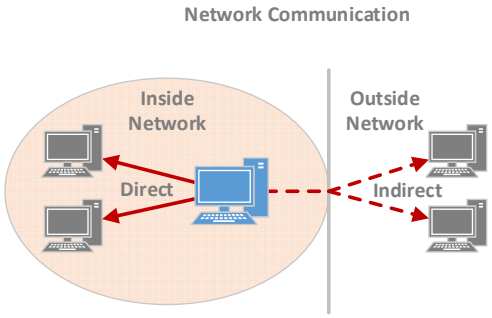
Because IPv4 operates at the network layer, it introduces the concept of a *network* into the five-layer Internet model. A network in this context is casually defined as *a group of devices and computers that communicate directly in the group and indirectly with devices and computers not in the group.* In other words, the quintessence of a network is the virtual boundary that it creates which demarcates direct and indirect communication. This concept is illustrated in the figure below.



Now to be sure, direct communication simply means that the recipient is directly reachable by addressing it at the data-link layer, and indirect communication means that the recipient is not directly reachable at the data-link layer, and can only be reached through routing the packet via layer-3 routers. If networks did not exist, all networked devices would communicate directly with each other, which would be technologically and administratively impractical.

For a computer to send an IPv4 packet to the recipient, it must first determine if the recipient is on the same network as itself. This is because two computers residing on the same network directly communicate with each other by transmitting data-link layer frames addressed to each other, but two computers residing on different networks send the data-link layer frames to a router in order to route the IPv4 packet to the recipient's network.

Humans tend to use one methodology to determine if a recipient is on the same network, while computers use another. As we learned previously, computers and computer networks deal with addresses in raw bits, while we as human beings find it easier to represent the addresses in other notations. It follows that we find it easier to determine if two IPv4 addresses are on the same network using a different method than computers.

In this lab you will learn to work with fundamental concepts of IPv4 networks – network identifiers and network addresses. You will also learn the human and computer methodologies of comparing networks.

LAB OBJECTIVES

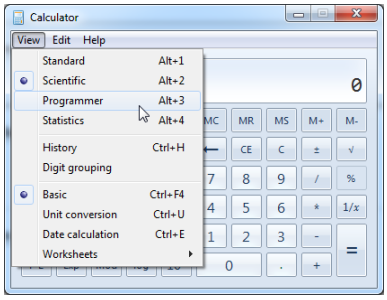
The objectives of this lab are:

- to teach you the basic concepts of networks, network identifiers and network addresses as defined by IPv4.
- to give you a working knowledge of using Classless Inter-Domain Routing (CIDR) notation with IPv4 addresses and its network identifier.
- to give you a working knowledge of determining whether two IPv4 addresses are on the same network.
- to give you a working knowledge of how computers determine whether two IPv4 addresses are on the same network.

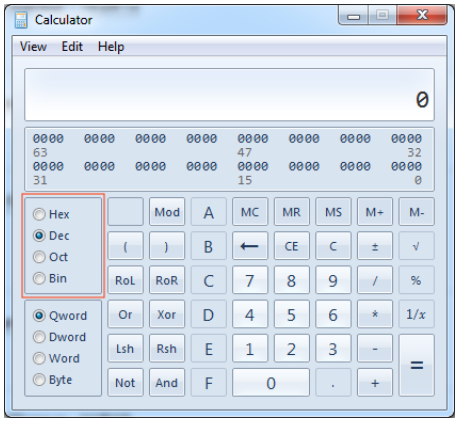
BINARY NUMBER CONVERSION WITH A CALCULATOR

In lab 1 you learned how to manually convert each decimal number into a binary number and vice versa. It is important that you know how to perform these calculations by hand. However, when working in the real world and in academia, we often use tools, most commonly a calculator, to do many conversions for us. Think of the process you have gone through as akin to learning how to perform basic addition, subtraction, multiplication, and division by hand so that you understand the concepts and can do so when no tool is available, but after you understand how to do so you often use a calculator to save yourself some time.

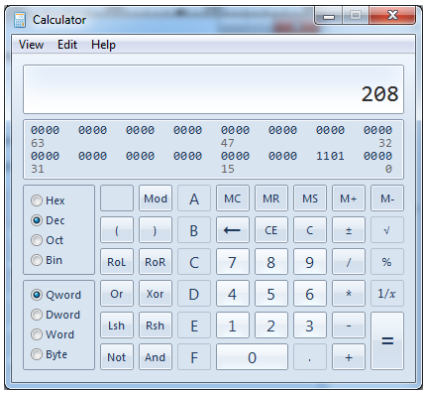
For example, in modern versions of Windows, one can switch the built-in calculator into "Programmer" mode like so.

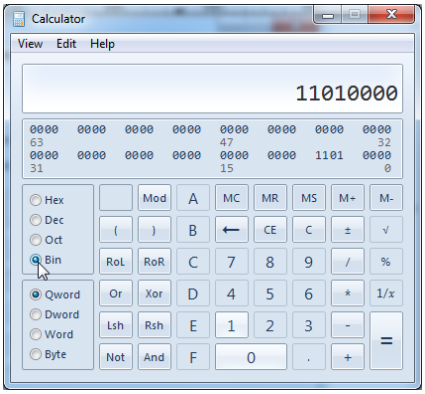


After doing so, you will notice that there are options to switch the calculator to a base other than base 10, including Hexadecimal (Hex), Octal (Oct), and Binary (Bin).

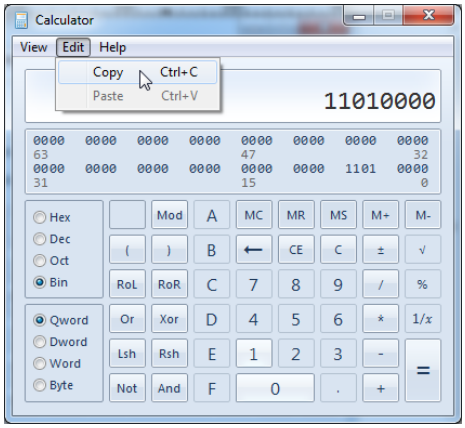


If you enter a number in one base, then switch to the other base, it will automatically convert the number for you. For example, to convert the first octet of the IPv4 address 208.64.121.161, we would enter 208 as decimal, then click “Bin”, to show its binary representation.





We can then simply use the Edit/Copy feature to paste it into our document.



Keep in mind while doing so that *the calculator removes any leading 0 bits*, so you will need to manually add leading 0 bits yourself. In other words, the calculator does not know we are specifically working with octets, and so it will not bother putting any leading 0 bits since leading 0 bits do not change the number's numeric value.

In this lab and in future labs, you are free to use a tool to perform these binary to decimal or decimal to binary conversions for you. However, please do keep in mind that you will not want to forget how to convert by hand.

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.

Section One - IPv4 Network Identifiers and Addresses

Overview

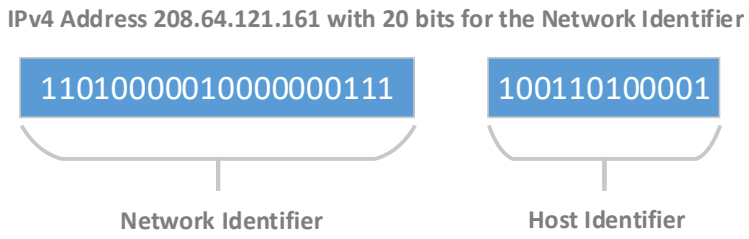
Even though the raw form of an IPv4 address is a 32-bit number, it is actually composed of two identifiers – a network or subnet identifier, and a host identifier. The network identifier distinguishes the network on which the host resides from other networks, and the host identifier distinguishes the 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 network identifiers and addresses in this section.

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 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.

Steps

1. Let us work through an example. Imagine we have an IP address of 208.64.121.161 in dotted decimal notation, or 11010000 01000000 01111001 10100001 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. That is, starting from the left, we count over 20 bits, and each of those bits is part of the network identifier. The remaining 12 bits to the right, 100110100001, are used to uniquely identify the host within the context of the network. This is illustrated in the following figure.



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.

2. Let us practice using CIDR notation and determining the network identifier from what we have learned thus far. For the following, indicate what its network identifier is, in binary.

   199.102.234.31/22

3. A *network address* is simply an IPv4 address which consists of a network identifier followed by all 0 bits for the host identifier. We can represent network addresses in dotted decimal notation the same as any IPv4 address. Using the address provided in step 1 with a 20 bit network identifier, 208.64.121.161/20, we calculate the network identifier to be 11010000010000000111, so we need to append 12 0s to it in order to turn it into a network address:

   11010000 01000000 01110000 00000000

   Then we convert that IPv4 address to dotted decimal notation. Below we use the by-hand approach, but we could also use a calculator.

   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.

4. 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 address and express it in dotted decimal notation. Keep in mind that for IPv4 addresses given in binary, you will need to calculate the network address in binary, then convert the result to dotted decimal notation. For identifiers given in CIDR notation, you will first need to convert the IP address to binary, calculate the network address, and then convert the result to dotted decimal notation.

   00001010 01110111 00010111 10101010 (25 bits for network identifier)
   74.119.98.141/19

5. Now let us learn a shortcut. If we look at the results from step 3, we see that the network address for 208.64.121.161/20 is 208.64.112.0. Do you notice something in common between the IPv4 address and its network address? The numbers in the first two octets, 208.64, are the same! 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.

Let us look at an example of using the shortcut. 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. We convert 72, the last octet in decimal, to binary, by hand, but we could also use a calculator.

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 the network address is thus 184.173.76.64.

To summarize the steps we just followed:
a. we copied the first three octets, 184.173.76 since they were fully spanned
b. converted the last octet, 72, to binary, which is 01001000
c. extracted the first two bits of that number, which are 01
d. appended 0s to the bits to arrive at 01000000
e. converted that back to decimal, which is 64
f. arrived at the network address for 184.173.76.72/26, which is 184.173.76.64.

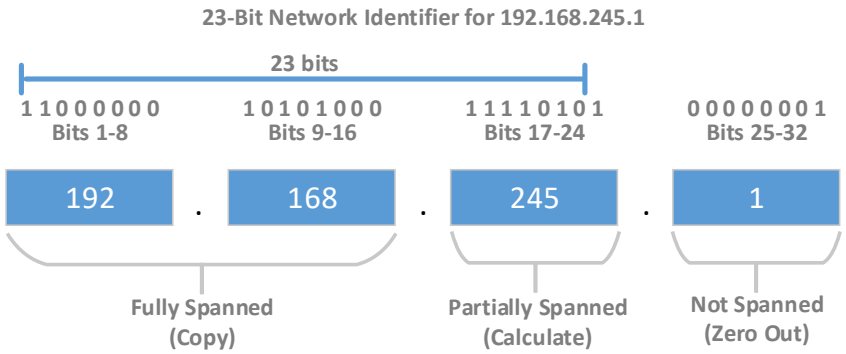
6. Now you give it a try by taking advantage of this shortcut. Represent the network address in dotted decimal notation for the following.

208.73.211.230/25

7. Another important shortcut is zeroing out octets that are not spanned at all by the network identifier. Any octets that are not spanned at all by the network identifier can simply be set to 0 without the need for calculation.

   Some octets may be fully spanned, some octets may not be spanned at all, and at most one octet may be partially spanned. If the length of the network identifier is 8, 16, or 24, then there is no partially spanned octet, and each octet is either fully spanned or not spanned at all. If the length of the network identifier is any other value, there is one partially spanned octet. We only need to convert to binary, capture the appropriate bits, append 0s, then convert that back to decimal, for the partially spanned octet.

   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. This is illustrated in the following figure.



   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.

8. Now you give it a try by taking advantage of both shortcuts. Represent the network address in dotted decimal notation for the following, making sure to take advantage of the shortcut mentioned in step 5 and in step 7.

   71.74.42.231/15

Section Two - Network Comparison by Humans

OVERVIEW

As we learned previously, computers and computer networks deal with addresses in raw bits, while we as human beings find it easier to represent the addresses in other notations. It follows that we find it easier to determine if two IPv4 addresses are on the same network using a different method than computers. You will learn how human beings do so in this section, and will learn how computers do so in Section Three.

STEPS

9. Determining whether two IPv4 addresses reside on the same network is actually a straightforward task now that we have learned how to extract a network address from an IPv4 address. All we need to do is determine both network addresses then compare them.

   Given the following two examples:

   209.18.47.61/19
   209.18.15.61/19

   we will first determine the network address for the first, then for the second, and compare them.

   For 209.18.47.61/19, we use the two shortcuts given in steps 5 and 7, which gives us 209.18.X.0 for the network address. We now need to determine 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	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➔ 32

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 address 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.

10. 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 pair, determine if they are on the same network as each other. Show your work for full credit, except of course for any binary-to-decimal or decimal-to-binary conversions which can be performed with a calculator.

    209.237.160.185/15
    209.236.15.34/15

11. 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 address, then group each address with its corresponding network address. Those addresses in the same network would thus be grouped together. Represent all network addresses and IPv4 addresses in your solution in dotted decimal notation. Show your work for full credit, except of course for any binary-to-decimal or decimal-

to-binary conversions which can be performed with a calculator.

74.208.88.242/20
01001010 11010000 01010101 00001101 (20 bits for network identifier)
01001010 11010000 01110101 11001100 (20 bits for network identifier)
74.208.117.239/20
74.208.1.19/20

Section Three - Network Comparison by Computers

OVERVIEW

Computers do not have the need to represent the addresses in decimal notation for this comparison, and instead use a *subnet mask* and the *bitwise AND* operator to calculate the network addresses of each IPv4 address for comparison. You will learn more about both of these in the steps below.

STEPS

12. A subnet mask is a 32-bit binary number that has the 1 bit set for each bit that corresponds to the network identifier for its associated IPv4 address, and the 0 bit set for each bit that corresponds to the host identifier. For example, if the length of the network identifier for an IPv4 address is 20 bits, then the corresponding subnet mask would be a 32-bit binary number consisting of 20 1s followed by 12 0s, like so:

    11111111 11111111 11110000 00000000

    The number of bits for the network identifier directly determines the number of 1s and 0s that appear in the subnet mask. More specifically, when configuring a network adapter, the subnet mask itself is what the computer uses to calculate the network address corresponding to an IPv4 address.

    If we are given a CIDR entry, say, 198.255.255.32/24, what would the subnet mask be? Simple! We just list out 24 1s followed by 8 0s.

    11111111 11111111 11111111 00000000

13. For the CIDR entry below, list its subnet mask as a binary number.

    10.2.9.33/17

14. Just as IPv4 addresses are commonly represented in dotted decimal notation, so are subnet masks. We can follow all of the same steps as we did when converting IPv4 addresses from binary to dotted decimal. You are already well familiar with this type of conversion, though you should also be aware of a shortcut we can use. If an entire octet consists of all 1 bits, we do not need to manually convert that octet, because we know its decimal representation is 255.

    A subnet mask will always consist of at least three octets that are fully spanned by 1 bits, or fully spanned by 0 bits, and we know the equivalent decimal numbers are 255

and 0, respectively. In the event that the length of the network identifier is not equal to 8, 16 or 24, then there will be one octet that is partially spanned with some 1s and some 0s. And it is only the partially spanned octet that we need to convert to decimal to correctly represent the subnet mask in dotted decimal notation.

Let's take an example, 208.64.121.161/14. From this CIDR entry, we know that the network identifier spans 14 bits, and the corresponding subnet mask can be represented as follows in binary.

11111111 11111100 00000000 00000000

Notice that the 1s fully span the first octet and partially span the second, and that the third and fourth octets consists of all 0s. We can now represent the full subnet mask simply by following the shortcut mentioned above, and converting the second octet to decimal. The shortcut tells us that the mask is 255.X.0.0, so we need only to determine X to obtain the full dotted decimal notation.

11111100 ➔ 252

Power of 2		128	64	32	16	8	4	2	1
Bit		1	1	1	1	1	1	0	0
Cumulative Amount	0	128	192	224	240	248	252	252	252

Replacing X with 252, the subnet mask is 255.252.0.0.

You may have noticed something interesting when converting the partial octet. Since the 1 bits must appear consecutively followed by consecutive 0 bits, there are actually only nine possibilities for any given octet in a subnet mask, as defined in the table below.

00000000 = 0 10000000 = 128 11000000 = 192 11100000 = 224
11110000 = 240 11111000 = 248 11111100 = 252 11111110 = 254
11111111 = 255

If we remember these nine conversions, we do not even need to perform a manual conversion of the partial octet.

15. You give it a try with the following CIDR entry.

23.3.96.50/19

Represent the subnet mask in dotted decimal notation following the methodology used in step 14. Use the lookup table below to convert the partial octet for the given

CIDR entry (if any).

00000000 = 0	10000000 = 128	11000000 = 192	11100000 = 224
11110000 = 240	11111000 = 248	11111100 = 252	11111110 = 254
11111111 = 255			

16. Now that we understand subnet masks, we can now learn about the bitwise AND operator, which is what computers use to calculate the network address corresponding to an IPv4 address. So that we understand the phrase better, let us break the phrase "bitwise AND operator" down piece by piece. First, an *operator* transforms one or more items into a new item. For example, the plus operator in mathematics, represented by the "+" sign, can be used transform two numbers into a new number by adding them together. A *bitwise* operator, as its name suggest, operates on individual bits of binary numbers. A bitwise *AND* operator compares each bit of the first binary number to the corresponding bit in the second binary number, and yields a 1 if both bits are 1, and a 0 otherwise. The bitwise AND operator is often represented mathematically with the "&" symbol.

    You may find useful the following table of all possible bitwise AND operations between two bits:

Bit A	Bit B	A & B
1	1	1
1	0	0
0	1	0
0	0	0

    Let us view a simple example, followed by a more complex example. If we were to apply the bitwise AND operator to two binary numbers – 1101 and 0101 – the result would be computed as follows.

```
  1101
& 0101
----------
  0101
```

    Did you catch what just happened? Starting from left to right:

    1 & 0 = 0
    1 & 1 = 1
    0 & 0 = 0
    1 & 1 = 1

The result is thus 0101 from this bitwise AND operation.

17. Now you give it try with the following two examples. Calculate the result of the bitwise and for each example.

```
  1010
& 0101
```

```
  1111
& 0111
```

18. Why does any of this matter? It matters because computers and devices perform the bitwise AND operation each time they need to send an IP packet, in order to extract the network address from the IPv4 address. They then compare their own network address to the destination network address in order to determine whether the destination node is on its same network. In the world today, there are literally billions of networked computers and devices that regularly perform the bitwise AND operation in order to communicate!

    Let us practice by using the example CIDR entry from step 14, 208.64.121.161/14. We know that computers deal with addresses and subnet masks in raw binary, so we need to convert both to binary. We know the subnet mask in binary is:

    11111111 11111100 00000000 00000000

    by simply listing out 14 1 bits followed by 0 bits.

    The address 208.64.121.161 in binary is:

    11010000 01000000 01111001 10100001

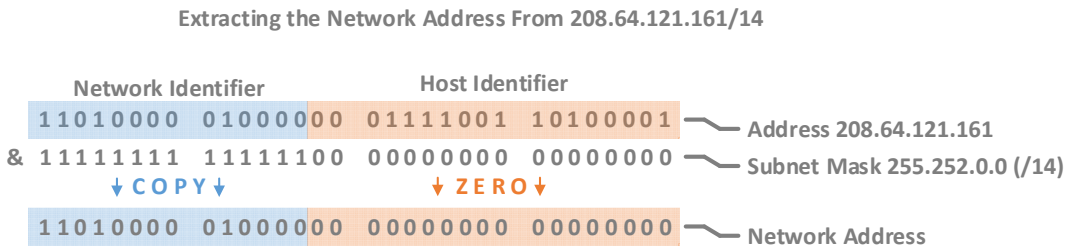
    We can now perform the bitwise AND operation between the IPv4 address and it subnet mask using the methodology presented in step 16.

```
  11010000 01000000 01111001 10100001
& 11111111 11111100 00000000 00000000
-----------------------------------------------------------
  11010000 01000000 00000000 00000000
```

    The result of the bitwise AND operation is the network address. Why? By definition of the bitwise AND operator, anywhere the subnet mask has a 1 bit, the corresponding bit from the IPv4 address is effectively copied. Thus the network identifier is copied into the result. Anywhere the subnet mask has a 0 bit, it does not matter what the corresponding bit in the IPv4 address is, because the resulting bit is

always 0. Thus the host identifier is zeroed in the result. The overall result of the bitwise AND operation is the network address.

This concept is illustrated in the following figure.



Thus, the subnet mask and bitwise AND operation are used by the computer to copy the network identifier from the IPv4 address, and zero out the host portion of the IPv4 address, the result being the network address of the IPv4 address. *Study the previous figure and think about the previous sentence until you are sure you understand it, as the concept they express is integral to understanding the value and purpose of a subnet mask.*

19. Now you have a chance to perform the same operations as computers and devices to calculate the network address. See the CIDR entry below:

    137.254.120.47/25

    Using the same methodology as in step 18, calculate the network address for this entry. The steps you need to perform are to:
    a. convert the address in the CIDR entry to binary.
    b. calculate the subnet mask for this entry.
    c. perform the bitwise AND operation between the two to calculate the network address in binary.

    In addition to binary, show the network address in dotted decimal notation after it has been calculated.

20. Now that we have tried out calculating the network address using a subnet mask, mirroring what computers do, we can take the next step of comparing the network addresses to each other to see if they are on the same network.

    What is the significance of being on the same network versus being on a different network? If the sender and receiver are on the same layer-3 network, the sending computer knows that, at the data-link layer, it can directly address the recipient. Otherwise, the sending computer will address the data-link layer frame to a router that knows how to route the packet to the recipient's network. This behavior

illustrates and emphasizes that *all computers and devices must utilize the data-link layer to send all communications*. An IPv4 packet cannot be sent without first being embedded in a data-link layer frame.

The data-link layer knows nothing about networks, and is only capable of sending frames to directly reachable network adapters. It is up to the sending node to determine to whom the data-link layer frame must be addressed. It is for this reason that each time a computer needs to transmit an IPv4 packet, it must first determine if it will transmit the packet directly to the recipient on the same network, or if it will transmit it to a router which will route the packet to another network. The data-link layer will not perform this logic on behalf of the sending computer.

If a computer's IPv4 address and network identifier length were given as 192.168.254.5/23, and it intended to send to another node with IPv4 address and length 192.168.255.39/23, how would the computer know if the two are on the same network? Simple! It would calculate the network addresses for both and compare them. If the network addresses are equal, they are on the same network; otherwise, they are not.

So we start with the sender's entry, 192.168.254.5/23, and utilize the subnet mask and the bitwise AND operation to determine the sender's network address.

```
   11000000 10101000 11111110 00000101
& 11111111 11111111 11111110 00000000
-----------------------------------------------------------
   11000000 10101000 11111110 00000000
```

And now we do the same for the recipient's entry, 192.168.255.39/23, to determine the recipient's network address.

```
   11000000 10101000 11111111 00100111
& 11111111 11111111 11111110 00000000
-----------------------------------------------------------
   11000000 10101000 11111110 00000000
```

Now we simply compare both network addresses.

```
11000000 10101000 11111110 00000000
11000000 10101000 11111110 00000000
```

When we look through them bit by bit, we see that they are in fact equal. Thus, 192.168.254.5/23 and 192.168.255.39/23 are on the same network.

21. A sending computer will often compare its own network address to hundreds or even thousands of other network addresses, because computer users and applications are oftentimes used to communicate to hundreds or thousands of other nodes. For example, how many different websites do you visit in a week? Probably plenty! As well, the IP address and subnet mask on a network adapter does not typically change often, so a sending node usually finds itself comparing its own network address to many other network addresses.

    Now you have a chance to try out the methodology used in step 20 with a similar scenario. Imagine that that the only network adapter on a computer has an IP address of 63.240.110.201 and that its network identifier spans 21 bits – 63.240.110.201/21. Further imagine that the computer wants to send a message to some other computers with the following CIDR entries.

    63.240.110.219/21
    63.240.104.12/21
    63.240.102.35/21
    67.119.61.37/21

    First calculate the sending computer's subnet mask and network address by using the methodology listed in step 20. Then, for each of the CIDR entries above, use the methodology listed in step 20 to determine if each entry is on the same network or a different network than the sending computer.

22. In a paragraph or two, explain in your own words how the use of a subnet mask and the bitwise AND operation enables a computer to determine if another IPv4 address is on the same network, and why this method is well suited to computers.

Congratulations! You have learned how billions of computers and devices determine where to send a data-link layer frame when they intend to communicate with another node at the network layer.