OBJECTIVES

In computer networks, data communication occurs between a sender endpoint and one or more receiver endpoints, and each endpoint is identified by a unique address at both the data-link layer and network layer. We previously took a look at some details of address representation at the data-link layer. The network layer address is commonly referred to as an Internet Protocol (IP) address, because IP is the de-facto network layer protocol on most networks. More specifically, the most widespread network layer protocol in use today is IP version 4, often written as "IPv4". IP version 6, often written as "IPv6", is designed to be the replacement for IPv4 and has been available for some time; however, IPv6 has yet to replace IPv4 at a global level. The objective of this lab is to learn important IPv4 addressing concepts.

PREREQUISITES

Before attempting this lab, it is best to read the textbook and lecture material for the week. While this lab shows you how to work with IPv4, the lab does not explain in full the framework and theory behind addressing and communication endpoints, with the assumption that you will first read the lecture and textbook material.

LAB SUBMISSION

Use the submission template provided in the assignment inbox to perform the steps requested by this lab. Return to the assignment inbox to submit your lab.

OVERVIEW

In this section, we will learn how to represent IPv4 addresses at the network layer.

We learned previously that computers and devices connect to LANs with network adapters, and that each adapter is assigned its own MAC address. When a computer needs to send a frame to another computer on its LAN, it finds out the MAC address for the adapter on the computer, then addresses the frame to that MAC address. One requirement imposed by Ethernet and Wi-Fi is that the intended recipient must be directly accessible by the sender. That is, the sender will simply address a frame to the receiver under the assumption that the frame has the capacity to reach the recipient directly.

If all computers were connected directly to each other, identifying senders and receivers solely by their MAC address might suffice. However, computers and devices today need to communicate with other computers across cities, states, and even nations, and this type of communication requires functionality beyond that which Ethernet and Wi-Fi provide. One major role of the network layer is to make provision for this type of inter-network communication. IPv4, the current de-facto network layer protocol, makes this provision through a formulaic means of packet forwarding across networks. One integral part of IP's packet forwarding formula is the assignment of a unique address, termed an IP address, to each communication endpoint. In practical terms, this means that each network adapter is assigned an IP address, in addition to a MAC address.

It may be confusing to you as to why a network adapter would be assigned two different, unique addresses. After all, isn't one unique address sufficient? Within the five layer Internet model, the answer is no. The reason lies in the way that layers work. Each layer on the sender communicates only with the same layer on the receiver. That is, the data-link layer on the sender communicates with the data-link layer on the receiver, and the network layer on the sender communicates with the network layer on the receiver. Any address used at the data-link layer is not used by the network layer, and any address used at the network layer is not used by the data-link layer. Otherwise, they would not be two layers, but one. So, we conclude that the address used at the data-link layer is used to uniquely identify senders and receivers at the data-link layer only, and the address used at the network layer is used to uniquely identify senders and receivers at the network layer only.

There is much to learn about the significance and use of the IPv4 address, but for now, we will focus on learning its format and representation. An IPv4 address is 32 bits in length, for example, 10101101111000111111011100101100. As you might expect, as human beings we find it difficult to practically work with binary numbers of this length, and so have adopted a standard representation easier for us to understand. Simply put, we divide the number into 4 sequences of 8-bits, then represent each 8-bit sequence as

a decimal number. Each 8-bit sequence is termed an *octet* in this context, and the decimal number for each octet is divided by a period. For example, the IPv4 address given in this paragraph, 10101101111000111111011100101100, is represented as 173.227.247.44.

This notation is referred to as *dotted decimal* notation, which is a presentation format whereby each of two or more decimal numbers are separated with a period. Note that although the notation for representing *exactly four* decimal numbers, each separated by a period, is *dotted-quad* notation, in practice almost all writers describing IPv4 addresses use the more generic term. To follow suit, the phrase “dotted decimal” will be used throughout the remainder of this lab.

STEPS

1. First, let us practice converting an IPv4 address in dotted decimal notation to binary. You have previously learned how to convert from decimal to binary, so the general methodology is nothing new to you. All we need to do is convert each of the four decimal numbers to binary, using the handy conversion tables below, then concatenate the results. For example, if we have the IPv4 address 192.168.5.33, we convert each number in turn like so.

192 ➔ 11000000

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	0	0	0	0	0
Amount Remaining	192	64	0	0	0	0	0	0	0

168 ➔ 10101000

Power of 2		128	64	32	16	8	4	2	1
Bit		1	0	1	0	1	0	0	0
Amount Remaining	168	40	40	8	8	0	0	0	0

5 ➔ 00000101

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	0	0	0	1	0	1
Amount Remaining	5	5	5	5	5	5	1	1	0

33 ➔ 00100001

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	1	0	0	0	0	1
Amount Remaining	33	33	33	1	1	1	1	1	0

Each number separately thus evaluates to 11000000.10101000.00000101.00100001. When we remove the periods, we can say that the IPv4 address of 192.168.5.33 is 11000000101010000000010100100001 as a raw binary number.

You might wonder why we bother converting addresses in dotted decimal notation to binary. After all, the dotted decimal notation is easier for us to work with. The reason is that computers and network adapters do not view the address in dotted decimal notation, and operate on the raw 32-bit binary number. We will learn some precise ways that computers use to operate on IPv4 addresses, and it is critical that we view and operate on the address in its raw form when doing so.

2. Now, you give it a shot by converting the following IPv4 addresses in dotted decimal notation to binary. Use the table provided to convert each one.

   137.254.120.50
   10.129.10.54

3. We also need some practice in converting IPv4 addresses in raw 32-bit format into dotted decimal notation. We simply reverse the process to do so. We separate the 32-bit number into 4 octets, convert each octet to a decimal number, then separate each decimal number with a period.

   For example, if we are given the IPv4 address 11010000010000000111100110100001, the first thing we need to do is separate each octet like so: 11010000.01000000.01111001.10100001. Then we convert each octet to decimal using the conversion tables below.

   11010000 ➔ 208

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	1	0	0	0	0
Cumulative Amount	0	128	192	192	208	208	208	208	208

   01000000 ➔ 64

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	0	0	0	0	0	0
Cumulative Amount	0	0	64	0	0	0	0	0	0

   01111001 ➔ 121

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	1	1	1	0	0	1
Cumulative Amount	0	0	64	96	112	120	120	120	121

   10100001 ➔ 161

Power of 2		128	64	32	16	8	4	2	1
Bit		1	0	1	0	0	0	0	1
Cumulative Amount	0	128	128	160	160	160	160	160	161

   Now that we know the decimal number for each octet, we simply separate each with a period to arrive out our final dotted decimal notation: 208.64.121.161. So the IPv4

address 11010000010000000111100110100001 is 208.64.121.161 in dotted decimal notation.

4. Now, you give it a try with the following IPv4 addresses displayed in their raw 32-bit format. Convert each to their dotted decimal notation using the same conversion tables as in step 3.

   11010001000011110000110110000110
   11001100010011111100010111001000

5. Even though the raw form of an IPv4 address is a 32-bit number, it is actually composed of two identifiers – a network or subnet identifier, and a host identifier. The network identifier distinguishes the network on which the host resides from other networks, and the host identifier distinguishes the host itself from other hosts on that same network. The number of bits used for each identifier is not the same across all IPv4 addresses, but we know for certain that the number of bits for the network identifier plus the number of bits for the host identifier comes to 32 bits for all IPv4 addresses. We will learn more details of how this works in the remainder of this lab, and in future labs.

   Recall that IPv4 uses a formulaic means of packet forwarding across networks. One integral part of this formula is a way to distinguish one network from another. To state this another way, in order for IPv4 to use its formula to forward packets across networks, it first must clearly identify each computer as being located on a specific network.

   IPv4 identifies a network by utilizing a certain number of the leftmost bits in an IPv4 address as the network identifier. The exact number of leftmost bits to use for the network identifier is specifically configured for each network adapter, and different network adapters may be configured with a different number of bits, depending upon the overall network design. Thus, each adapter is assigned an IP address, and a way to determine the number of leftmost bits that constitute the network identifier in that IP address. For example, the IP address given in step 3 is 208.64.121.161 in dotted decimal notation, or 11010000010000000111100110100001 in binary. If this IP address is assigned to a network adapter, and the network adapter is configured so that the first 20 bits are the network identifier, then the adapter's network is identified with 11010000010000000111.

   One common notation used to concisely specify the IP address, as well as the number of bits used for the network identifier, is to put a slash after the IP address followed by the number of bits. So for the example we just worked through, with 208.64.121.161 as the IP address, and 20 bits for the network identifier, we would represent it as 208.64.121.161/20. To be sure, this notation comes from *Classless Inter-Domain Routing* (CIDR), which is the modern method of allocating IP addresses

across networks. CIDR is a complex subject with many details, and we will learn more details about it in future weeks. For now it suffices for you to know that the IP address, followed by a slash, followed by the number of bits, is a standard way to specify an IP address and the number of bits used for the network identifier.

6. Let us practice using CIDR notation and determining the network identifier from what we learned in step 5. For each of the following, indicate what the network identifier is, in binary. Use the tables used in step 1 to convert each needed octet to binary.

   199.102.234.31/22
   205.178.189.131/18

7. Just as we can represent the 32-bit address in dotted decimal notation, we can also represent the network address in dotted decimal notation. We follow the same process as in step 3, except that we first append 0s to the remainder of the network identifier to fill out the remaining 32-bits. Using the example provided in step 5, 208.64.121.161/20, where the network identifier is calculated to be 11010000010000000111, we need to append 12 0s to it in order to make it 32-bits:

   11010000010000000111000000000000

   Then we convert that number using the same process as in step 3.

   11010000 ➔ 208

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	0	1	0	0	0	0
Cumulative Amount	0	128	192	192	208	208	208	208	208

   01000000 ➔ 64

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	0	0	0	0	0	0
Cumulative Amount	0	0	64	0	0	0	0	0	0

   01110000 ➔ 112

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	1	1	0	0	0	0
Cumulative Amount	0	0	64	96	112	112	112	112	112

00000000 ➔ 0

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	0	0	0	0	0	0
Cumulative Amount	0	0	0	0	0	0	0	0	0

Thus the dotted decimal notation of our network address 11000000101010000000 is 208.64.112.0.

Putting this all together we can say that for the IP address 208.64.121.161, if the network identifier is 20 bits, then its network address is 208.64.112.0.

8. Now it's time for you to try it out representing network addresses in dotted decimal notation. For each of the following examples, extract the network identifier and express it in dotted decimal notation. Keep in mind that for IPv4 addresses given in binary, you will need to extract the network identifier, then follow the process as in step 7. For identifiers given in CIDR notation, you will first need to convert the IP address to binary, extract the network identifier, then follow the process in step 7.

   00001010011101110001011110101010 (25 bits for network identifier)
   74.119.98.141/19

9. Now let us learn a shortcut. If we look at the results from step 7, we see that the network address for 208.64.121.161/20 is 208.64.112.0. Do you notice something interesting about these two? The numbers in the first two octets, 208.64, are the same between both the IP address and the network address. Why would that be? Simple! It's because the network identifier *fully* spans the first and second octets. The first octet consists of bits 1 through 8, and the second octet consists of bits 9 through 16. Since the network identifier spans 20 bits, the first and second octets' bits are all included.

   So, the first shortcut we can take when representing a network address in dotted decimal notation is that we can simply duplicate the octets that are fully spanned by the network identifier. We do not need to calculate them. To take another example, how many octets could we duplicate for 184.173.76.72/26? Simple! We would duplicate the first three, because the first three octets span bits 1 through 24. We only need to calculate the fourth octet.

   So we know that the first three octets of the network address are 184.173.76, and we just calculate the last one. First, we convert 72, the last octet in decimal, to binary.

72 ➔ 01001000

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	0	0	1	0	0	0
Amount Remaining	72	72	8	8	8	0	0	0	0

Then we include the first two bits of this octet, 01, as part of the network identifier. Why two bits? Because the network identifier spans 26 bits, and the first three octets span 24 bits, so that leaves 2 bits remaining. Then we simply append 0s to it so that it's 01000000, and convert that back to decimal.

01000000 ➔ 64

Power of 2		128	64	32	16	8	4	2	1
Bit		0	1	0	0	0	0	0	0
Cumulative Amount	0	0	64	64	64	64	64	64	64

So combining it all together, we copy the first three octets, 184.173.76, then calculate the fourth octet as 64, to arrive at the solution, which is that the network address of 184.173.76.72/26 is 184.173.76.64.

10. Now you give it a try by taking advantage of this shortcut. Represent the network identifier in dotted decimal notation for each of the following. Note that the IP address is the same for each, but the number of bits used for the network identifier differs.

    208.73.211.230/25
    208.73.211.230/23

11. Another important shortcut is zeroing out octets that are not spanned at all by the network identifier. That is, some octets are full spanned by the network identifier, some octets are partially spanned, and some octets are not spanned at all by the network identifier. It is only the partially spanned octets that require us to convert to binary, capture the appropriate bits, append 0s, then convert that back to decimal. Any octets that are not spanned at all are going to be 0.

    For example, with 192.168.245.1/23, we can see that 23 bits fully spans the first two octets, partially spans the third octet, and does not span the fourth octet at all. So, we can simply copy the first two octets, and zero out the last octet, arriving at 192.168.X.0. Now we just need to calculate X.

    So we convert 245 to binary.

245 ➔ 11110101

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	1	1	0	1	0	1
Amount Remaining	245	117	53	21	5	5	1	1	0

Then we extract the first 7 bits, 1111010, then append a 0 to it. 11110100. Last, we convert that number back to decimal.

11110100 ➔ 244

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	1	1	0	1	0	0
Cumulative Amount	0	128	192	224	240	240	244	244	244

Last, we fill in the third (partially spanned) octet with our calculated value to arrive at the network address, which is 192.168.244.0.

You can see how taking these two shortcuts will save you much time when performing these types of calculations, because you only need to calculate the octets that are partially spanned by the network identifier.

12. Now you give it a try by taking advantage of both shortcuts. Represent the network identifier in dotted decimal notation for each of the following. Note that the IP address is the same for each, but the number of bits used for the network identifier differs.

    71.74.42.231/15
    71.74.42.231/24

13. Let us now explore the task of determining whether two IPv4 addresses reside on the same network. Computers using IPv4 must do this every time they need to send a message to another computer, in order to determine whether the recipient is on the same network. This is actually a straightforward task now that we have learned how to extract network addresses from an IPv4 address. All we need to do is extract both network identifiers and compare them. For example, given the following two examples:

    209.18.47.61/19
    209.18.15.61/19

    we will first extract the network identifier from the first, then from the second, and

compare them.

For 209.18.47.61/19, we use our two shortcuts to give us 209.18.X.0 for the network address, and then we figure out X.

47 ➔ 00101111

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	1	0	1	1	1	1
Amount Remaining	47	47	47	47	15	15	7	3	1	0

Extracting the first three bits from this, 001, then appending 0s, gives us 00100000, which in decimal is:

00100000➔ 244

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	1	0	0	0	0	0
Cumulative Amount	0	0	0	0	32	32	32	32	32	32

So the network identifier for 209.18.47.61/19 is 209.18.32.0.

Now we repeat the same process for the second example, 209.18.15.61/19. Because the network identifier fully spans the first two octets, and does not span the last octet at all, we use our two shortcuts to arrive at 209.18.X.0. We then convert the third octet, 15, to binary.

15 ➔ 00001111

Power of 2		128	64	32	16	8	4	2	1
Bit		0	0	0	0	1	1	1	1
Amount Remaining	15	15	15	15	15	7	3	1	0

We then extract the first three bits from 00001111, which are 000, then append 0s to fill out the octet, 00000000. We know that 00000000 in binary is 0 in decimal, so we conclude that the network identifier for 209.18.15.16/19 is 209.18.0.0.

The last step is for us to compare the two network identifiers. We ask ourselves, is 209.18.32.0 the same as 209.18.0.0? The answer is no! Therefore, we conclude that the IPv4 addresses in the two examples – 209.18.47.61/19 and 209.18.15.61/19 – are not on the same network.

14. Now you give the same process a try, determining if two IPv4 addresses are on the same network given the number of bits used for the network identifier. For the following two pairs of examples, determine if they are on the same network as each other. Show your work for full credit. It is desirable to take the two shortcuts described in this lab when appropriate, but when a binary to decimal or decimal to binary conversion is needed, use the tables given in this lab to perform the conversion. While one could use a calculator or an online tool to perform these types of conversions, it is very important that you are able to perform these calculations by hand when necessary. As well, you will not have access to either tool when taking the final exam.

    Pair 1
    209.237.160.185/15
    209.236.15.34/15

    Pair 2
    198.1.104.141/19
    198.1.127.254/19

15. For this final step, you have the chance to incorporate much of what you have learned in this lab. Five IPv4 addresses are given below, some in binary, and some in dotted decimal notation, along with the number of bits used for the network identifier. Your task is to group the addresses that are in the same network. To do so, identify each unique network identifier, then group each address with its corresponding network identifier. Those addresses in the same network would thus be grouped together. Represent all network identifiers and IPv4 addresses in your solution in dotted decimal notation. Show your work for full credit.

    74.208.88.242/20
    01001010110100000101010100001101 (20 bits for network identifier)
    01001010110100000111010111001100 (20 bits for network identifier)
    74.208.117.239/20
    74.208.1.19/20