, indicating that the frame contains an IP packet.

A ping is an Internet Control Message Protocol (ICMP) echo request. ICMP is an integral part of IP.

Forwarding between Subnets

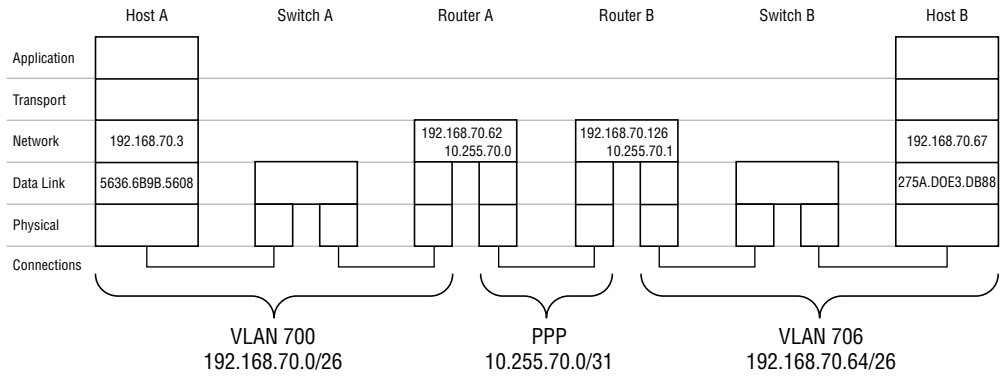
On the other hand, if Host A (192.168.70.3) attempts to ping Host B (192.168.70.67), the following happens:

1. Host A compares its IP address with Router B's IP and determines that they are in different subnets.
2. Host A consults its IP routing table for a closest-match route to 10.255.70.1. Not finding an exact match, the closest match is the default route (0.0.0.0/0). Host A's default gateway is 192.168.70.62, the IP belonging to Router A's Ethernet interface.

3. Because Router A's and Host A's IP addresses are in the same subnet, Host A sends an ARP request asking for Router A's MAC address.
4. Switch A floods the ARP request to Router A.
5. Router A sends an ARP reply to Host A's MAC address. The reply contains Router A's IP address and MAC address.
6. Host A encapsulates the IP packet inside an Ethernet frame addressed to Router A's MAC address. The Type field will contain the value `0x0800` to indicate that the Data field contains an IP packet.
7. Router A receives the Ethernet frame and, based on the Type field in the Ethernet frame, knows it contains an IP packet.
8. Router A looks at the destination IP address in the IP packet and checks its forwarding information base (FIB) for an exact match. Cisco Express Forwarding (CEF) uses the FIB to make forwarding decisions. The FIB is fed by the IP routing table (also known as the Routing Information Base, or RIB). Not finding an exact match for the destination IP address in the FIB, it will use the default route, which has Router B (10.255.70.1) as its next hop.
9. Router A will encapsulate the IP packet in a PPP frame and send it to Router B.
10. Router B will decapsulate the IP packet, look at the destination IP address, and check its FIB for a match. Because the destination IP (192.168.70.67) is in the same subnet as Router B's Ethernet interface, Router B will send an ARP request.
11. Switch B will flood the ARP request to Host B, which will send an ARP reply to Router B's MAC address. The ARP reply will contain Host B's MAC address and IP address.
12. Router B will encapsulate the IP packet in an Ethernet frame addressed to Host B's MAC address.

The reason that the process takes so many steps is that the ICMP data is zigzagging up and down the different layers at each node. Figure 1.6 puts everything in context with a layered view of the topology.

FIGURE 1.6 Layered representation of the network



It's worth noting that when you create multiple VLANs on a switch, you're simply creating separate broadcast domains. Nodes in one VLAN can't communicate with nodes in the other using MAC addresses because they're in different subnets. Even if those nodes are connected to the same switch, they must use IP and go through a router to communicate with nodes in the other VLAN.

It's crucial that a CIDR block belong to only one subnet—that is, one section of the network where all the connected nodes use the same Data Link layer protocol. A common mistake is to try to split a CIDR across different subnets that are usually in geographically separated areas, like different data centers. The rationale for subnet splitting is to achieve some sort of resiliency with minimal inconvenience, particularly by being able to migrate virtual machines from one site to another without changing any IP addresses. This requires using some network virtualization technology like Virtual Extensible LAN (VXLAN) to create the illusion of extending the subnet, when in fact it's stuffing Ethernet frames inside of IP packets and sending them across multiple subnets, in essence creating a virtual subnet! Remember that a subnet can't scale beyond a few hundred nodes—not even a virtual subnet.

Address Resolution Protocol

Most devices with an IP address—including workstations, servers, routers, and switches—maintain an ARP cache to store ARP replies. The purpose of the ARP cache is to avoid having to send an ARP request every time the node needs to resolve an IP address to a MAC address.

When a node needs to resolve the MAC address of an IP address not in its ARP cache, it sends an ARP request to the broadcast address (`FFFF.FFFF.FFFF`). Upon receiving a reply, it stores the mapping in its ARP cache. The following example illustrates the process using two switches:

- SW3 has a switched virtual interface (SVI) in VLAN 20 with an IP address of 10.10.20.3.
- SW4 has an SVI also in VLAN 20 with an IP address of 10.10.20.4.

SW3:

```
! Show the ARP cache on SW3
SW3#show arp dynamic
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.10.10.4              0   0c3c.8ad7.800a  ARPA   Vlan10
! Trigger an ARP request for 10.10.20.4 by sending a ping to it
SW3#ping 10.10.20.4
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.20.4, timeout is 2 seconds:
.!!!!
```

```
Success rate is 80 percent (4/5), round-trip min/avg/max = 10/10/10 ms
! The ping succeeded, implying an ARP reply was received. Show the ARP cache
again.
SW3#show arp dynamic
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.10.10.4              0   0c3c.8ad7.800a  ARPA   Vlan10
Internet  10.10.20.4             0   0c3c.8ad7.8014  ARPA   Vlan20
```

SW4:

```
! ARP Snooping debugging has been enabled on SW4. Note the destination broadcast
! address.
SW4#
ARP Packet (Gi1/0/20) Src: 0c3c.8aab.8014, Dst: ffff.ffff.ffff, SM:
0c3c.8aab.8014, SI: 10.10.20.3, TM: ffff.ffff.ffff, TI: 10.10.20.3
Packet bridged by platform.
ARP Packet (Gi1/1/20) Src: 0c3c.8aab.8014, Dst: ffff.ffff.ffff, SM:
0c3c.8aab.8014, SI: 10.10.20.3, TM: ffff.ffff.ffff, TI: 10.10.20.3
Packet bridged by platform.
! Although not shown in the output, SW4's ARP reply is addressed to SW3's
! SVI MAC address.
```

The default timeout for an ARP entry is 4 hours. You can modify this on a per-interface basis, as shown on SW3:

```
SW3#show interfaces vlan 20 | i ARP
  Encapsulation ARPA, loopback not set
  ARP type: ARPA, ARP Timeout 04:00:00
SW3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#interface vlan 20
SW3(config-if)#arp timeout ?
  <0-2147483>  Seconds
```

You'll hear disagreement as to whether ARP is a layer 2 or layer 3 protocol, some even going so far as to call it a layer 2.5 protocol! ARP packets fit the definition of what the OSI model calls protocol control information. In addition to just providing a mapping between MAC and IP addresses, the fact that a node sends ARP packets indicates its willingness to use IP. In that respect, ARP is decidedly a layer 2 protocol.

Fragmentation

If an IP packet exceeds the MTU of any interfaces it has to traverse (the path MTU), then any intermediate router may fragment the packet into multiple datagrams. Each datagram

must be no greater than the path MTU. The sender can optionally set the don't fragment (DF) bit in the IP header to prevent intermediate routers from fragmenting the packet.

IPv6 differs from IPv4 when it comes to fragmentation. IPv4 packets can be fragmented by any router along the path unless the DF bit is set. IPv6 can be fragmented only by the sender. If an IPv6 packet will exceed an intermediate router's interface MTU, the router will respond to the sender with an ICMPv6 "packet too big" message and discard the packet.

Routing vs. Forwarding

What's the difference between routing and forwarding? Not much, really. Forwarding is about sending the data one step closer to its destination. Routing is about figuring out what that next step is.

The routing versus forwarding distinction has nothing to do with layers. Recall that switches perform a crude version of routing by snooping the data plane to find out which port a MAC address is connected to. They compile this into a MAC address table, which they use to make forwarding decisions.

When it comes to IP, route calculation and route advertisements are performed by interior gateway routing protocols such as Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First (OSPF). Although we don't normally think of them in this way, routing protocols are actually applications that run on routers. They just populate the IP routing table that feeds into the FIB, but CEF does the forwarding.

Layer 4: The Transport Layer

So far, we've seen how protocols at the first three layers enable communication between two host interfaces. The primary purpose of the Transport layer is to facilitate application-to-application (end-to-end) data transfer. Whereas Network layer protocols (e.g., IPv4, IPv6) provide a way to move data from one host's interface to another host's interface, the Transport layer protocols—TCP and UDP—provide a means for applications to distinguish different communication streams. They both do this using 16-bit port numbers, as shown in Table 1.3.

TABLE 1.3 Common applications and their TCP and UDP port numbers

Application protocol	Transport protocol	Source IP	Source port	Destination IP	Destination port
HTTP	TCP	192.168.88.10	5230	18.213.128.4	80
HTTP	TCP	192.168.88.10	5231	18.213.128.4	81
DNS	UDP	192.168.88.10	56801	192.168.88.1	53

For example, when a web browser retrieves a web page it may open multiple TCP connections to the same web server. Each TCP connection originates from a different ephemeral (short-lived) source port chosen by the operating system, allowing the web browser and web server to keep track of which requests go with which connection.

The protocol data unit for TCP is called a *segment*, and for UDP it's called a *datagram*.

When a host receives an IP packet, the host's networking stack looks at the Protocol field to determine to which upper-layer protocol to send the data. If the Protocol field in the IP header is 6, the data contains a TCP segment. If it's 17, then it contains a UDP datagram. Consequently, a single host can use the same UDP and TCP port numbers simultaneously.

Transport layer protocols aren't always necessary. The interior gateway protocols EIGRP and OSPF ride directly over IP, using the IP protocol numbers 88 and 89, respectively.

Transmission Control Protocol

RFC 793 somewhat redundantly describes TCP as a "connection-oriented, end-to-end reliable protocol." TCP provides the following features:

- Reliable delivery
- Ordering
- Error control
- Flow control
- Congestion avoidance

The phrase "connection oriented" refers to TCP's attempt to simulate the properties of a physical connection. Sound familiar? An ideal connection provides reliability and order. The data the sender sends is exactly what the receiver receives and in the same order.

TCP guarantees both order and reliability through the use of sequence numbers and acknowledgments. The sender marks each TCP segment with a sequence number that increments in a predictable fashion. This allows the receiver to reassemble the segments in the correct order should they arrive out of order. The receiver sends an acknowledgment (ACK) in response to each segment, indicating the sequence number of the segment it received. If the sender fails to receive an ACK after a period of time, it will retransmit the segment.

Connection Establishment

You should be familiar with TCP's famous three-way handshake. Its purpose is to allow both sides to establish initial sequence numbers. Here's how it works:

1. Node A sends Node B a segment with the synchronize (SYN) flag set, along with its initial sequence number (ISN), which is usually random.

2. Node B responds with a SYN and its own ISN, which is different from Node A's ISN. Node B also sends an ACK to acknowledge Node A's initial SYN.
3. Node A acknowledges Node B's SYN by sending an ACK.

Connection Termination

When the client is ready to terminate the connection, it sends a segment with the finish (FIN) flag set. The server responds with an ACK; however, it can continue to send data. This is called a half-close state. When the server is ready to terminate the connection, it too sends a segment with the FIN flag set. The client ACKs this segment, and the connection is fully closed.

Connection Reset

The TCP Reset (RST) flag is useful for cleaning up old and duplicate connections. Generally, when a TCP receives a segment for a port that it's not listening on, it will respond with a RST, letting the sender know that the connection was refused. This can happen if the sender tries to send data over an old connection that it believes is still open.

In another case, if a TCP doesn't receive any ACKs from a host, it will abort the connection by sending a RST. This may happen if there's a unidirectional link or someone abruptly shuts down the host, precluding it from gracefully terminating the connection. On the off chance that the host receives the RST, it will abandon the connection and may attempt to open a new one.

Error Control

Error control detects whether application data was lost or corrupted during transit. Aside from port numbers, this is the only feature both TCP and UDP have in common. Both TCP and UDP provide error control by calculating a checksum over the entire packet and including it in the header. The recipient calculates its own checksum over the entire packet and compares it to what's in the header. If they don't match, the recipient discards the packet.

Flow Control

Flow control prevents a sender from sending data faster than the receiver can receive it. TCP provides flow control through the use of windows and acknowledgments. The receiver specifies a window—the number of bytes that the sender may send before stopping to wait for an acknowledgment from the receiver.

Congestion Avoidance

Congestion avoidance can be described as TCP's politeness algorithm. If a sender fails to receive an ACK or receives duplicate ACKs, TCP assumes that there's congestion somewhere in the network and slows down the rate at which it sends.

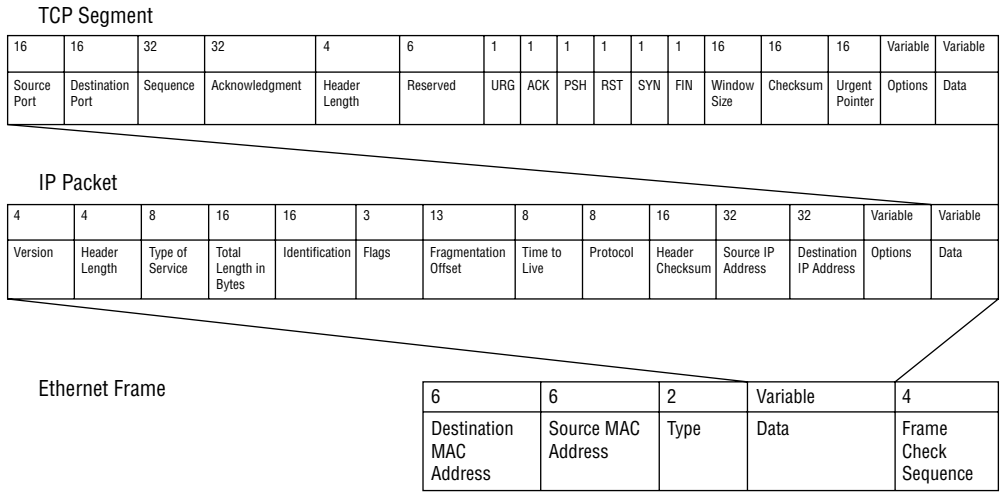
Encapsulation and Decapsulation

The entire process of moving data from one application to another over the network rests on two processes: encapsulation and decapsulation. Without encapsulation and decapsulation, there is no network. In short, each time a higher-layer protocol passes data down to a lower-layer protocol, it's encapsulated. Whenever a lower-layer protocol passes data up to a higher-layer protocol, it's decapsulated.

Encapsulation and Multiplexing

Although they're technically different, in networking parlance multiplexing and encapsulation are used interchangeably. Multiplexing is the act of sending multiple streams of data over a single connection. Nodes achieve multiplexing by using encapsulation, as shown in Figure 1.7.

FIGURE 1.7 Encapsulation of a TCP segment and IP packet inside an Ethernet frame



As an example, suppose a TCP-based client application needs to send some data to a server. The application instructs the host operating system to open a TCP connection to the server. Once the connection is open, the application uses it to send some data to the server. Here's what happens: The host's operating system creates a TCP segment, places the application data in the Data field, and populates the rest of the fields in the TCP header appropriately. The OS encapsulates the TCP packet inside an IP packet, with the Protocol field set to 6 (TCP). It then sends the IP packet to the Ethernet driver, which encapsulates the IP packet in an Ethernet frame, which has the Type field set to `0x0800` (IP). The Ethernet driver passes the frame to the network interface, which converts the data to bits and puts it out on the wire.

Decapsulation and Splitting

Splitting or demultiplexing is the multiplexing process played backward. Continuing with the preceding example, when the server receives the Ethernet frame containing the IP packet from the client, the Ethernet driver looks at the Type field in the Ethernet header and sees that it's `0x0800`, indicating an IP packet. The Ethernet driver decapsulates the IP packet from the frame and passes it to the operating system's IP stack. The IP stack looks at the Protocol field and sees that it's 6, indicating that the Data field contains a TCP segment.

Summary

The backstory of the OSI model and Ethernet seems to have little relevance to "modern" networking. But upon closer inspection, many of the fundamentals of networking haven't changed in 50 years. Essentially, the goal of networking is and always has been to enable applications to transfer data. The challenge of networking is to make that happen without having to hard-code gritty networking details into each application. That's where the layering concept of the OSI model comes in. Each layer is an abstraction that conceals some part of the network from the application. Although the OSI model is abstract, the protocols that networks implement today in a mostly consistent manner are very real.

Data Link layer protocols such as Ethernet, PPP, and WLAN conceal the details of network interfaces and provide a common frame format that can be used across different media types. Network layer protocols such as IPv4 and IPv6 hide the different Data Link layer protocols in use across different networks. This is why the Network layer is sometimes called the *Inter*network layer. Applications can use Transport layer protocols such as TCP and UDP to distinguish different communication streams.

The upper layers—Application, Session, and Presentation—have always posed a challenge for the OSI model. It's clear that these are all one layer: the Application layer. What makes Application layer protocols unique is that they are "the end of the line." That is, there is no higher-layer protocol for an application to pass data up to. Hence, many things that we previously thought of as simply networking protocols are actually applications: ARP, BGP, EIGRP, and OSPF, just to name a few. PDUs generated by these aren't passed up to any higher-layer protocol. When you look at the Application layer in this way, the network suddenly looks a lot simpler.

Exam Essentials

Understand why IP addresses and MAC addresses name interfaces, not nodes. An interface is bound to a subnet, and a node can have multiple interfaces. Other nodes in the same subnet can't determine whether any two MAC or IP addresses belong to the same node.

Know how switches extend a subnet across different physical media. Switches use flooding to forward broadcasts and unknown unicasts to all connected nodes in a given VLAN.

Understand how routers enable IP connectivity between subnets. When a router receives a layer 2 frame containing an IP packet, it decapsulates the packet and looks at the destination IP address. It checks its FIB to determine the next hop's IP address. If the next hop is reachable via Ethernet, it re-encapsulates the packet in an Ethernet frame addressed to the next-hop node's MAC address and forwards it. If the next hop is reachable via a PPP or HDLC connection, it encapsulates the IP packet in a PPP or HDLC frame and forwards it.

Know the encapsulation and decapsulation process for the protocols at each layer. With the exception of the Application layer, the PDU at each layer contains a reference to a protocol in the layer above. For example, an Ethernet frame contains a Type field that indicates a Network layer protocol, such as IPv4 (`0x0800`) or IPv6 (`0x86DD`). An IP packet contains a Protocol field indicating a Transport layer protocol, such as TCP (6) or UDP (17).

Understand the primary purpose of each layer. The Data Link layer facilitates data transfer between two nodes connected to a shared medium. The Network layer enables data transfer between nodes that may or may not be in the same subnet. The Transport layer facilitates application-to-application data transfer and provides error detection.

Review Questions

You can find the answers in the appendix.

1. Which of the following protocols operate at the Physical layer?
   - **A.** HDLC
   - **B.** IEEE 802.3
   - **C.** PPP
   - **D.** Twisted pair
2. What protocol operates at the Data Link layer?
   - **A.** IPv4
   - **B.** OSPF
   - **C.** PPP
   - **D.** UDP
3. What layer does ARP operate at?
   - **A.** 2
   - **B.** 3
   - **C.** 4
   - **D.** 7
4. Protocols at what layer provide for node-to-node data transfer over a shared medium?
   - **A.** Physical
   - **B.** Data Link
   - **C.** Network
   - **D.** Transport
5. What are two examples a single collision domain?
   - **A.** A single fiber strand that carries separate light frequencies for sending and receiving
   - **B.** Wireless
   - **C.** Two fiber strands, one for sending and another for receiving
   - **D.** A twisted-pair cable with a single pair of wires for sending and receiving
6. What can be done to achieve full-duplex communication?
   - **A.** Use fiber-optic cables
   - **B.** Use TCP
   - **C.** Use different pairs of wires for transmitting and receiving
   - **D.** Set the interface speeds to something greater than 10 Mbps

7. Workstation A is connected to a switch port that's in VLAN 10. Workstations B and C are connected switch ports in VLAN 20 on the same switch. Both workstations are configured with IP addresses in the 172.16.7.0/24 range. How many broadcast domains are there?
   - **A.** 0
   - **B.** 1
   - **C.** 2
   - **D.** 20

8. A server is connected to a switch. Both the server's network interface card (NIC) and the switch port are configured for full-duplex communication. How many collision domains are there?
   - **A.** 0
   - **B.** 1
   - **C.** 2
   - **D.** 3

9. What's the default global aging time for the MAC address table?
   - **A.** 30 seconds
   - **B.** 5 minutes
   - **C.** 1 hour
   - **D.** 4 hours

10. What occurs when the MAC address table is full?
    - **A.** All entries in the table are deleted.
    - **B.** The oldest entries are aged out.
    - **C.** Frames addressed to a MAC address that's not in the table are flooded.
    - **D.** Frames addressed to a MAC address that's not in the table are dropped.

11. What is an Ethernet interface MTU?
    - **A.** The minimum size of an Ethernet frame
    - **B.** The maximum size of an Ethernet frame
    - **C.** The speed of an Ethernet interface
    - **D.** The maximum size of the Data field in an Ethernet frame

12. Which of the following is a function of a bridge?
    - **A.** MAC-based routing
    - **B.** IP-based routing
    - **C.** Connecting two VLANs together
    - **D.** Reducing the size of a broadcast domain

13. What information does the MAC address table store? (Choose two.)

- **A.** IP address
- **B.** VLAN
- **C.** ARP entries
- **D.** Interface

14. What does ARP do?

- **A.** Maps MAC addresses to IP addresses
- **B.** Maps MAC addresses to interfaces
- **C.** Maps IP addresses to MAC addresses
- **D.** Maps IP addresses to interfaces
- **E.** Maps MAC addresses to VLANs

15. Client A in VLAN 3 has the IP address 172.16.3.3/24. Server A in VLAN 4 has the IP address 172.16.3.10/24. What will occur if client A attempts to ping Server A? (Choose two.)

- **A.** The ping will succeed.
- **B.** Client A will send an ARP request for 172.16.3.10.
- **C.** The ping will fail.
- **D.** Server A will send an ARP reply.

16. What destination address is an ARP request sent to?

- **A.** The MAC address 0000.0000.0000
- **B.** The MAC address FFFF.FFFF.FFFF
- **C.** The MAC address 0100.5E01.0001
- **D.** The IP address 255.255.255.255

17. What destination address is an ARP reply sent to?

- **A.** The IP address that sourced the ARP request
- **B.** 255.255.255.255
- **C.** FFFF.FFFF.FFFF
- **D.** The MAC address that sourced the ARP request

18. What's the default ARP entry timeout on Cisco routers and switches?

- **A.** 5 minutes
- **B.** 1 hour
- **C.** 4 hours
- **D.** 6 hours

19. What's the purpose of TCP connection establishment?

A. Error control

B. Synchronization of sequence numbers

C. Reservation of bandwidth

D. Synchronization of polling intervals

20. What is the IP protocol number for TCP?

A. 1

B. 6

C. 17

D. 89