40555A Networking Fundamentals

Module 5: Implementing IP

Contents

Learning objectives based on MTA exam objectives

#	Lesson title	Learning objectives	Exam objectives mapped
1	Implementing IPv4	• Describe IPv4. • Identify IPv4 address classes. • Explain classful IPv4 addressing. • Differentiate between public and private IPv4 addresses. • Identify reserved addresses. • Configure an IPv4 host.	1.2.2 Addressing 1.2.3 Reserved address ranges for local use (loopback) 3.2.3 Why use Internet Protocol version 4 (IPv4) addressing 3.2.7 Gateway 3.2.10 Reserved address ranges for local use 3.6.7 Reserved address ranges for local use
2	IPv4 Subnetting	• Describe subnets. • Explain the purpose of a subnet mask. • Describe classless IPv4 addressing. • Determine subnet IDs. • Determine hosts in each subnet.	1.2.2 Addressing 3.2.6 Subnet mask
3	Implementing IPv6	• Identify the limitations of IPv4. • Describe the IPv6 address space.	1.2.2 Addressing 3.2.4 IPv4 to IPv6 tunneling protocols to ensure backward compatibility

#	Lesson title	Learning objectives	Exam objectives mapped
		• Identify different types of IPv6 address. • Explain Windows 10's dual IP stack. • Describe IPv4 to IPv6 transition technologies. • Configure an IPv6 host.	3.2.5 Dual stack 3.3.1 Subnetting 3.3.2 Ipconfig 3.3.3 Why use IPv6 3.3.4 Addressing 3.3.5 IPv4 to IPv6 tunneling protocols to ensure backward compatibility 3.3.6 Dual IP stack 3.3.7 Subnet mask 3.3.8 Gateway 3.3.11 Reserved address ranges for local use 3.6.7 Reserved address ranges for local use

Module overview

Earlier in this course, we examined the TCP/IP stack, which Figure 1 depicts. Now it's time to zoom in and talk a bit more about the IP layer. You might remember that IPv4 and IPv6 operate at the Internet layer of the TCP/IP protocol stack (the network layer of the Open Systems Interconnection (OSI) model). This layer is responsible for delivering packets of information between hosts that might (or might not) be in different subnets. And, of course, we're going to talk about subnets in this module as well.

Devices attached to an IP network are assigned a logical network address—an Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6) address. This unique address ensures that other devices can communicate with the device. In this module, we'll examine both IPv4 and IPv6 and learn about the two addressing schemes used by these protocols. A heads up: there's a bit of math in the following pages, so you might want to go and turn on the coffee machine now. Again, Figure 1 depicts a TCP/IP stack:

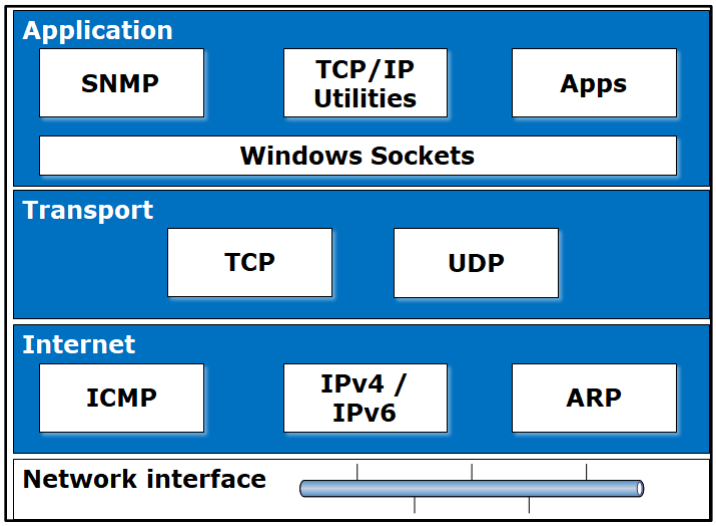

Figure 1. TCP/IP stack

Objectives

After completing this module, you will be able to:

- Implement IPv4.
- Implement IPv4 subnets.
- Implement IPv6.

Lesson 1: Implementing IPv4

IPv4 is the primary protocol used to link devices on intranets, extranets, and the internet. The protocol was devised in the mid-eighties and has changed only slightly in the intervening 35 years. To properly implement and support any network, you should be familiar with IPv4, its addressing scheme, and how to configure devices to use IPv4.

Objectives

After you complete this lesson, you will be able to:

- Describe IPv4.
- Identify IPv4 address classes.
- Explain classful IPv4 addressing.
- Differentiate between public and private IPv4 addresses.
- Identify reserved addresses.
- Configure an IPv4 host.

An overview of IPv4

To configure network connectivity successfully, you should be familiar with IPv4 addresses and understand how they work. Computers can communicate only when they identify each other. When you assign a unique IPv4 address to each networked computer, the IPv4 address identifies the computer to the other computers on the network. That IPv4 address, combined with the subnet mask, identifies the computer's location on the network, just as the combination of a number and a street name identify the location of a house.

Note

Remember that computer devices in the same network share a network address but have a different host address, in the same way that houses have their own unique number even though they share the same street name.

Each IPv4 address is based on a 32-digit binary number. In other words, the address is expressed in 32 bits of binary. If you're an IPv4 addressing newbie, you might be a little daunted by the idea of using binary; but don't worry, it's not too difficult.

Note

A binary number can be either a zero or a one. For example, 101 is a three digit (three bit) binary number. With three digits, you can express eight different combinations: 000, 001, 010, 011, 100, 101, 110, 111.

A 32-bit binary number enables you to express 4,294,967,296 (or around 4.29 billion) different combinations. This means that the IPv4 addressing scheme enables you to express up to 4.29 billion different host IP addresses. However, as we shall learn later, it's not *quite* as simple as that.

To check this, open the Windows 10 Calculator app, as Figure 2 depicts, and switch to Scientific view. Select **2** (the base number), select the **X^Y** button, and then select **32**. Select **Equals**. This displays the number of combinations. We express this as 2^32. Give it a try now!

Although the addressing scheme is based on binary, we don't usually enter IPv4 addresses using binary. Instead, we use decimal (or base 10). Let's run through an example. Here is a 32-bit binary IPv4 address:

11000000101010000000000111001000

We divide the address into four octets (or groups of eight digits), as in the following example:

11000000.10101000.00000001.11001000

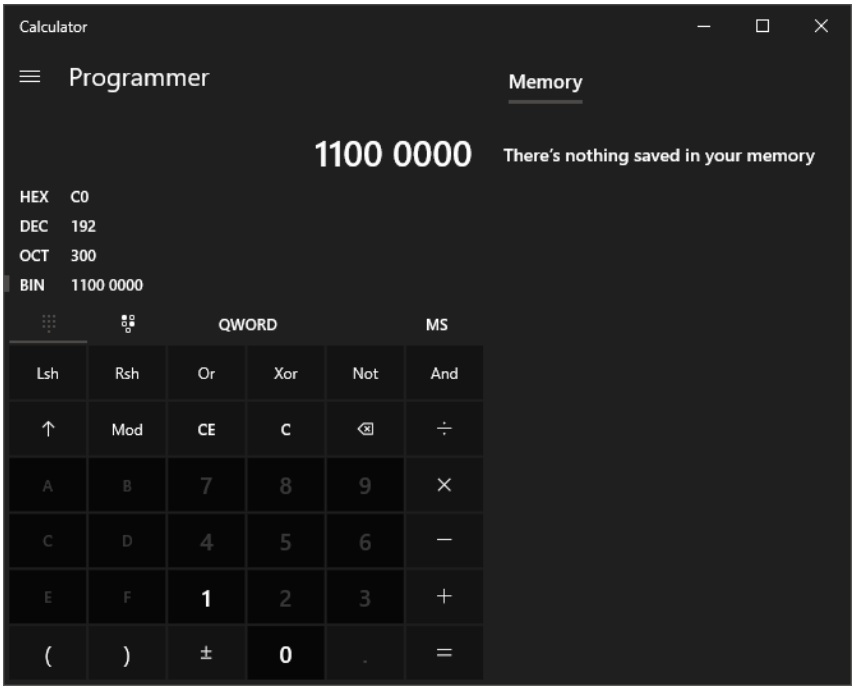

Figure 2. Windows 10 Calculator binary calculation results

Note

Why do we divide the 32-bit number into four octets and convert each one? It's easier to work with four smaller numbers rather than one enormous number. And none of the numbers (in decimal) ever exceeds 255 because that's the largest number you can express with eight bits of binary.

Finally, we convert each number into its decimal equivalent:

192.168.1.200

Note

You can use the Windows 10 Calculator app to do the conversion. Select the **Programmer** view, and then select **BIN**. Enter each of the octets in turn, and you should be able to find the decimal equivalent.

In conjunction with a subnet mask, as Figure 3 depicts, the address identifies:

- The computer's unique identity, which is the host ID
- The subnet on which the computer resides, which is the network ID

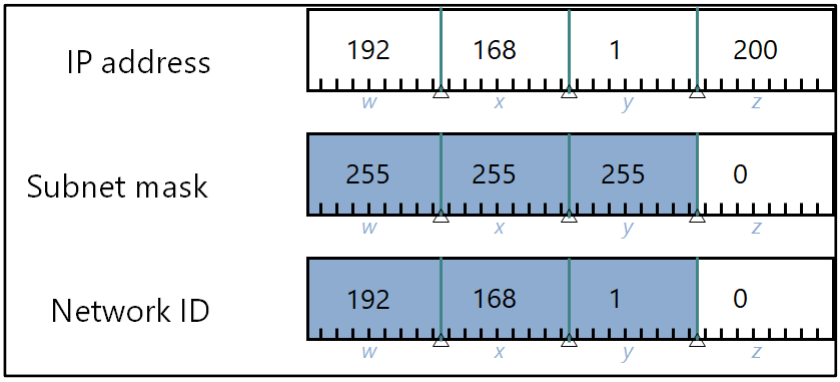

Figure 3. Host IP address, subnet mask, and network ID

The subnet mask, which we'll examine shortly, determines how much of the IPv4 address is the network portion, and how much is the host portion—or going back to our previous example, how much is the street name, and how much is the house number. It should be reasonably obvious that if you have a fixed number of bits—32—then the more bits you allocate to the network the fewer bits remain for each host in each network. So, for example, if you have 20 bits for the network, you only have 12 bits remaining for the hosts. This is important because the number of bits you have determines the number of networks and hosts you can express.

For example, as Figure 4 depicts, if you have 20 bits for each network, you can have 2^20 different combinations of ones and zeros. This results in 1,048,576 different combinations. You have 12 bits left for hosts, and 2^12 is 4,096. So, using an addressing scheme that allocates 20 bits for networks and 12 bits for hosts, you can have about a million networks, each with around 4,000 hosts. This is an over-simplification, but it'll do for now.

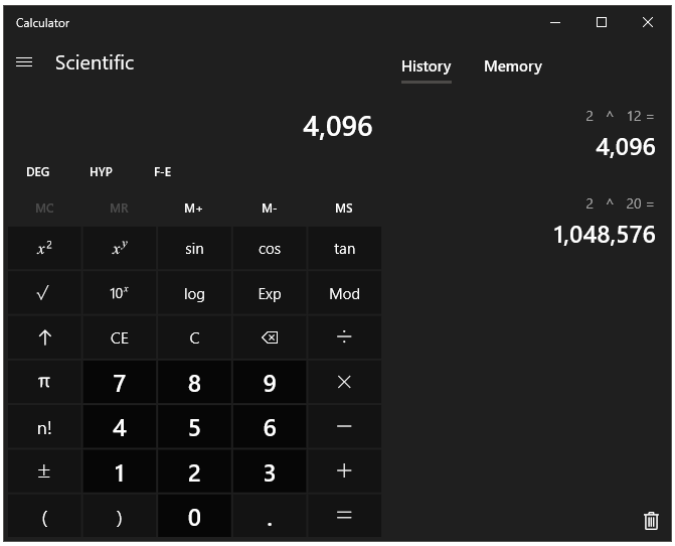

Figure 4. Windows 10 Calculator host calculation results

Address classes

When the IPv4 protocol was being developed, it was decided that it would be a good idea to define a scheme that prescribed the number of bits that could be assigned to networks. This scheme was known as *classful addressing*. For hosts, three classes of address—A, B, and C—were defined: Class D and E were also defined, but we'll talk more about those in a moment.

Class A, B, and C

The IANA (Internet Assigned Numbers Authority) decided that organizations with large networks would use Class A addressing. Those that had small networks would use Class C addressing. This means that organizations whose networks were sized "just right" (or *medium*) would use Class B.

The addressing scheme defined that a Class A network would use a whole octet (eight bits) for the network, leaving three octets (24 bits) for hosts. Class B networks would use two octets (16 bits) for both networks and hosts. The Class C scheme would use the first three octets (24 bits) for the network, leaving only one octet (eight bits) for the hosts.

Now, the interesting questions are:

- When you read an IPv4 address, how do you know what class it is?
- And, how do you know how many octets are network bits and hosts bits?

The folks that designed IPv4 thought of these questions. They defined a standard that based on the very first binary digit in the 32-bit address, a computer device would be either Class A, B, or C (we're still getting to Classes D and E).

Specifically, if the leading digit (or digits) is:

- 0, then the device has a Class A IPv4 address.
- 10, then the device has a Class B IPv4 address.
- 110, then the device has a Class C IPv4 address.

What are Class D and E?

Class A, B, and C addresses are assigned to individual computer devices, whether those devices are Windows 10 computers, iPads, routers, smart phones, or any other type of device. These types of address are sometimes described as being *unicast*. A *unicast address*, illustrated in Figure 5, is assigned to a single device.

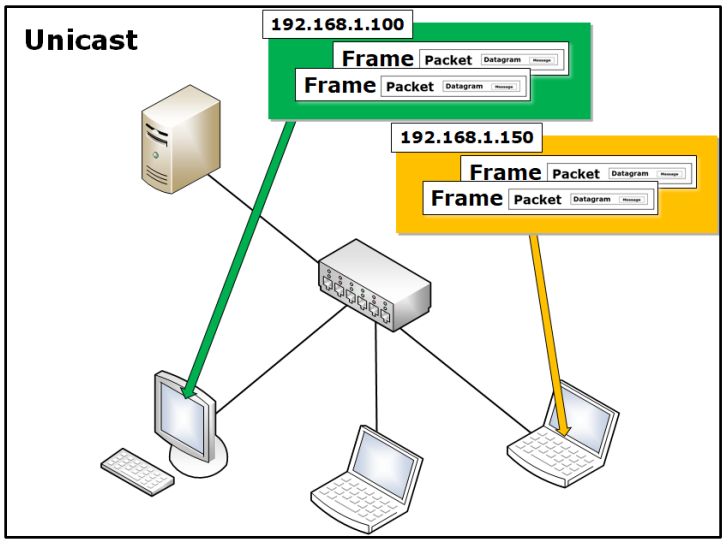

Figure 5. Unicast

But what happens when a server wants to communicate with multiple devices? If the same piece of information must be disseminated to hosts on a network, with a unicast system the server must create a set of packets for each target host, and then merge each set onto the wire for onward delivery to the target hosts. This consumes network capacity (bandwidth), and server processor bandwidth. Figure 6 illustrates a broadcast configuration.

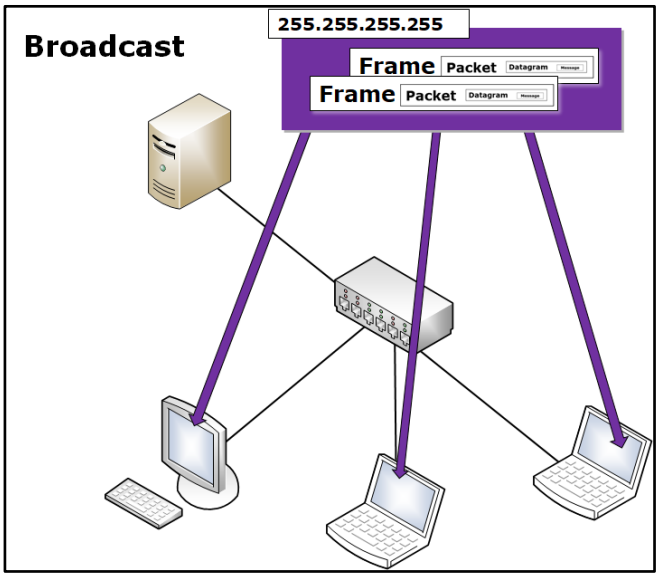

Figure 6. Broadcast

Instead, the server sends a single set of packets to all network devices; this is called a *broadcast*. A broadcast is sent to every device on the network, which reduces the workload on the server (it only needs to create and send one set of packets). It also reduces network bandwidth consumption (again, only needing one set of packets).

However, suppose that only half of the network's devices truly need that information. It's unfortunate, but the way that Ethernet networks function is that each host will get the broadcast and say, "Hey, that's for me." The host will capture a copy of the packet and pass it up the protocol stack to the required application—only for the application to recognize

that the packet and its contents, are not relevant. This wastes processor capacity on those devices that don't need the data. Figure 7 illustrates a multicast configuration:

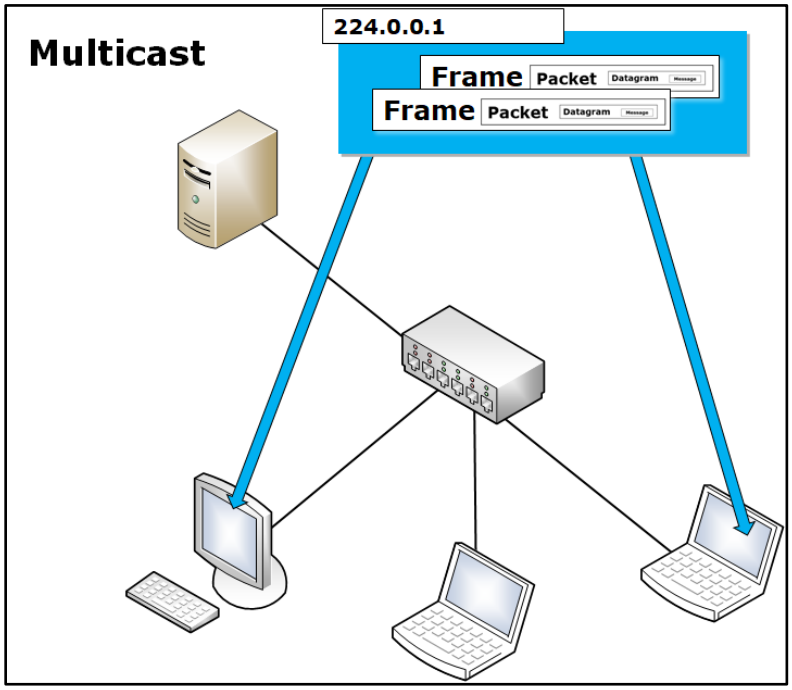

Figure 7. Multicast

This is where a multicast is useful. A *multicast address* is an address that's assigned to multiple, but not all, devices. You instruct a computer to be a part of a multicast group by assigning it a multicast address. Now our server creates one set of packets (network bandwidth and server processor efficient) and merges them onto the wire addressed to the group of devices with a specific multicast address. Any computer that's not part of the multicast group (that is, not configured with the appropriate multicast address), disregards the packets as they pass by on the wire.

We use Class D addresses for multicasting. Class D addresses start with a binary 1110.

Class E addresses start with a binary 11110 and are reserved; in other words, they're not assigned to devices or to multicast groups.

IPv4 classful addressing

In simple IPv4 networks, you can use classful addressing. This is an addressing system that adheres to the standard requiring you to use whole octets to express the network address. For many situations, this works well.

To indicate that you're using the classful scheme, you must use a subnet mask that blocks the entire octet for network use. This means using the decimal number 255 in the appropriate octet. We'll talk more about the subnet mask in the next lesson.

The subnet mask defines full octets as part of the network and host IDs. A 255 represents an octet that is part of the network ID, and a 0 represents an octet that is part of the host ID. Class A, B, and C networks use default subnet masks.

The following table lists the characteristics of each IP address class, including Class D, the multicast group address class.

Class	First octet	Default subnet mask	Number of networks	Number of hosts per network
A	1 to 127	255.0.0.0	126	16,777,214
B	128 to 191	255.255.0.0	16,384	65,534
C	192 to 223	255.255.255.0	2,097,152	254
D	224 to 239	Not defined	N/A	N/A

Notice something interesting about this table? Because the leading digit in a Class A network is a zero, there are no longer eight bits to play with to express your networks; instead, there are only seven. So, in fact, you have half as many networks. Likewise, a Class B network is not expressed with 16 binary digits because the first two must be 10. This means you have only 14 bits to express your networks. With a Class C network, you have 21 bits rather than 24.

Note

We converted the first octet for each address class to a decimal range so we can more easily identify a host's class by which range it falls into. For example, 131.107.0.200 is a Class B address because 131 falls in the range 128 to 191. Therefore, the network address is 131.107.0.0 and the host address is 0.200.

Public vs. private IPv4 addressing

You might remember that at the beginning, we said that the IPv4 32-bit binary address scheme enables us to define 4.29 billion addresses. In 1985, that probably seemed like a lot. At the time, there might only have been a million hosts connected to the early internet. In fact, there are not 4.29 billion addresses. There are some inherent inefficiencies in the system. When we use Classful addressing, we have:

- 126 Class A networks, each with 16,777,214 hosts
- 16,384 Class B networks, each with 65,534 hosts
- 2,097,152 Class C networks, each with 254 hosts

The following table summarizes this.

Class	Number of networks	Number of hosts per network	Total number of hosts
A	126	16,777,214	2,113,928,964
B	16,384	65,534	1,073,709,056
C	2,097,152	254	532,676,608
Total			3,720,314,628

In other words, we've managed to lose 570 million addresses. Well, we didn't actually lose them. Rather, we've rendered them unusable for host addressing. As the Internet grew, and as more organizations and their computing devices were connected, this lack of address space became an increasingly pressing matter. At one point, it was suggested that the Class E range be allocated for use by hosts.

In the end, IANA—which organizes IP standards and addressing—decided on another solution. They designed a division in IPv4 addresses for devices connected directly to the internet, and those connected to an organization's internal network, or *intranet*.

Devices and hosts that connect directly to the internet require a public IPv4 address. However, hosts and devices that don't connect directly to the internet don't require a public IPv4 address; these hosts would be allocated a private IPv4 address by the organization itself.

Public IPv4 addresses

Public IPv4 addresses must be unique. Usually, your Internet Service Provider (ISP) allocates you one or more public addresses from its address pool. The number of addresses that your ISP allocates to you depends upon how many devices and hosts you have connected to the internet. For most organizations, they will require one or more public IPv4 addresses for each router with internet connections, and for services that they want to publish to the internet such as a web server, or email services for employees.

Private IPv4 addresses

Technologies such as network address translation (NAT) enable administrators to use a relatively small number of public IPv4 addresses and enable local hosts to connect to remote hosts and services on the internet. Figure 8 illustrates this configuration:

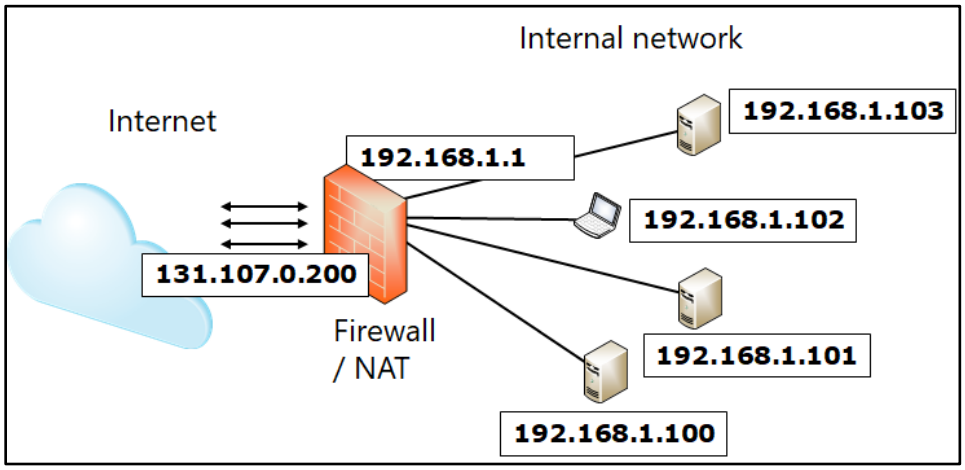

Figure 8. NAT configuration

Note

A NAT-enabled device such as the router at your home with which you connect to the internet maintains a table that identifies which internal hosts (with a private address) requested what resource from which internet-based server. When traffic arrives on the external interface, the NAT knows where the traffic must be routed internally. We'll talk about NAT in more detail later in this course.

IANA defines the following address ranges as private, and internet-based routers don't forward packets originating from, or destined to, these ranges.

Class	Mask	Range
A	10.0.0.0/8	10.0.0.0 - 10.255.255.255
B	172.16.0.0/12	172.16.0.0 - 172.31.255.255
C	192.168.0.0/16	192.168.0.0 - 192.168.255.255

10.0.0.0/8

The 10.0.0.0/8 private network is a Class A network ID that includes the following range of valid IP addresses: 10.0.0.1 to 10.255.255.254. The 10.0.0.0/8 private network has 24 host bits that are available for any subnetting scheme within a private organization.

172.16.0.0/12

The 172.16.0.0/12 private network can be interpreted as a block of 16 Class B network IDs or as a 20-bit assignable address space (20 host bits) that are available for any subnetting scheme within a private organization. The 172.16.0.0/12 private network includes the following range of valid IP addresses: 172.16.0.1 to 172.31.255.254.

192.168.0.0/16

The 192.168.0.0/16 private network can be interpreted as a block of 256 Class C network IDs, or as a 16-bit assignable address space (16 host bits) that are available for any subnetting scheme within a private organization. The 192.168.0.0/16 private network includes the following range of valid IP addresses: 192.168.0.1 to 192.168.255.254.

The result of many organizations' use of private addresses is the reuse of the private address space, which helps prevent the depletion of public addresses. Because the Internet Corporation for Assigned Names and Numbers (ICANN) will never assign IP addresses in the private address space as public addresses, routes in internet routers for private addresses will never exist. Private addresses are not reachable on the internet. Therefore, internet traffic from a host that has a private address must either:

- Send its requests to an application layer gateway with a valid public address, such as a proxy server
- Have NAT translate its private address into a valid public address before sending it on the internet.

Reserved addresses

There are some specific IPv4 addresses and ranges of addresses that you cannot assign to hosts. These are referred to as *reserved addresses*.

Loopback

The host address 127.0.0.1 should, by rights, be a host address in the Class A range because it starts with a leading binary zero. However, this address is reserved for loopback testing. In effect, the address 127.0.0.1 is assigned to all hosts as a host can use this address to verify the integrity of the protocol stack. A local host can also use it to identify itself.

Class D and E

We already discussed Class D and E addresses, but you cannot assign addresses from either of these classes to hosts. In the case of Class D, these are multicast addresses for groups of computers, while Class E addresses are reserved for non-commercial and testing purposes.

APIPA address

Windows 10 computers that are configured to obtain an IP address automatically from a centralized server (which is the default behavior), will often use Automatic Private IP Addressing (APIPA) when they fail to obtain a valid IPv4 address automatically. This address has the decimal prefix 169.254.

A host using APIPA can communicate with other APIPA devices in the same subnet but cannot communicate outside the local subnet. You cannot, or rather, you should not allocate an address in the range 169.254.0.1 to 169.254.255.254 to a host computer.

Note

If you find a device using an APIPA address, verify that it's physically connected to the network, and verify that the device that allocates addresses is online and has available addresses for use.

Configuring an IPv4 host

You can configure an IPv4 host in several ways. However, you typically must define the following properties:

- An IPv4 address to uniquely identify a host
- A subnet mask to define which part of a host's address is the network, and which is the host
- A default gateway to enable communications with devices in other networks
- One or more Domain Name System (DNS) server addresses for name resolution

Static configuration

You can configure static IPv4 configuration manually for each of your network's computers. The disadvantage of static configuration is that it's relatively easy to enter incorrect information resulting in devices being unable to communicate. Static configuration also requires that you visit each computer and input the IPv4 configuration. This method of computer management is time-consuming if your network has more than 10 to 12 computers.

To configure a Windows 10 computer manually, you can use Network Connections in the Windows user interface (UI), a command prompt, or a Windows PowerShell cmdlet.

Using Network Connections

To use the Windows UI to configure IPv4, complete the following steps:

1. Select **Start**, select **Settings**, and then select **Network & Internet**.
2. Select Change adapter settings.
3. In Network Connections, right-click or access the context menu of your network adapter, and then select **Properties**.

4. In the ***network adapter*** **Properties** dialog box, which Figure 9 depicts, select (or press the **Spacebar** and then **Enter**) the option for **Internet Protocol Version 4 (TCP/IPv4)**:

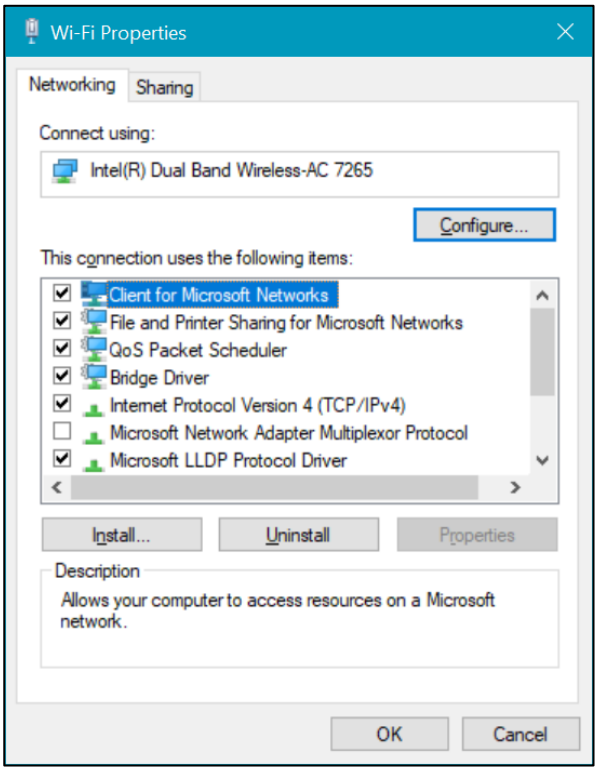

Figure 9. Wi-Fi Properties dialog box

5. In the **Internet Protocol Version 4 (TCP/IPv4)** dialog box, which Figure 10 depicts, select **Use the following IP address**, and then enter the required information:

   - IP address
   - Subnet mask
   - Default gateway
   - Preferred DNS server

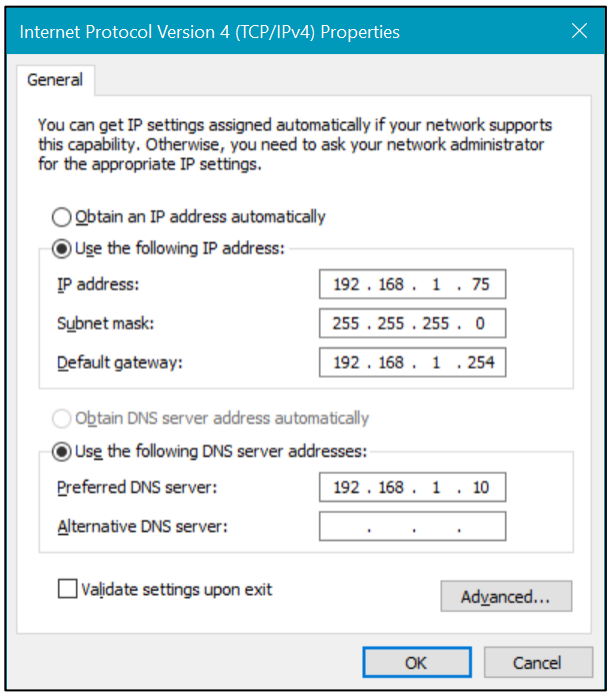

Figure 10. Internet Protocol Version 4 (TCP/IPv4) Properties dialog box

6. Select **OK**, and then select **Close**.

Using the command prompt

To configure the IPv4 settings using the command prompt, use the **netsh.exe** command:

1. Open a **Command Prompt** window.
2. Run the following command, substituting the details for values appropriate for your network:

```
Netsh interface ipv4 set address name="Local Area Connection" source=static
addr=192.168.1.75 mask=255.255.255.0 gateway=192.168.1.254
```

3. Close the window.

Using Windows PowerShell

You can also use Windows PowerShell cmdlets to configure IPv4 settings. To configure the IPv4 settings using Windows PowerShell:

1. Open **Windows PowerShell**.
2. Run the following command, substituting the details for values appropriate for your network:

```
Set-NetIPAddress –InterfaceAlias Wi-Fi –IPAddress 192.168.1.75
```

3. Close Windows PowerShell.

Note

In Windows 10, to reconfigure network settings, you must be able to elevate your user account's privileges to those of an administrator.

Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol (DHCP) version 4 (DHCPv4) enables you to assign IPv4 configurations automatically for a large number of computers without having to assign each one individually. The DHCP service receives requests for IPv4 configuration from computers that you configure to obtain an IPv4 address automatically. It also assigns IPv4 information from a database (known as a *scope*) that you define for each of your network's subnets. The DHCP service identifies the subnet from which the request originated and assigns IP configuration from the relevant scope.

DHCP helps simplify the IP configuration process. However, keep in mind that if you use DHCP to assign IPv4 information, you need to:

- Include resilience in your DHCP service design so that the failure of a single server does not prevent the service from functioning.
- Configure the scopes on the DHCP server carefully, because a mistake can affect the entire network and prevent communication.

To configure a Windows 10 computer automatically, you can use Network Connections in the Windows UI, the command prompt, or a Windows PowerShell cmdlet.

Using Network Connections

To use the Windows UI, use the following procedure:

1. Select **Start**, select **Settings**, and then select **Network & Internet**.
2. Select **Change adapter settings**.
3. In **Network Connections**, right-click or access the context menu of your network adapter, and then select **Properties**.
4. In the ***network adapter*** **Properties** dialog box, which Figure 11 depicts, double-click (or press the **Spacebar** and then **Enter**) the option **Internet Protocol Version 4 (TCP/IPv4)**:

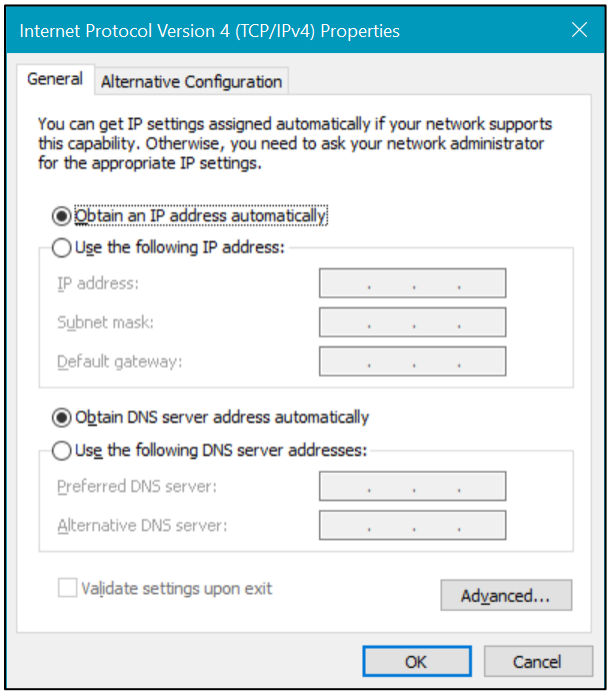

Figure 11. TCP/IPv4 Properties dialog box

5. In the **Internet Protocol Version 4 (TCP/IPv4)** dialog box, select **Obtain an IP address automatically**.
6. Select **Obtain DNS server address automatically**.

7. Select **OK**, and then select **Close**.

Note

To configure non-Windows devices with IPv4 settings, you must use a different procedure. However, all devices require that you configure the same settings: an IP address, a subnet mask, a default gateway, and one or more servers for name resolution. In most cases, devices default to obtain an IPv4 configuration automatically, and so there's little for you to do.

Lesson 2: IPv4 Subnetting

A *subnet* is a network segment. Single or multiple routers separate the subnet from the rest of the network. When your internet service provider (ISP) assigns a network to a Class A, B, or C address range, you often must subdivide the range to match the network's physical layout. Subdividing enables you to break a large network into smaller, logical subnets.

Objectives

After you complete this lesson, you will be able to:

- Describe subnets.
- Explain the purpose of a subnet mask.
- Describe classless IPv4 addressing.
- Determine subnet IDs.
- Determine hosts in each subnet.

What are subnets?

As you discovered earlier in this course, Ethernet is a contention-based network; this means devices compete for available bandwidth. While this isn't a problem on networks with fewer connected devices, it can become an issue as more devices connect. Under load, Ethernet becomes inefficient.

One solution to this is to reduce the number of hosts on a network segment. You can do this by dividing the network into smaller pieces, called *subnets*.

When you subdivide a network into subnets, you must create a unique ID for each subnet. As you might recall, you derive these IDs from the main network ID. To create subnet IDs, you must allocate some of the bits in the host ID to the network ID. By doing so, you can create more networks.

By using subnets, you can:

- Use a single Class A, B, or C network across multiple physical locations.
- Reduce network congestion by segmenting the traffic and reducing broadcasts on each segment. (Broadcasts don't normally transit routers—the devices that separate your subnets)
- Overcome the limitations of current technologies, such as exceeding the maximum number of hosts that each segment can have.

Subnet mask

A subnet mask specifies which part of an IPv4 address is the network ID, and which is the host ID. As we learned earlier, a subnet mask has four octets, like an IPv4 address. The following table reminds us about the default masks.

Class	Default subnet mask (decimal)	Mask expressed in binary
A	255.0.0.0	11111111.00000000.00000000.00000000
B	255.255.0.0	11111111.11111111.00000000.00000000
C	255.255.255.0	11111111.11111111.11111111.00000000

In classful addressing, you use only the default subnet masks, which define a particular number of high-order contiguous bits.

Note

High order just means the bits have a higher value; in other words, the bits on the left are worth more than the bits on the right. Contiguous means *one after the other*; This means you have all ones until you get a zero in the mask, and then you have all zeros.

Default masks use 8, 16, and 24 bits to express the network portion of a Class A, B, and C address, respectively. However, what if you want to break a Class B network down into subnets? One solution would be to use multiple Class C networks using the default mask. Another solution is to vary the number of bits in the mask of a Class B address; to use more than 16 bits, for example. This solution is referred to by one of the following names:

- Classless IPv4 addressing
- Variable length subnet mask (VLSM),
- Classless Inter-domain Routing (CIDR)

When attempting to determine your subnet mask, consider the following:

- Determine the number of physical segments on the network, including any wide area network (WAN) connections.
- Determine the number of hosts required per segment.

Then, assign:

- One subnet mask for your organization.
- A unique subnet ID for each segment.
- A range of hosts for each subnet.

Classless IPv4 addressing

With classless IPv4 addressing, you determine how many subnets you think you're going to need, and then allocate some of the host bits from your address space to accommodate those subnets. Suppose, for example, you required 12 subnets—each to map to a physical floor in a new building. You need to figure out how many binary digits you need to express at least 12.

One way to do this is to convert 12 (in decimal) to binary. If you do this with the Windows Calculator app in Programmer view, you end up with 1100. That's four digits. In fact, using four digits of binary we could express sixteen subnets—so we have space for growth.

Note

How did we get sixteen subnets? Well, if 12 is 1100, which is four digits, the smallest number (in binary) with four digits is 0000, and the largest is 1111. That's zero to fifteen in decimal, or sixteen different combinations.

So, having figured the number of subnets you need, how do you go about actually configuring IPv4 to use your new subnets? One thing you'll need to do is deploy router devices at the boundary of each subnet and configure them with the necessary routing information so that they know how to route packets around your network. (We'll talk about how to do this later in the course.)

But first you'll need to allocate each host on your intranet a unique IP address and subnet mask so that it knows where it is. This means not just what network it's in, but also what *subnet*. You do that by adjusting the subnet mask from its default value, and (in this case) adding four additional bits (because four bits is needed to express 12 subnets).

For example, suppose you had a device configured as follows:

- IP address: 172.16.16.1
- Default mask: **255.255**.0.0
- Default mask (binary): **11111111.11111111**.00000000.00000000
- IP address expressed using CIDR: 172.16.16.1/16
- Subnet address: 172.16.0.0/16

That's a Class B device (it falls within the range 128-191 in the first octet). So, the default mask is 16 bits, or 255.255.0.0.

However, we need four more bits, as Figure 12 depicts. That's 20 bits in total. That changes the configuration:

- IP address: 172.16.16.1
- Default mask: **255.255.240**.0

- Default mask (binary): **11111111.11111111.1111**0000.00000000
- IP address expressed using CIDR: 172.16.16.1/20
- Subnet address: 172.16.16.0/20

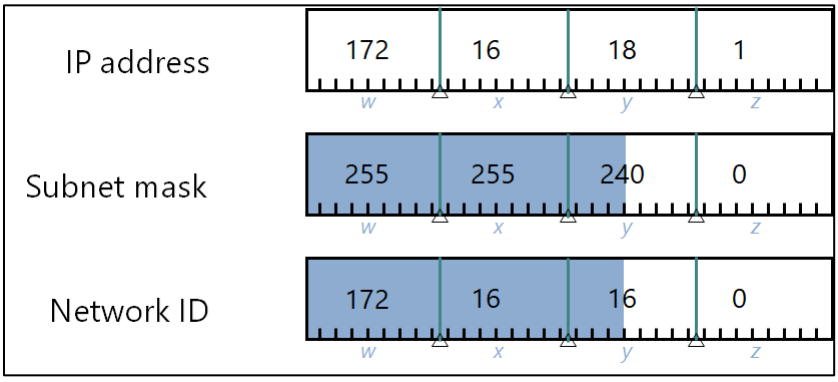

Figure 12. Subnetting more than an octet

The extra four bits in the binary mask are underlined for identification. But what's that 240 all about? Well, 240 is the decimal expression of the binary number **1111**0000. The first four bits (the most significant or *high-order*) are now part of the subnet, and therefore part of the mask. If you convert 11110000 to decimal, it's 240.

Note

Here's something that might be useful: given that a subnet mask only uses high-order contiguous 1's in binary, then any given octet in the subnet mask (expressed in decimal) must be one of the following: 0, 128, 192, 224, 240, 248, 252, 254, or 255. This means that if I need six bits for subnetting a Class B address range, then the third octet (in the mask) will be 252 because that uses six bits.

To summarize this step, when determining your subnet mask:

1. Calculate the number of subnets required.
2. Convert the number into binary. For example, if you require 12 subnets, you should use binary 1100.

3. This requires 4 bits, so convert into high order format. That is, configure the first four bits of the host ID into the subnet ID, thus: 11110000

4. This number is then converted to decimal for the mask, as 240.

Determine subnet IDs

Having figured out that you need to subnet and determined the number of subnets, and then calculated the mask, you must now determine the subnet addresses so you can configure hosts (and routers) to use the sub-netted addresses. The following table demonstrates how to do this for the four-bit example we've been working with. The first subnet has a binary value of 0000 in the first four bits of the third octet. Then 0001, and 0010 and so on. This corresponds to the decimal values in the second column.

Binary mask	Subnet address
00000000	0
00010000	16
00100000	32
00110000	48
01000000	64
01010000	80
01100000	96
01110000	112
10000000	128
10010000	144
10100000	160
10110000	176

Binary mask	Subnet address
11000000	192
11010000	208
11100000	224
11110000	240

Some people maintain that a subnet value of zero and a subnet value of 240 (in this specific example)—that is, a subnet value of all zeros or all ones—is invalid. However, to enable more efficient use of the address space, this is no longer considered the case.

Note

Notice that each subnet is in this case, 16 higher than the preceding subnet. 16, 32, 48, and so forth. Here's a quick way to remember this. If you deduct the subnet mask that you calculated (240 in this example), from 256 (the maximum number of permutations possible with 8 binary bits), you get 16. That's your subnet increment value—and we didn't need to go near binary.

Determine host IDs

The next step is to determine what each of the host addresses is in each of the subnets. To define the addresses within a subnet, the start host address is one binary bit greater than the subnet address. The last host address is two binary bits less than the next subnet address. This is demonstrated in the following table.

Binary mask	Subnet address	Beginning range	End range
00000000	0	x.x.0.1	x.x.15.254
00010000	16	x.x.16.1	x.x.31.254

Binary mask	Subnet address	Beginning range	End range
00100000	32	x.x.32.1	x.x.47.254
00110000	48	x.x.48.1	x.x.63.254
01000000	64	x.x.64.1	x.x.79.254
01010000	80	x.x.80.1	x.x.95.254
01100000	96	x.x.96.1	x.x.111.254
01110000	112	x.x.112.1	x.x.127.254
10000000	128	x.x.128.1	x.x.143.254
10010000	144	x.x.144.1	x.x.159.254
10100000	160	x.x.160.1	x.x.175.254
10110000	176	x.x.176.1	x.x.191.254
11000000	192	x.x.192.1	x.x.207.254
11010000	208	x.x.208.1	x.x.223.254
11100000	224	x.x.224.1	x.x.239.254
11110000	240	x.x.240.1	x.x.255.254

In the example we have here, we're using a Class B address and a subnet of four bits. But the system works whatever the class, and however many bits you decide you need to subnet.

As with the subnets themselves, it was the case that a subnet value of zero and a subnet value of 240 (in this specific example)—or, a subnet value of all zeros or all ones, was invalid. This meant that the hosts in those subnets would also be invalid. However, to enable more efficient use of the address space, this is no longer considered the case.

The final step, of course, is to assign these addresses. Typically, that would mean configuring a DHCP server with the scopes containing the necessary addresses. In a Windows Server environment, it's also wise to configure Active Directory Domain Services (AD DS) sites and subnets. These objects enable client computers to know where they are in the physical network, based on their IP address.

Lesson 3: Implementing IPv6

IPv6 is a technology that helps the internet support its growing user base and the increasing number of IP-enabled devices. As we have learned, IPv4 has been the underlying internet protocol for over 30 years. But the ever-increasing need for new IP addresses is challenging IPv4's robustness, scalability, and limited feature set. IPv6 offers possible solutions to these problems.

Objectives

After you complete this lesson, you will be able to:

- Identify the limitations of IPv4.
- Describe the IPv6 address space.
- Identify different types of IPv6 address.
- Explain the Windows 10 dual IP stack.
- Describe IPv4 to IPv6 transition technologies.
- Configure an IPv6 host.

IPv4 limitations

IPv4 has not changed substantially from the time of its design in the 1980s. While it has proven to be robust, easy to implement, able to support heterogenous networks, and fairly scalable, this protocol faces some serious challenges that almost no one could have predicted decades ago.

Consequently, in 1994, Internet Engineering Task Force (IETF) initiated design and development of a suite of protocols and standards, now known as IPv6. To understand why organizations should seriously consider using IPv6, you need to fully grasp IPv4's limitations.

- Shortage of IPv4 address spaces. The exponential growth of the internet is leading to the impending exhaustion of the IPv4 address space.
- Forced reliance on NAT. IPv4 addresses have become relatively scarce, forcing some organizations to use NAT to map multiple private IP addresses to a single public IP address. While NATs promote the reuse of private address spaces, they don't support standards-based network layer security, or the correct mapping of all higher layer protocols. In addition, NATs can cause problems when connecting two organizations that use the same private IP address space. Additionally, the increasing pervasiveness of internet-connected devices and appliances ensures the eventual depletion of the public IPv4 address space.
- The growth of the internet and the ability of the internet backbone routers to maintain large routing tables. Because of how IPv4 network IDs have been and are currently allocated, there are many tens of thousands of routes in the routing tables of internet backbone routers. This is because the current IPv4 internet routing infrastructure is a combination of both flat and hierarchical routing.
- The need for simpler configuration. You must configure most current IPv4 implementations either manually or through a stateful address configuration protocol such as DHCP. (By stateful, we mean that a database stores the current configuration state for various devices). With more computers and appliances using IP, there's a need for a simpler and more automatic address configuration that doesn't rely on DHCP infrastructure administration. IPv6 can use Stateless Address Auto Configuration (SLAAC) to provide simple plug and play networking.

Note

SLAAC enables IPv6 devices to obtain an IPv6 configuration without using either a manually assigned address or a DHCP server to allocate an IPv6 address. This is because the device recognizes IPv6 router announcements can learn where in the world of IPv6 it is, and therefore what address would be suitable.

- The requirement for security at the IP layer. Private communication over a public medium such as the internet requires encryption services to protect the sent data from being accessed or modified in transit. Although a standard now exists for providing security for IPv4 packets (known as Internet Protocol security (IPsec)), this standard is optional, and not widely implemented.
- The need for better support for real-time delivery of data (also known as *Quality of Service* (QoS)). Some applications require QoS to ensure that datagrams arrive in the correct order without loss.

IETF intentionally designed IPv6 for minimal impact on upper-layer and lower-layer protocols by avoiding the arbitrary addition of new features. Designers of the IPv4 address space didn't recognize that public IPv4 addresses would eventually be exhausted. Because of changes in technology and an allocation practice that didn't take into account the growth in the number of internet hosts, it was clear by 1992 that a replacement solution would be necessary.

When the IPv6 address space was designed, addresses were made 128 bits long so that the address space could be subdivided into hierarchical routing domains that reflect modern-day internet topology. With 128 bits, there are enough bits to create multiple levels of hierarchy and the flexibility to design hierarchical addressing and routing—all features currently lacking in the IPv4-based internet.

Comparing IPv4 and IPv6

The following table highlights additional differences between IPv4 and IPv6.

IPv4	IPv6
Both the routers and the sending host perform fragmentation.	Only the sending host, and not routers, perform fragmentation.
Address Resolution Protocol (ARP) uses broadcast ARP request frames to resolve an IPv4 address to a media access control (MAC) address.	Multicast neighbor solicitation messages replace ARP request frames.

IPv4	IPv6
Internet Group Management Protocol (IGMP) manages local subnet group membership.	Multicast Listener Discovery messages replace IGMP.
Internet Control Message Protocol (ICMP) Router Discovery (which is optional) determines the IPv4 address of the best default gateway.	ICMPv6 Router Solicitation and Router Advertisement messages replace ICMP Router Discovery.
Uses host (A) resource records in the DNS to map host names to IPv4 addresses.	Uses IPv6 host resource records (AAAA records) in DNS to map host names to IPv6 addresses.
Uses PTR resource records in the IN ADDR.ARPA DNS domain to map IPv4 addresses to host names.	Uses PTR resource records in the IP6.ARPA DNS domain to map IPv6 addresses to host names.
Must support a 576-byte packet size (possibly fragmented).	Must support a 1280-byte packet size (without fragmentation).

The IPv6 address space

The most obvious distinguishing feature of IPv6 is its use of much larger addresses. IPv4 addresses are expressed in four groups of decimal numbers, such as 192.168.1.1. Each grouping of numbers represents a binary octet. In binary, the preceding number is:

11000000.10101000.00000001.00000001

An address in IPv6 is four times larger than an IPv4 address. Because the addresses are so large (in binary), IPv6 addresses are expressed in hexadecimal. In decimal, numbers can have a value of 0 through 9; in binary, numbers have values of 0 through 1. In hexadecimal, numbers can have a value of 0 through 15. However, you express the numbers 10 through 15 using a single digit, and so these are expressed as letters A through F. For example, the binary number 110110111000 is expressed in decimal as 3,512, and in hexadecimal as DB8.

In the same way that a 32-bit number is too large to express as a single decimal number, and is divided into four octets, so a 128-bit number is far too large to express in octets. Each address consists of eight groups of numbers, each separated by a colon. This means that each group represents 16 bits of the address. The address space in IPv6 supports billions of addresses. To be specific, 3.4 x10^38. Or, 34 followed by 37 zeros.

Here's an example:

2001:0DB8:0000:2F3B:02AA:00FF:FE28:9C5A

You'll notice that its quite long. In IPv6, we can remove the leading zeros in any 16-bit grouping, thereby shortening the number to:

2001:DB8:0000:2F3B:2AA:FF:FE28:9C5A

We can also remove any 16-bit grouping that contains only zeros, providing we still include the placeholder colon, as in the following example:

2001:DB8::2F3B:2AA:FF:FE28:9C5A

This might seem complex for the average human end users, but the assumption is that they will rely on DNS names to resolve hosts, meaning that they will rarely enter IPv6 addresses manually. The IPv6 address in hexadecimal is also easier to convert to binary. This makes it simpler to work with subnets and calculate hosts and networks.

Additional Reading

For more information on IPv6 addressing, go to IPv6 Addressing.

Types of IPv6 addresses

IPv6 address types are like IPv4 address types. The following list describes these:

- Unicast. An IPv6 unicast address is equivalent to an IPv4 unicast address. You use this address type for one-to-one communication between hosts. Each IPv6 host has multiple unicast addresses. There are three types of unicast addresses:
    - Global unicast addresses. These are equivalent to public IPv4 addresses. They're globally routable and reachable on the IPv6 portion of the internet.
    - Link-local addresses. Hosts use link-local addresses when communicating with neighboring hosts on the same link. For example, on a single-link IPv6 network with no router, hosts communicate by using link-local addresses. *Link-local addresses* are local-use unicast addresses with the following features:
        - IPv6 link-local addresses are equivalent to IPv4 APIPA addresses.
        - Link-local addresses always begin with fe80.

Note

Even if you do nothing with IPv6, your Windows 10 computer will still have at least one link-local address assigned to its network interface(s). To find it, open a Command Prompt window, enter **ipconfig**, and then press Enter. Any address starting with fe80 is a link-local address.

    - Unique local unicast addresses. Unique local addresses provide an equivalent to the private IPv4 address space for organizations, without the overlap in address space when organizations combine.
- Multicast. An IPv6 multicast is equivalent to an IPv4 multicast address. You use this address type for one-to-many communication between computers that you define as using the same multicast address. In IPv4, you might remember that multicast addresses are Class D addresses.

- Anycast. An *anycast address* is an IPv6 unicast address that is assigned to multiple computers. When IPv6 addresses communicate with an anycast address, only the closest host responds. You typically use this address type for locating services or the nearest router.

In IPv4, you typically assign a single host with a single unicast address. However, in IPv6, you can assign multiple unicast addresses to each host. To verify communication processes on a network, you must know the purposes for which IPv6 uses each of these addresses.

Interface identifiers

The last 64 bits of an IPv6 address are the interface identifier. This is equivalent to the host ID in an IPv4 address.

Each interface on an IPv6 network must have a unique interface identifier. Because the interface identifier is unique to each interface, IPv6 uses interface identifiers rather than MAC addresses to identify hosts uniquely.

Prefixes

Like the IPv4 address space, the IPv6 address space is divided by allocating portions of the available address space for various IP functions. The high-order bits (bits that are at the beginning of the 128-bit IPv6 address) define areas statically in the IP space. The high-order bits and their fixed values are known as a *format prefix*.

IANA is responsible for managing IPv6.It also has defined how the IPv6 address space will be divided initially and specified the format prefixes. The following table breaks down the IPv6 address-space allocation by format prefixes:

Allocation	Prefix binary value	Begins with	Fraction of address space
Reserved	0000 0000	N/A	1/256
Global unicast addresses	001	2 or 3	1/8

Allocation	Prefix binary value	Begins with	Fraction of address space
Link-local unicast addresses	1111 1110 1000	FE8	1/1024
Unique local unicast addresses	1111 1100	FD	1/256
Multicast addresses	1111 1111	FF	1/256

The prefix is the part of the address that indicates the bits that have fixed values, or that are the subnet prefix's bits. Prefixes for IPv6 subnets, routes, and address ranges are expressed in the same way as CIDR notation for IPv4.

An IPv6 prefix is written in address/prefix-length notation, for example, 2001:DB8::/48 and 2001:DB8:0:2F3B::/64.

Dual-stack approach in Windows 10

Windows 10 uses IPv6 by default and supports both IPv6 and IPv4 in a dual-stack configuration. The dual IP stack helps reduce maintenance costs by providing the following features:

- Shared transport and framing layer
- Shared filtering for firewalls and IPsec
- Consistent performance, security, and support for both IPv6 and IPv4

When you connect to a new network that advertises the ability to route via IPv6, Windows 10 tests IPv6 connectivity, and it will only use IPv6 if its connectivity is actually functioning. Windows 10 also supports a functionality called address sorting. This functionality helps the Windows 10 operating system determine which protocol to use when applications that support both IPv4 and IPv6 addresses are configured for both protocol stacks.

IPv4-to-IPv6 transition technologies

Right now, we're operating in an internet world that uses both IPv4 and IPv6. An eventual successful transition to IPv6 requires interim coexistence of IPv6 nodes in today's predominantly IPv4 environment. To support this, IPv6 packets are tunneled automatically over IPv4-only routing infrastructures, which Figure 13 illustrates, enabling IPv6 clients to communicate with each other by using Teredo, 6to4, or Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) addresses, and tunneling IPv6 packets across IPv4 networks.

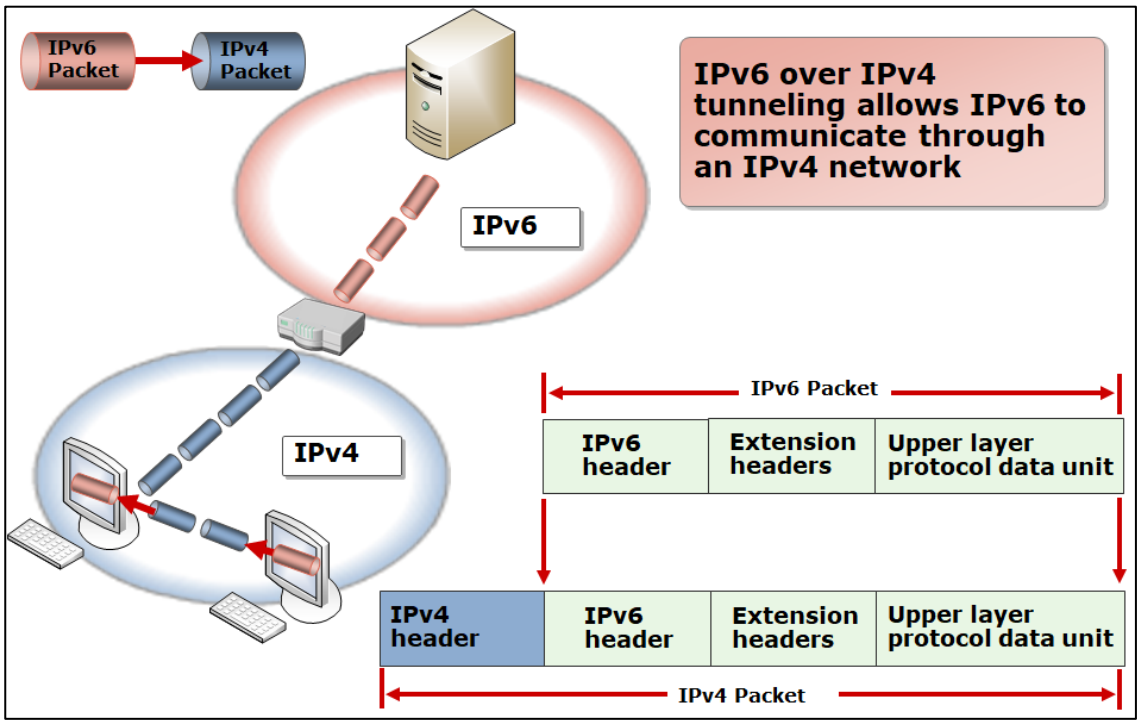

Figure 13. IPv6 over IPv4 tunneling.

The IPv6 transition technologies include:

- ISATAP. *ISATAP* is an address-assignment technology that you can use to provide unicast IPv6 connectivity between IPv6/IPv4 hosts across an IPv4 intranet. ISATAP hosts don't require any manual configuration and can create ISATAP addresses using standard address autoconfiguration mechanisms.

- Local intranets use ISATAP tunneling, which utilizes autoconfiguration. This is the primary way in which IPv6 nodes communicate over an IPv4-only intranet. You mainly use ISATAP within an organizations' site. Although the ISATAP component is enabled by default, it only assigns ISATAP-based addresses if it can resolve the name ISATAP on your network.
- 6to4. 6to4 allows IPv6 hosts with public IPv4 addresses to communicate over the IPv4-only internet. You can use 6to4 to provide unicast IPv6 connectivity between IPv6 sites and hosts across the IPv4 internet. 6to4 treats the entire IPv4 internet as a single link. In a 6to4 address (2002:WWXX:YYZZ:Subnet_ID:Interface_ID), WWXX:YYZZ is the colon-hexadecimal representation of w.x.y.z, a public IPv4 address.
- Teredo. Teredo allows IPv6 hosts with private IPv4 addresses and are located behind NATs to communicate over the IPv4-only internet. Teredo tunneling enables you to tunnel across the IPv4-only internet when the clients are behind an IPv4 NAT.
- Teredo was created because many internet connections use private IPv4 addresses behind a NAT. However, it's a last-resort transition technology for IPv6 connectivity. If native IPv6, ISATAP, or 6to4 connectivity is present between communicating nodes, Teredo is not used. As more IPv4 NATs are upgraded to support 6to4, and as IPv6 connectivity becomes ubiquitous, Teredo will be used less frequently, until eventually it's not used at all.

Configuring an IPv6 host

One highly useful aspect of IPv6 is its ability to configure itself automatically without the use of a stateful configuration protocol, such as DHCPv6. By default, an IPv6 host can configure a link-local address for each interface. By using router discovery, a host can also determine the addresses of routers, additional addresses, and other configuration parameters. The Router Advertisement message includes an indication of whether the host should use a stateful address configuration protocol.

Types of autoconfiguration

Types of autoconfiguration include:

- Stateless. With stateless autoconfiguration, address configuration is based on the receipt of Router Advertisement messages only. Stateless autoconfiguration includes a router prefix but doesn't include additional configuration options such as DNS servers.
- Stateful. With stateful autoconfiguration, address configuration is based on a stateful address configuration protocol such as DHCPv6, to obtain addresses and other configuration options. A host uses stateful address configuration when:
    - It receives instructions to do so in a Router Advertisement message.
    - There are no routers present on the local link.

With stateful configuration, organizations can control how IPv6 addresses are assigned by using DHCPv6. If you need to configure any specific scope options—such as the IPv6 addresses of DNS servers—a DHCPv6 server is necessary. When IPv6 attempts to communicate with a DHCPv6 server, it uses multicast IPv6 addresses. This is different from IPv4, which uses broadcast IPv4 addresses.

- Both stateless and stateful. With both autoconfigurations, configuration is based on receipt of Router Advertisement messages, and on DHCPv6.

You can assign IPv6 addresses to a network interface in any of four ways:

- Manually assigning one or more IPv6 addresses to a network interface
- Using stateful address autoconfiguration by implementing a DHCPv6 server
- Using stateless address autoconfiguration by using Router Advertisement messages
- Using a combination of both stateless and stateful address autoconfiguration

Note:

The operating system always configures a link-local address on a network interface automatically, regardless of whether stateless or stateful address autoconfiguration is in use.

The main difference between IPv6 and IPv4 address assignment is that the IPv6 protocol is designed to be configured automatically. This means that you usually don't need to assign addresses manually or deploy a DHCPv6 server. Instead, you can implement stateless address autoconfiguration for most of your network hosts. Consequently, in contrast to network adapters on IPv4 hosts, which are usually *single-homed* (meaning they have only a single address assigned), most network adapters on IPv6 hosts are *multi-homed* (meaning they have multiple assigned addresses). Specifically, an IPv6 network interface typically has two or more addresses:

- A link-local address, which is automatically generated and used for local link traffic
- An additional unicast address (which might be a global address or a unique local address), and which is used for traffic routed beyond the local link

To configure TCP/IP settings on a Windows 10 computer manually, you can use Network Connections in the Windows user interface (UI), a command prompt, or Windows PowerShell.

Using Network Connections

To use the Windows UI, use the following procedure:

1. Select **Start**, select **Settings**, and then select **Network & Internet**.
2. Select **Change adapter settings**.

3. In **Network Connections**, right-click or access the context menu of your network adapter, and then, as Figure 14 depicts, select **Properties**:

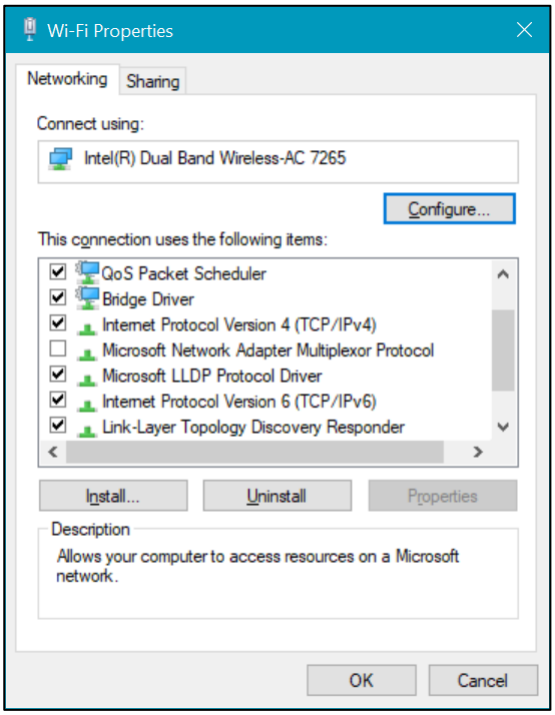

Figure 14. Wi-Fi Properties dialog box

4. In the ***network adapter*** **Properties** dialog box, which Figure 15 depicts, double-click (or press the **Spacebar** and then **Enter**) the option for **Internet Protocol Version 6 (TCP/IPv6)**:

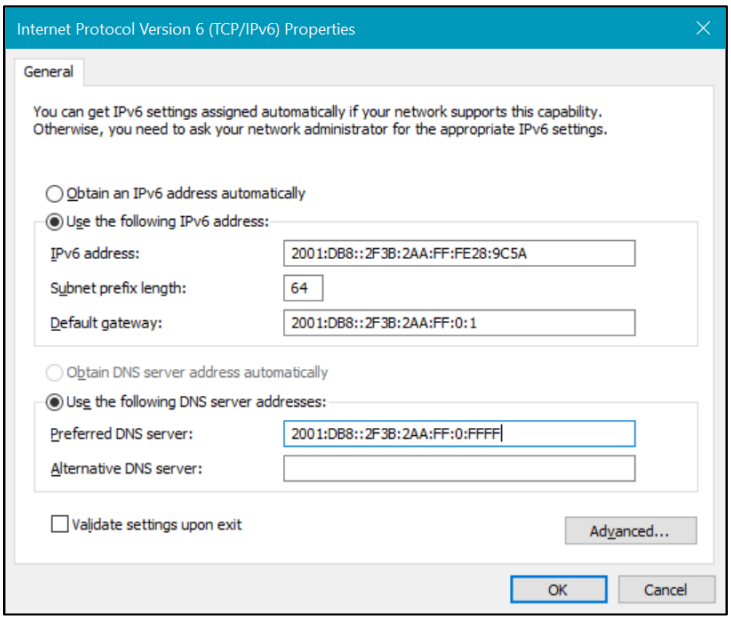

Figure 15. Internet Protocol Version 6 (TCP/IPv6) Properties dialog box

5. In the **Internet Protocol Version 6 (TCP/IPv6)** dialog box, select **Use the following IP address**, and then enter the following information:
   - IPv6 address
   - Subnet prefix length
   - Default gateway
   - Preferred DNS server
6. Select **OK**, and then select **Close**.

Note

It's unlikely that you'll ever need to manually enter an IPv6 address as we depict in this module. It's far more likely that all IPv6 hosts will be configured automatically, by using either DHCPv6 or stateless autoconfiguration.

Learning in action: Implementing IP

Scenario

Lucerne Publishing plans to open a number of new offices and retail outlets around the United Kingdom. Virtually all network communications will be IPv4 based. However, Lucerne Publishing also intends to deploy a new app based on IPv6 that will be installed on all host computers throughout the company. Read the following questions and answer as best as you can to help Lucerne Publishing complete its project.

Questions

1. **One of the first steps you need to take is to configure the new Windows 10 computers to use IPv4. What type of IPv4 address should you use within the retail outlets and branch offices? Choose two of the following:**

   A. You should assign public IPv4 addresses to all hosts at the branch offices and retail outlets.

   B. You should assign private IPv4 addresses to all hosts at the branch offices and retail outlets.

   C. You should assign public IPv4 addresses to all internet-facing interfaces on routers that interconnect the branches and retail outlets.

   D. You should assign private IPv4 addresses to all internet-facing interfaces on routers that interconnect the branches and retail outlets.

2. **You examine a map of the intended new offices and retail outlets across the UK, and learn that there are 22 new locations, each with around a dozen devices. If you're using a Class B network address throughout your organization, how many subnet bits (minimum) do you need to allocate to support this number of locations?**

   A. Three

   B. Four

   C. Five

   D. Six

3. **One of the host computers at the head office has an IP address of 169.254.0.1. What kind of address is this? Choose the best answer:**

   A. A public IPv4 address

   B. A private IPv4 address

   C. A Class B address

   D. An APIPA address

4. **The new publishing app arrives. You now need to deploy this to the computers at the head office. You want to ensure that only computers that need the app will receive it. You also want to minimize network bandwidth during deployment. What kind of network communication would be best for deploying targeted software?**

   A. Unicast communication

   B. Broadcast communication

   C. Anycast communication

   D. Multicast communication

5. **You now need to link the IPv6 app running at the branches and the retail outlets with the database server that supports the app. This database and the client-side apps communicate using IPv6. What tunneling technologies could you use to enable communications between the clients and server?**

   A. ISATAP

   B. Teredo

   C. 6to4

Test your knowledge

1. **Which of the following is a private Class B IPv4 address?**

   A. 172.16.16.1

   B. 192.168.1.254

   C. 131.107.0.200

   D. 2001:DB8::2F3B:2AA:FF:FE28:9C5A

2. **How many decimal numbers can you express with 7 bits of binary?**

   A. 8

   B. 127

   C. 128

   D. 256

3. **How many hosts can you have in a single Class C IPv4 network?**

   A. 1024

   B. 512

   C. 256

   D. 254

Fill in the blanks for the following sentences.

4. **You use Class ( ) network addresses for multicasting in IPv4.**
5. **192.168.0.0/24 is a block of a 254 ( ) IPv4 host addresses.**
6. **A host with the IPv4 address of 172.16.16.1/16 is part of the ( ) /16 subnet.**
7. **True or false: The host with the IPv4 address of 172.16.16.1/20 is in the same subnet as the host with the IPv4 address of 172.16.18.2/20.**

   True

   False

8. **True or false: IPv6 addresses are 128 bits long and are expressed in decimal.**

   True

   False

Read the following scenarios and answer the questions.

9. **Fourth Coffee is in the process of implementing an IPv6 application. Their various coffee shops in New York City are connected over the internet using IPv4 routers.**

   Which IPv4 to IPv6 transition technology would you consider implementing?

10. **Fourth Coffee has transitioned their workplace networks to use IPv6 on Windows 10 computers. The routers that connect Fourth Coffee network infrastructure are IPv6-based.**

    What do you need to do to ensure all the Windows 10 computers are assigned valid IPv6 configurations?

Glossary

Term	Definition
APIPA	Automatic Private IP Addresses are assigned automatically to IPv4 clients that fail to obtain a valid IPv4 configuration from a DHCP server. Addresses start with the prefix 169.254.0.0/16.
Binary	A numbering system that uses base 2; that is, ones and zeros.
Fragmentation	The process of dividing transport layer datagrams—Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)—into manageable pieces at the Internet layer for onward routing.
Hexadecimal	A numbering system that uses base 16. Numbers 0 through 9 and letters A through F are used to express numbers zero to sixteen.
IPv4 classful addressing	A system based on allocating fixed numbers of bits for the network address portion on an IPv4 address. Class A, B and C addresses are available for large, medium, and small networks.
IPv4 Classless Addressing	A system that allocates a variable number of bits to the subnet mask so that a network administrator can decide how many subnets they need.
MAC address	A media access control address is a unique 48-bit binary address (usually expressed in hexadecimal) that identifies a network interface card (NIC). Typically, the MAC address is the serial number of the NIC.
NAT device	Network Address Translation devices enable the use of Private IPv4 addresses within an organization's intranet. The NAT device translates all outbound and inbound packets to ensure that the correct internal address is used.
Protocol stack	Responsible for taking messages from applications and packaging and addressing them for transmission to remote hosts. At the remote end, the protocol stack is responsible for

Term	Definition
	passing the received data up the stack to the appropriate application.
Router	An internetwork device that propagates and receives network packets at layer 3 of the OSI reference model. Routers enable network administrators to separate networks into distinct subnets to help to manage network traffic. Routers are also used to join remote local area networks (LANs) to create WANs. A router is network transport specific; that's to say, it runs a specific network protocol, such as TCP/IP.
Stateful autoconfiguration	A process that allocates an IPv4 or IPv6 configuration to a client computer device. The allocated configuration is stored in a database. The Windows Server DHCP server role can perform this function.
Stateless autoconfiguration	A process that enables a client computer device to obtain a valid IPv6 configuration without necessarily requiring a DHCP server to allocate it a configuration.
Tunnel	A transition technology that encapsulates IPv6 packets in an IPv4 packet for routing over an intranet or the internet between IPv6 hosts. The three IPv4-to-IPv6 transition tunnels commonly used are ISATAP, 6to4, and Teredo.