SEC301 | INTRODUCTION TO CYBER SECURITY
GIAC Information Security Fundamentals (GISF)

301.2

Computer Function and Networking

SEC301.2 | Introduction to Cyber Security

SANS

Computer Function and Networking

This page intentionally left blank.

Module 5: How Computers Work

- Understand Numbering Systems
  - Decimal/Binary/Hexadecimal
- ASCII Basics
- Computer Math Primer
- Computer operation
  - Hard Drive vs. RAM

COURSE ROADMAP

- **Module 5: How Computers Work**
  - Lab 2.1: Converting Number Systems and Decoding ASCII
- Module 6: Networking 101
- Module 7: Networking 102
  - Lab 2.2: Networking

Many diagrams in this material are difficult to read when printed as slides in the book.

These are enlarged in the notes.

Module 5: How Computers Work
In this module, we look at some concepts that are fundamental to understanding computers, specifically different numbering systems used in the Information Technology (IT) field. We also learn to convert numbers from one numbering system to another.

With an understanding of numbering systems, you can then understand how computers encode information to make it more convenient for human beings. The most common method of this type of encoding is called ASCII (American Standard Code for Information Interchange), which is pronounced "ASKEE." In this module, we will understand how computers use mathematical formulas to create the huge numbers needed for processing information.

Finally, in this module, we will gain an understanding of the difference between a hard disk drive and RAM (Random Access Memory) and how they work together. In other words, we will see computer operation at the most fundamental level.

NOTE: Students do not need to be experts at anything in this module. You don't have to be able to convert decimal to binary to hexadecimal in your head. But you do need to have a basic understanding of the concepts. Many topics coming up in this course require this knowledge. Of course, if you are going to work in and around the IT field, you also need to have this understanding.

Computers at the Most Basic Level

- In reality, a computer can do exactly one thing: **It can add**
  - That is literally the only thing a computer can do
- It looks at a string of ones and zeros:
  - The placement of the ones represent particular values
  - The computer adds those values
- Depending on the point in the software, the value has meaning:
  - The value may mean to put a character on the screen
  - Or it may mean to print a document
  - Or it could mean any one of countless other things

Computers at the Most Basic Level

When you really boil a computer's functionality down to the most basic possible level, you find that a computer can do precisely one thing and one thing only. A computer can add.

Inside the computer, it sees a string of ones and zeros. Depending on the exact placement of the ones, they have a particular value (the system ignores zeros as simple placeholders). The computer adds the values represented by the location of the ones to come up with a specific value.

Depending on the point in the software and what the logic of the software currently says, that value has meaning. In one place in the software, it may mean to place a particular character on the screen. For example, perhaps the value the computer adds up to is 65. The software may tell the computer, "if the value equals 65, put an uppercase A on the screen, but if the value equals 66, put an uppercase B on the screen" and so on.

Elsewhere in the same software, it may mean to print a document. Each value could have countless other meanings as well. It all goes back to what the logic in the software currently says and the value the computer arrives at.

Remember – A computer can only do one thing…
IT CAN ADD!

Why Numbering Systems?

- Computers see only binary 1s and 0s:
  - A computer sees the letter *A* as: 01000001
  - A computer sees the letter *a* as: 01100001
    - Notice: That is **eight** 1s and 0s (more on this shortly)
- Humans don't handle strings of 1s and 0s well
- We need a relatively simple way to work with these values:
  - We need to turn binary numbers into decimal numbers
  - And let's not forget our good friend hexadecimal! ☺

Why Numbering Systems?

To gain an understanding of computers, one of the things you need to get is that computers see everything, absolutely everything, as strings of 1s and 0s. They perform a mathematical process on these 1s and 0s to reach a particular value. That value has some meaning depending on the software the computer is currently running. For example, it may mean to print a file, to put a character on the screen, or any number of other possibilities.

Those strings of 1s and 0s are called binary, or more specifically, the binary numbering system. Computers use binary because early computers used electrical contacts that were either closed (a 1) or open (a 0). Those contacts were arranged into groups of eight, meaning they represented eight 1s or 0s. Those eight digits became known as a byte.

The problem is that you and I do not handle strings of 1s and 0s well. We are used to thinking in terms of the decimal numbering system that we learned in school. 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10, 100, 1,000, and so on.

Therefore, we needed a way to convert a decimal number into binary and back to decimal again. But also, because we don't deal with those eight digits well, we needed a way to express a byte of data using a shorthand of two digits. Hence, something called hexadecimal was born. (Note that hexadecimal is often referred to as simply *hex*.)

Anatomy of a Byte (1)

- A single 1 or 0 is called a **bit** 1
- Eight 1s and 0s are called a **byte** 01000001
- Four bits are called a **nibble** 0100
- Each 1 or 0 has a place value twice its neighbor:

High-order bit **Low-order bit**

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	1

- If there is a 1, the place value is counted:
  - So here the value of this byte is 64 + 1 = 65

Anatomy of a Byte (1)
Again, computers see everything as strings of ones and zeros called bits. Bits are logically grouped into 8 bits to comprise one byte. A byte of 8 bits is what computers most often work with when processing data. For example, one 8-bit byte comprises one letter of the alphabet. Occasionally, computers work with one-half a byte, which we call a nibble (more on nibbles on the next slide).

Each of the 8 bits of a byte has a place value. The smallest value (called the *low-order bit*) is 1. Each place value is two times its neighbor, so the place values are 1, 2, 4, 8, 16, 32, 64, and 128. The highest place value (128) is called the *high-order bit*.

When we need to determine the value of a byte, we add the place value for any position that has a 1 present. We do not add the place value if there is a zero in that position. Therefore, if you take the byte above, 01000001, and add the place values, you can see that the 64-place value has a 1, and the 1 place value has a 1. Because 64 plus 1 equals 65, this byte has a value of 65.

Representing the entire byte in a mathematical formula would look like this:
0 + 64 + 0 + 0 + 0 + 0 + 0 + 1 = 65
The byte 10101010 would look like this:
128 + 0 + 32 + 0 + 8 + 0 + 2 + 0 = 170

A computer adds these place values precisely this same way to arrive at a value for that particular byte. The byte's value has a meaning to the software the computer is running.

Indeed, the same byte value might have a different meaning in different parts of the same software. For example, in a single piece of software, the byte value of 65 might cause an uppercase *A* to display on the screen in one place but may indicate to print a document in another place in the same software. For that matter, the byte value 65 might have a hundred or more meanings at different points of a single software program.

Anatomy of a Byte (2)

Values of a byte…

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	= 0
1	1	1	1	1	1	1	1	= 255
0	1	0	0	0	0	0	1	= 65
0	1	1	0	0	0	0	1	= 97

- Bytes can equal 0 though 255
- 0 to 255 is 256 total possible values

- There is only one combination of 1's and 0's to equal any of the 256 possible values
  - e.g. only 01000001 will equal 65 (displaying an "A")
  - e.g. only 01100001 will equal 97 (displaying an "a")

  - There is never ambiguity in byte values

Anatomy of a Byte (2)

It is good to understand possible byte values. If you have a byte where every bit is zero, in other words, eight 0's, then:

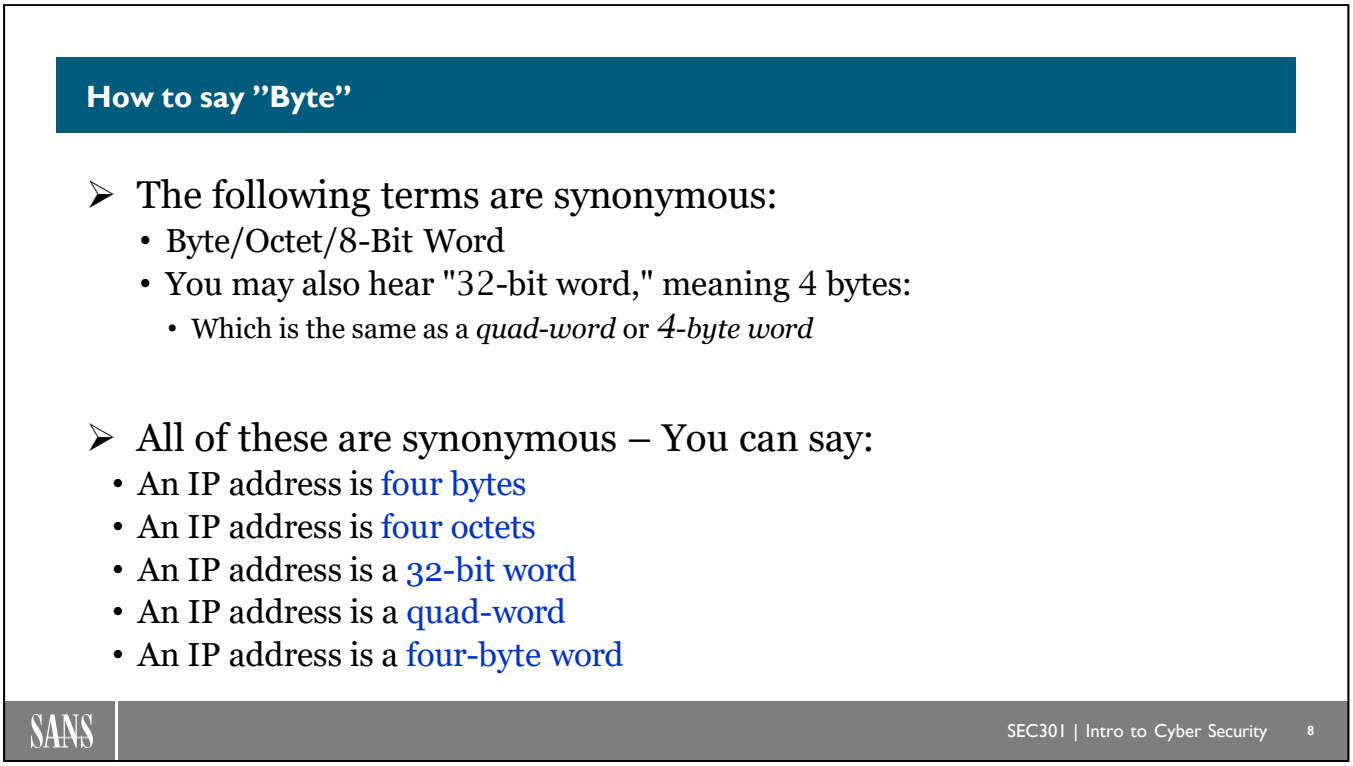
0+0+0+0+0+0+0+0=0

By contrast, if you have a byte of eight 1's, then:

128+64+32+16+8+4+2+1=255

Therefore, the possible values that can "fit" in a single byte are 0 through 255. **NOTE** that 0 through 255 is a total of 256 possible values. This fact will be important shortly.

It is also essential to realize that there is precisely one, and only one combination of 1's and 0's that will equal any of the possible byte values. For example, only 01000001 will equal 65 and display an upper case "A" if that is what the software says to do with that value. Likewise, only 01100001 will equal 97 and show a lower case "a" if that is what the software currently says to do with that value.



Anatomy of a Byte (3)

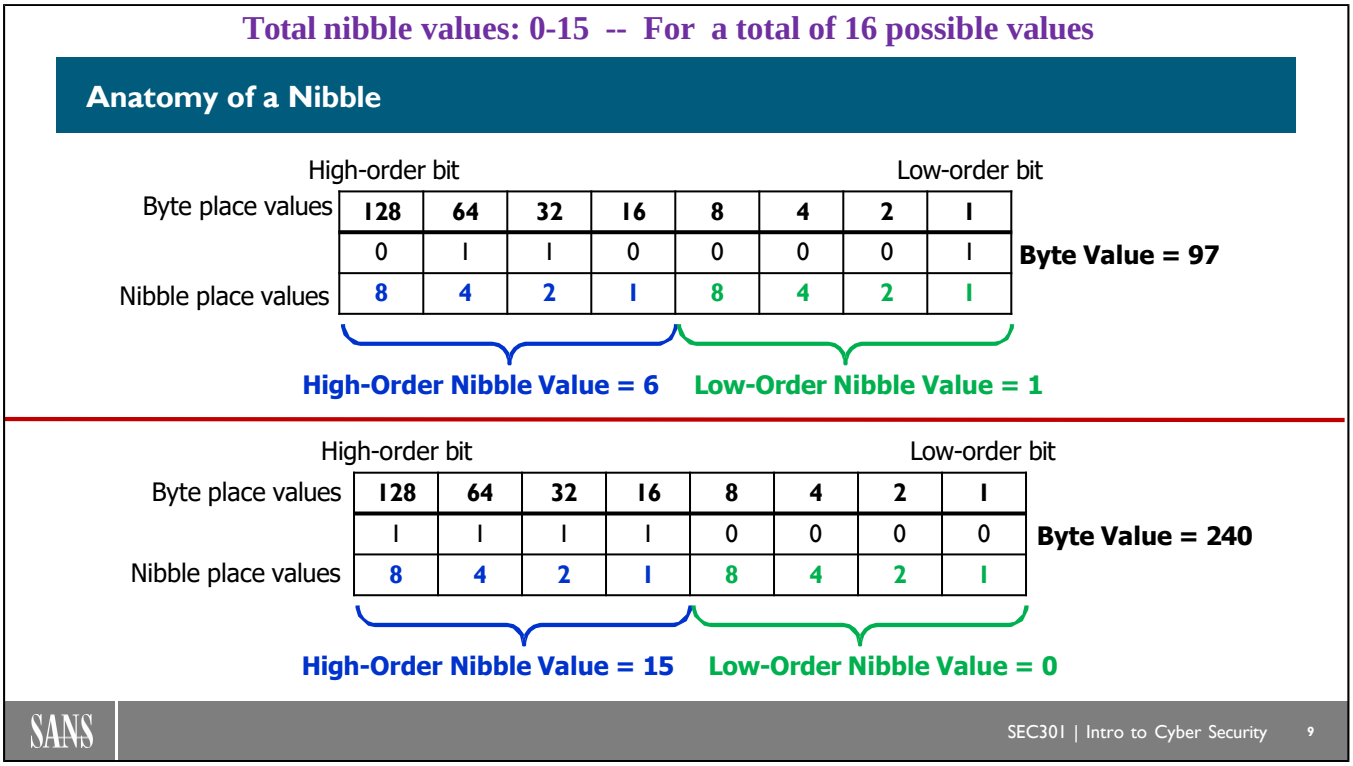
For some reason, the IT world is rife with synonymous terms. We run into this a great deal in this industry. One example is that the terms byte, octet, and 8-bit word all mean exactly the same thing. You will hear IT professionals using these terms interchangeably.

By the way, you may also hear an IT person use the terms 32-bit word, quad-word, and 4-byte word interchangeably as well. Those are also synonymous terms that are all ways of talking about 4 bytes. Sorry …

Occasionally, we do have a need to work with nibbles, which you will see in an example shortly. When we work with nibbles, we change how we handle place values slightly. We simply break a byte into two halves. The place values of the lower half (called the low-order nibble) are 1, 2, 4, and 8, and the place values of the upper half (called the high-order nibble) are also 1, 2, 4, and 8. Note that the true place values for the byte (1 through 128) are still there. This is a function we do to make calculations easier.

So, the byte 01000001, as you see here, now has a high-order value of 4 and a low-order value of 1. So, you might communicate this to someone by saying, "the byte has nibble values of 4-1."

Notice that the entire byte still has a full value of 65, and that is how the computer processes it. But occasionally you and I have an easier time by breaking things into nibbles. (Also, as we will see later in the networking section, some fields are only 4 bits in length.)



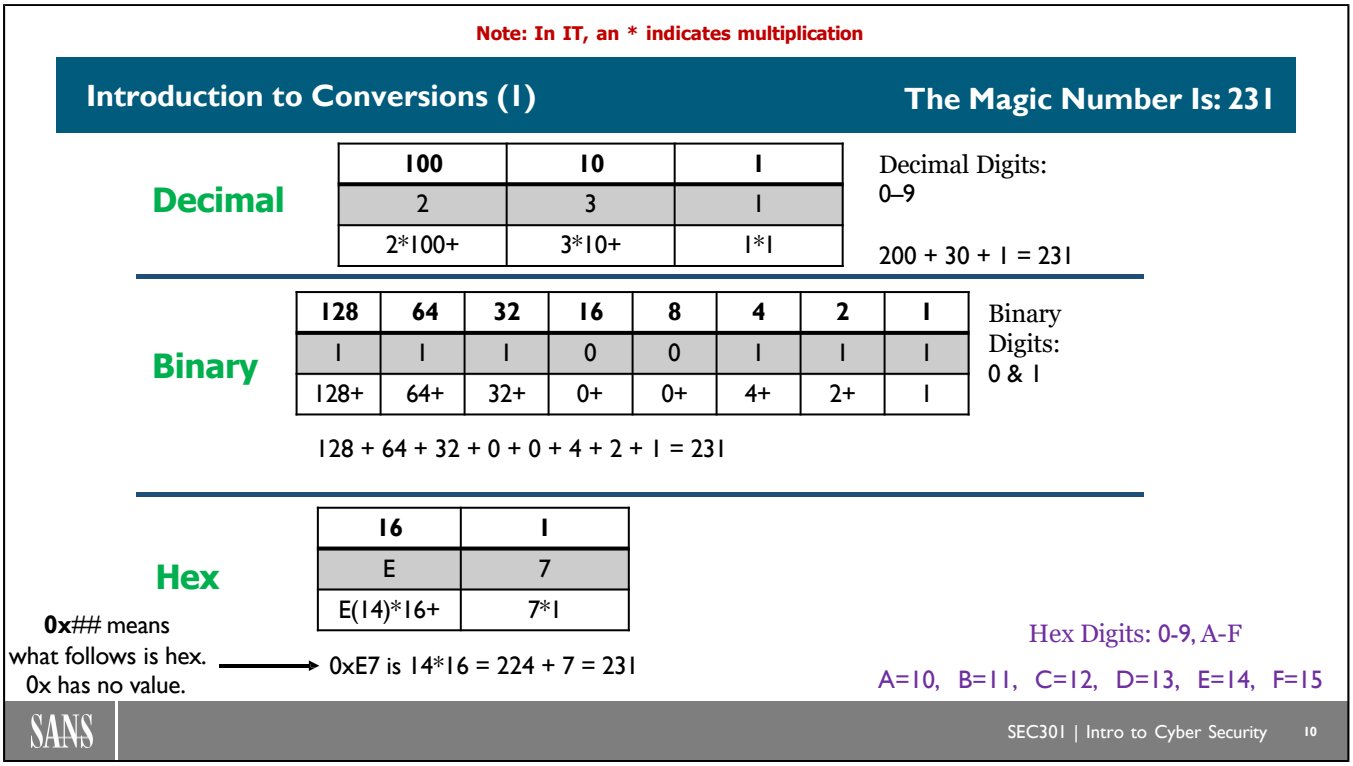
Anatomy of a Nibble

Sometimes, we work with nibbles. At this point in the explanation, why we might work with nibbles may not be clear to you. Take our word for it, in the slides coming up, nibbles will become your friend.

Above, you see two examples. The first has a binary value of 01100001 for a byte value of 97. The leftmost nibble is the high-order nibble. Those four bits are 0110. If we use nibble values for the high-order nibble, as you see above, those would be 1, 2, 4, 8. Adding the nibble place values together would mean 0+4+2+0 = 6. The low-order nibble (the rightmost four bits) is 0001. With the nibble values shown, we would add 0+0+0+1 = 1.

Why would we do this? For easier calculation as you will soon see. Without a calculator, is it easier to add 4+2, or is it easier to add the byte values of 64+32?

When working with nibbles, note the values that can fit in one nibble of 4 bits. If you have four 1's and add all of the nibble values, it would be 8+4+2+1 = 15. So as you can see in the slide above, a byte of 11110000 has a byte value of 240 because:
128+64+32+16+0+0+0+0 = 240. It has a high-order nibble value of 15 and a low-order nibble value of 0. The purpose of showing this particular byte is to illustrate that the possible nibble values are 0-15 which results in a total of 16 possible values.



Introduction to Conversions

The Magic Number Is: 231

When we work with any numbering system, there are two things we have to know. First, we must know the digits we can use, and second, we have to know the place values.

Let's begin with the decimal number 231. Decimal is a base-ten numbering system, which means that there have to be exactly 10 digits. The decimal digits are 0 through 9. Also, as a base-ten numbering system, each place value is 10 times its neighbor. So, we have place values of 1, 10, 100, and so on. Therefore, if we take 2 hundreds, plus 3 tens, plus 1, it equals 231. We realize this is a review, but the concepts of digits and place values have to be firmly fixed in your mind before we move on.

Binary is a base-two numbering system, so we need two digits. Those digits are 1 and 0. With a base-two system, each place value is two times its neighbor. So, if we need to represent the value of 231, we would need to "turn on" the ones in certain place values. It happens that 128 + 64 + 32 + 0 + 0 + 4 + 2 + 1 equals 231. We do not add the place values of 16 and 8 because those positions have a 0 instead of a 1.

Now to hexadecimal. Hex is a base-sixteen numbering system. This means that each place value is 16 times its neighbor, so we have place values of 1, 16, and so on. It also means that we need a total of 16 digits, but each digit can be only one character in length. So, we will use 0 through 9 to represent those values. But to represent 10 through 15, we will use A through F (or a through f if you prefer). In other words, the digit A represents 10, B is 11, C is 12, D is 13, E is 14, and F is 15. This gives us our total of 16 digits: 0 through F.

To arrive at a value of 231, we would place an E (representing 14) in the high-order position (place value of 16) and 7 in the low-order position (place value 1). Because E times 16 equals 224, plus 7 equals 231, we see that the hex number 0xE7 is equivalent to 231 and also 11100111.

So, the actual formula is:

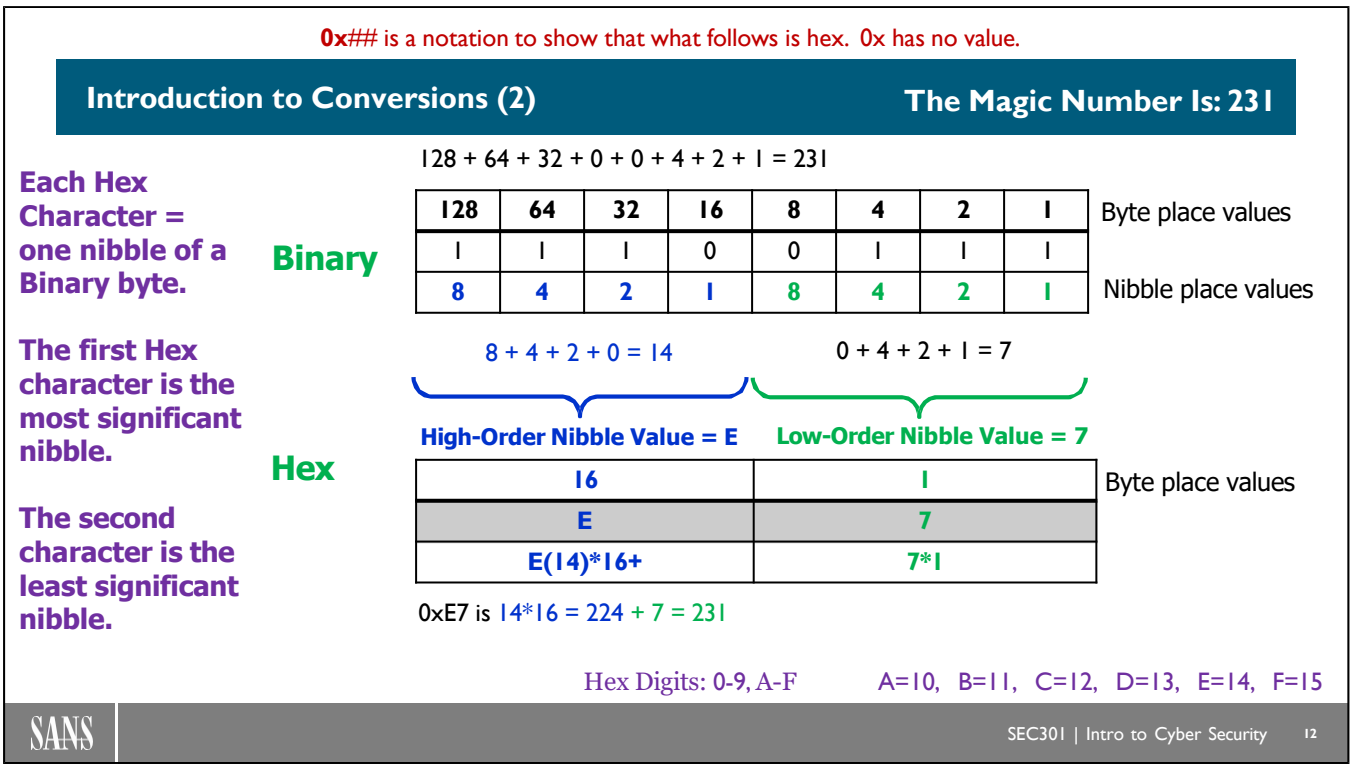
(E times 16) plus (7 times 1) = 231

Or

(14 times 16) plus (7 times 1) = 231

Or to use the proper IT notation where an * represents multiplication, it would look like:

(E * 16) + (7 * 1) = 231



Introduction to Conversions (2)

The first time you see hexadecimal, it is often very confusing. In our example here, where in the world did we get "E7"? It is quite simple, once you understand it.

If you recall earlier, we explained that while every byte has byte values of 1, 2, 4, 8, 16, 32, 64, and 128 - they also have nibble values. The nibble values are 1, 2 4, 8 for the low-order nibble, and 1, 2, 4, 8 for the high-order nibble.

If you look closely at our example of E7, you will notice that each digit represents one nibble of the binary byte. We can find the value of each hexadecimal digit by adding the nibble values.

E represents the numeric value of 14. In the high-order nibble, the 1's are on in the 8, 4, and 2 place values. Therefore, 8+4+2+0 = 14 or E.

7 represents the numeric value of 7. In the low-order nibble, the 1's are on in the 4, 2, and 1 positions. Therefore, 0+4+2+1 = 7.

Scenario Introduction

- Because we understand these numbering systems so well now, let's have a test!
  - We will learn to convert these numbers easily
  - You have been hired in the world of fast food!
  - You are working the cash drawer:
    - **Dec-A-Bux:** Denominations of $1, $10, and $100 bills only
    - **Bin-A-Bux:** Denominations of $1, $2, $4, $8, $16, $32, $64, and $128 bills only
    - **Hex-A-Bux:** Denominations of $1 and $16 bills only
  - You must *always* give the smallest number of bills possible

Scenario Introduction
It is time to find out if we fully understand these numbering systems and how we can translate numbers between them. Let's use a simple scenario to accomplish this.

You have just been hired in the innovative world of fast food. You are working the cash drawer, accepting money from customers and making change as necessary.

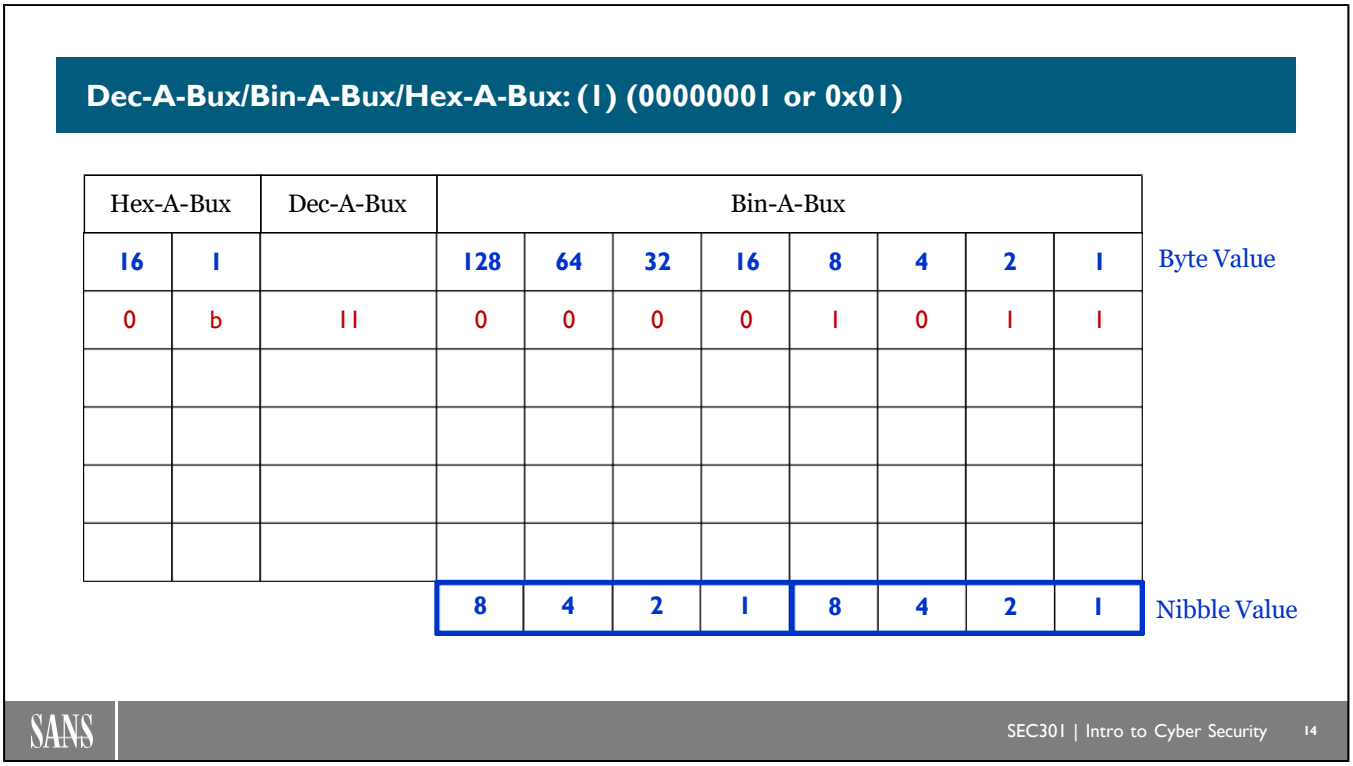
Unfortunately, you have a manager who is continuously swapping out your cash drawer with three kinds of money.

You have:
- Dec-A-Bux with denominations of $1, $10, and $100 bills only
- Bin-A-Bux with denominations of $1, $2, $4, $8, $16, $32, $64, and $128 bills only
- Hex-A-Bux with denominations of only $1 bills and $16 bills

- In all cases, you must *always* give the smallest number of bills possible as change.

Now, let's find out how this might work.

(Note: Dec-A-Bux, Bin-A-Bux, and Hex-A-Bux are used here with the written permission of George Mays.)



Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (1) (00000001 or 0x01)
To begin, you have Bin-A-Bux in your cash drawer. You owe a customer $11 in change. Using the smallest number of bills possible, which Bin-A-Bux bills will you give the customer?

Would you give them a $128 bill? Of course, not (so we have a zero in that position). Likewise, you would not give the customer any $64, $32, or $16 bills (so those positions are also zeros). But you would give the customer an $8 bill (so we will put a 1 in that position). At which point, you still owe the customer three more dollars since $11 minus $8 is $3. You would give zero $4 bills, but one $2 and one $1 bill.

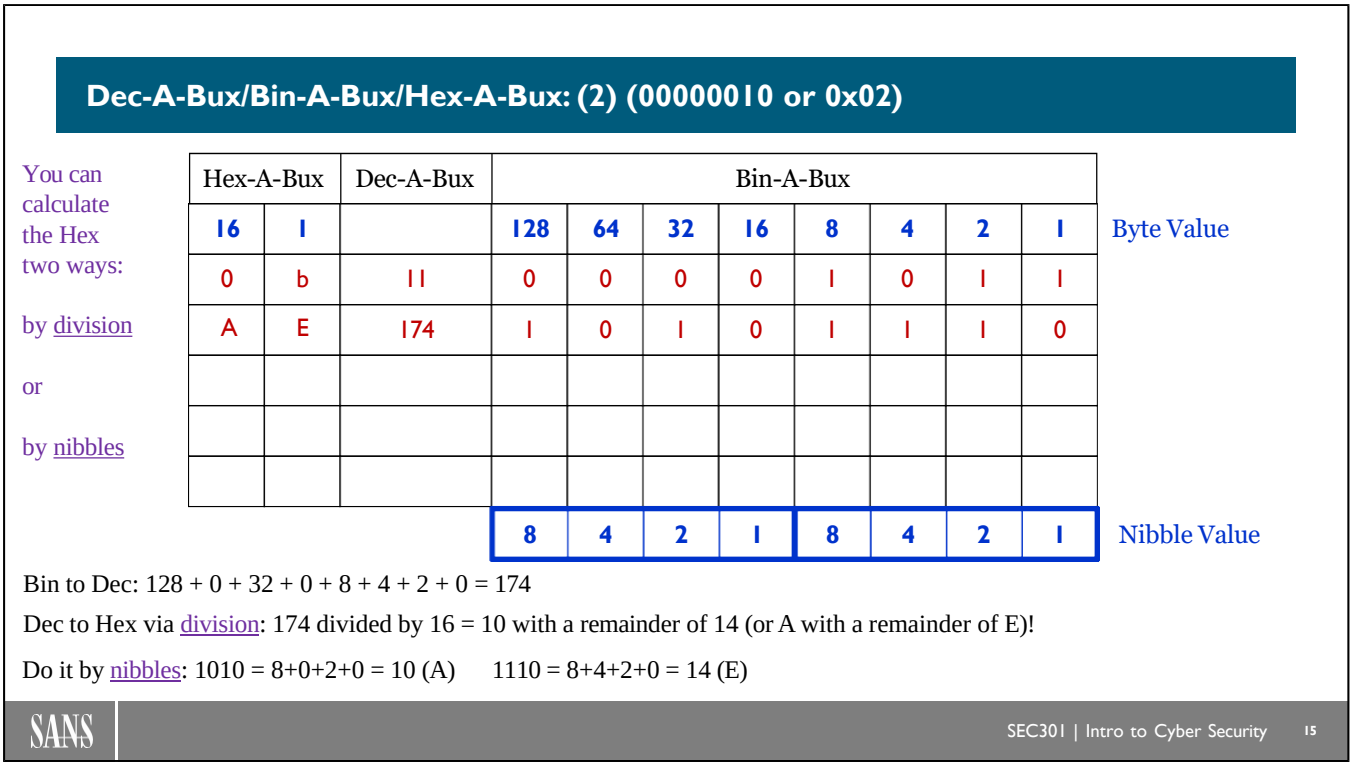
And so, we can see that in Bin-A-Bux (or binary) 11 is 00001011 (0 + 0 + 0 + 0 + 8 + 0 + 2 + 1 = 11)

But what if your cash drawer contains Hex-A-Bux? How many bills would you give a customer if you owed them $11?

Well, you clearly would not give them any $16-dollar bills, so we can put a zero in that position. But how many $1 bills would you give them?

If you answer 11, I'm afraid that is incorrect. There is no such thing as 11 in hex … but there is B, which is equivalent to 11. So, you would give them a B dollar bill.

Therefore, 11 decimal is equal to the hex number 0B – written 0x0B.



Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (2) (00000010 or 0x02)
Now a customer has come in and handed you a stack of Bin-A-Bux. Specifically, they gave you a $128, a $32, an $8, a $4, and a $2 bill. How much is that? Simple addition tells us that 128 + 32 + 8 + 4 + 2 = 174.

Now, how much did they give us in Hex-A-Bux? There is more than one way to calculate this.

First, let's find the answer using division. If you were to divide the decimal value of 174 by our high-order place value of 16, you would get an answer of 10 with a remainder of 14. You would put the divisible value of 10 in the high-order nibble and the remainder in the low-order nibble. In other words, this would make the high-order hex digit A (10) and the low-order hex digit E (14).

Of course, you can also calculate the binary to hexadecimal using nibbles. Because each position of a hexadecimal number is one-half of a byte (or a nibble), we could calculate the binary into hex by using nibble place values instead, as you see on this slide. With both the low-order and high-order nibbles using place values of 1, 2, 4, and 8, it becomes much simpler.

So, the high-order nibble is: 1010 is 8 + 0 + 2 + 0 = A
And the low-order nibble is: 1110 is 8 + 4 + 2 + 0 = E

So, the number 0xAE (or 0xae) is equivalent to 10101110, which is also equivalent to 174.

Who knew this could be so simple!

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (3) (00000011 or 0x03)

Hex-A-Bux		Dec-A-Bux	Bin-A-Bux								
16	**1**		**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Byte Value
0	b	11	0	0	0	0	1	0	1	1	
A	E	174	1	0	1	0	1	1	1	0	
1	7	23	0	0	0	1	0	1	1	1	
Hex to Dec: 1*16 = 16 + 7 = 23			**8**	**4**	**2**	**1**	**8**	**4**	**2**	**1**	Nibble Value

Hex to Bin: 2 nibbles of 4 bits: number 1 in the first 4 bits and number 7 in the second 4 bits
0001 = the first nibble 0111 = the second nibble Together = 00010111

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (4) (00000100 or 0x04)
In this case, a customer handed you 0x17 in Hex-A-Bux. How much is this in both Dec-A-Bux and Bin-A-Bux (or decimal and binary as it were)?

First, let's convert to Dec-A-Bux. We take the 1 in the high-order position times the place value of 16. Meaning that 1 * 16 = 16. We then add the 7 from the low-order nibble and come to 23 (16 + 7 = 23).

Now, let's convert it to Bin-A-Bux. It is simple to do if we do it by the nibble.

The value 1 in a nibble is 0001

The value 7 in a nibble is 0111

When we combine the two nibbles, they become the byte 00010111.

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (4) (00000100 or 0x04)

Hex-A-Bux		Dec-A-Bux	Bin-A-Bux								
16	**1**		**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Byte Value
0	b	11	0	0	0	0	1	0	1	1	
A	E	174	1	0	1	0	1	1	1	0	
1	7	23	0	0	0	1	0	1	1	1	
0	F	15	0	0	0	0	1	1	1	1	
			8	**4**	**2**	**1**	**8**	**4**	**2**	**1**	Nibble Value

Hex to Dec:
0*16 = 0 + F = 15

Bin to Dec:
0+0+0+0+8+4+2+1 = 15

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (4) (00000100 or 0x04)
On this slide, we have the Hex value of 0x0F. With this byte, we take the first nibble of 0 times 16 for a value of 0. The low-order nibble has a value of F, or 15. Hence, 0+15=15 in decimal.

The binary is also very straightforward. A high-order nibble of 0 = 0000 in binary. A low order nibble of F = 1111 in binary. Hence, the binary of 0x0f is 00001111.

Note, the first sentence on this page is a good example of why having hex around can be nice. It was much simpler to type “0x0F” and get the value correct as opposed to typing “00001111”.

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (5) (00000101 or 0x05)

Hex-A-Bux		Dec-A-Bux	Bin-A-Bux								
16	**1**		**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Byte Value
0	b	11	0	0	0	0	1	0	1	1	
A	E	174	1	0	1	0	1	1	1	0	
1	7	23	0	0	0	1	0	1	1	1	
0	F	15	0	0	0	0	1	1	1	1	
f	0	240	1	1	1	1	0	0	0	0	
			8	**4**	**2**	**1**	**8**	**4**	**2**	**1**	Nibble Value

Hex to Dec:
F*16 = 240 + 0 = 240

Bin to Dec:
128+64+32+16+0+0+0+0 = 240

Dec-A-Bux/Bin-A-Bux/Hex-A-Bux: (5) (00000101 or 0x05)
On this slide, we have a Hex value of 0xF0 to complete the chart. To reach the decimal value, take F which represents the number 15 times 16. This will equal 240. The low order nibble of the byte is 0, so add 240+0=240 in decimal.

The binary is very straightforward since a hex digit of f = four 1's and a hex digit of 0 = four 0's. Therefore, the binary of 0xF0 = 11110000 (and please remember that with hexadecimal, upper case F or lower case f makes no difference).

We hope these conversions and how the three numbering systems relate to each other is clear by this point. As stated earlier, for this class, you do not have to be fast at these conversions, and you certainly do not need to do them in your head.

We do need you to recognize a binary byte and nibble. We need you to recognize a hexadecimal number when you see it. We also need you to realize that binary and hexadecimal numbers are just another way of representing values. In cryptography, we will mention key lengths expressed in a specific number of bits. For example, we will tell you that a particular cryptographic system uses "56-bit key" while another uses "128-bit key". We need you to recognize these statements as meaning that those systems use 56 1's and 0's in a string or 128 1's and 0's in a series to create their cryptographic key.

Note: ASCII is just one type of encoding used by computers.

ASCII Basics

- An earlier slide said:
  - The letter *A* looks like 01000001 to computers
  - That is because computers encode using ASCII:
    - American Standard Code for Information Interchange
- An ASCII Table contains all the letters of the alphabet and their associated value:
  - There are other characters in the ASCII table that are "non-printable":
    - That is, carriage returns and such, every key on a keyboard and more

An ASCII chart is in your book.

Character	char	Decimal	Binary	Hex
Capital A	A	65	01000001	41
Capital B	B	66	01000010	42
Capital C	C	67	01000011	43

SANS SEC301 | Intro to Cyber Security 19

ASCII Basics

As stated earlier, computers see everything as a string of 1s and 0s. We now know that 8 of those 1s and 0s make up a byte and that the place values are added together to reach a value. Earlier, you saw "the letter *A* looks like 01000001 to computers." This is a method of encoding characters, in other words, determining what value equals what printable character. There are several encoding schemes that can be used in computers, but by far the most common is called ASCII (pronounced "ASKEE"), the American Standard Code for Information Interchange.

If we take the byte 01000001 and add the place values, it would look like this:

0 + 64 + 0 + 0 + 0 + 0 + 0 + 1 = 65

In ASCII, the decimal number 65 (or the binary number 01000001 or the hex number 0x41) represents the character A. For example, if the software running on the computer wants to put a character on the computer screen, it adds the place values of a single byte to reach a value. If the byte is 01000001, then that value is 65, and the software knows to place an uppercase A on the screen. By contrast, if the byte were 01100001, the place values would add up to 97 decimal (0x61) and the software would know to put a lowercase *a* on the screen.

The following pages show an ASCII table. There are more ASCII characters than what you see represented here, including extended ASCII codes for Latin characters, etc. The complete

ASCII chart is available at http://www.ascii-code.com/ We should also point out that ASCII is just one common method of encoding characters on a computer. There are others used. For example, on the Web, you often find UTF-8, which is the same concept with a different chart.

In both cryptography and networking, a basic understanding of ASCII is necessary.

ASCII Table Page 1

Character	char	Decimal	Binary	Hex
Null Character	NUL	0	00000000	00
Start of heading	SOH	1	00000001	01
Start of text	STX	2	00000010	02
End of text	ETX	3	00000011	03
End of Transmission	EOT	4	00000100	04
Enquiry	ENQ	5	00000101	05
Acknowledgement	ACK	6	00000110	06
Bell	BEL	7	00000111	07
Back Space	BS	8	00001000	08
Horizontal Tab	HT	9	00001001	09
Line Feed	LF	10	00001010	0A
Vertical Tab	VT	11	00001011	0B
Form Feed	FF	12	00001100	0C
Carriage Return	CR	13	00001101	0D
Shift Out	SO	14	00001110	0E
Shift In	SI	15	00001111	0F
Data Line Escape	DLE	16	00010000	10
Device Control 1	DC1	17	00010001	11
Device Control 2	DC2	18	00010010	12
Device Control 3	DC3	19	00010011	13
Device Control 4	DC4	20	00010100	14
Negative Ack	NAK	21	00010101	15
Synchronous Idle	SYN	22	00010110	16
End Transmit Block	ETB	23	00010111	17
Cancel	CAN	24	00011000	18
End of Medium	EM	25	00011001	19
Substitute	SUB	26	00011010	1A
Escape	ESC	27	00011011	1B
File Separator	FS	28	00011100	1C
Group Separator	GS	29	00011101	1D
Record Separator	RS	30	00011110	1E
Unit Separator	US	31	00011111	1F
Space	Space	32	00100000	20

ASCII Table Page 2

Character	char	Decimal	Binary	Hex
Exclamation	!	33	00100001	21
Double Quote	"	34	00100010	22
Pound/Hash	#	35	00100011	23
Dollar Sign	$	36	00100100	24
Percent	%	37	00100101	25
Ampersand	&	38	00100110	26
Single Quote	'	39	00100111	27
Left Paren	(	40	00101000	28
Right Paren	)	41	00101001	29
Asterisk	*	42	00101010	2A
Plus Sign	+	43	00101011	2B
Comma	,	44	00101100	2C
Hyphen	-	45	00101101	2D
Period	.	46	00101110	2E
Forward Slash	/	47	00101111	2F
Zero	0	48	00110000	30
One	1	49	00110001	31
Two	2	50	00110010	32
Three	3	51	00110011	33
Four	4	52	00110100	34
Five	5	53	00110101	35
Six	6	54	00110110	36
Seven	7	55	00110111	37
Eight	8	56	00111000	38
Nine	9	57	00111001	39
Colon	:	58	00111010	3A
Semicolon	;	59	00111011	3B
Less-Than	<	60	00111100	3C
Equals sign	=	61	00111101	3D
Greater-Than	>	62	00111110	3E
Question Mark	?	63	00111111	3F
At Sign	@	64	01000000	40

ASCII Table Page 3

Character	char	Decimal	Binary	Hex
Capital A	A	65	01000001	41
Capital B	B	66	01000010	42
Capital C	C	67	01000011	43
Capital D	D	68	01000100	44
Capital E	E	69	01000101	45
Capital F	F	70	01000110	46
Capital G	G	71	01000111	47
Capital H	H	72	01001000	48
Capital I	I	73	01001001	49
Capital J	J	74	01001010	4A
Capital K	K	75	01001011	4B
Capital L	L	76	01001100	4C
Capital M	M	77	01001101	4D
Capital N	N	78	01001110	4E
Capital O	O	79	01001111	4F
Capital P	P	80	01010000	50
Capital Q	Q	81	01010001	51
Capital R	R	82	01010010	52
Capital S	S	83	01010011	53
Capital T	T	84	01010100	54
Capital U	U	85	01010101	55
Capital V	V	86	01010110	56
Capital W	W	87	01010111	57
Capital X	X	88	01011000	58
Capital Y	Y	89	01011001	59
Capital Z	Z	90	01011010	5A
Left Bracket	[	91	01011011	5B
Backslash	\	92	01011100	5C
Right Bracket	]	93	01011101	5D
Caret	^	94	01011110	5E
Underscore	_	95	01011111	5F
Back quote	`	96	01100000	60

ASCII Table Page 4

Character	char	Decimal	Binary	Hex
Lower A	a	97	01100001	61
Lower B	b	98	01100010	62
Lower C	c	99	01100011	63
Lower D	d	100	01100100	64
Lower E	e	101	01100101	65
Lower F	f	102	01100110	66
Lower G	g	103	01100111	67
Lower H	h	104	01101000	68
Lower I	i	105	01101001	69
Lower J	j	106	01101010	6A
Lower K	k	107	01101011	6B
Lower L	l	108	01101100	6C
Lower M	m	109	01101101	6D
Lower N	n	110	01101110	6E
Lower O	o	111	01101111	6F
Lower P	p	112	01110000	70
Lower Q	q	113	01110001	71
Lower R	r	114	01110010	72
Lower S	s	115	01110011	73
Lower T	t	116	01110100	74
Lower U	u	117	01110101	75
Lower V	v	118	01110110	76
Lower W	w	119	01110111	77
Lower X	x	120	01111000	78
Lower Y	y	121	01111001	79
Lower Z	z	122	01111010	7A
Left Brace	{	123	01111011	7B
Vertical Pipe	|	124	01111100	7C
Right Brace	}	125	01111101	7D
Tilde	~	126	01111110	7E

Computer Math Primer (1): A 2-Byte Field

- We know how place values equal byte values:
  - But the largest byte value is 255
  - What if we need to represent a bigger number?
- Example: A 2-byte number can equal 65,535:
  - The first byte * 256 and then add the second byte
  - Example: 2 bytes of 1's: 11111111 11111111
  - Eight 1's in a byte = byte value 255. So:
  - 255 * 256 = 65,280 + 255 = 65,535
  - 11111111 * 256 + 11111111 = 65,535

Remember: Byte values range from 0 to 255, or **256** total possible byte values.

Computer Math Primer (1): A 2-Byte Field

We now have a solid understanding of bytes, place values, byte values, and such. But this raises a question. Because the largest value that can fit in a single byte is 255, what happens when we need to represent a larger number?

That is where some more math comes in.

For example, let's say that we need to represent the number 65,535. That happens to be the largest number that can fit in 2 bytes. We determine this with the following math formula:

The first byte * 256 + the second byte. In other words:
255 * 256 + 255 = 65,535

When we get to the networking section later, you will find that there are 2-byte fields that can have up to 65,536 possible values (0 through 65,535).

Formula: (First byte * 256) + second byte

Computer Math Primer (2): A 2-Byte Field

First Byte	Second Byte	The math
01010101	11001101	(01010101 * 256) + 11001101 = 21,965
64+16+4+1=85	128+64+8+4+1=205	(85 * 256) = 21,760 + 205 = 21,965
85	205	85 * 256 = 21,760 + 205 = 21,965
00000001	11111111	(00000001 * 256) = 256 + 11111111 = 511
1	255	(1 * 256) = 256 + 255 = 511
00000000	01010000	(00000000 * 256) = 0 + 01010000 = 80
0	80	(0 * 256) = 0 + 80 = 80

An * means multiply

Computer Math Primer (2): A 2-Byte Field Continued

To continue the 2-byte field example for greater clarity, let's look at some examples. The formula for calculating the value of a 2-byte field is: (first byte * 256) + second byte = 2-byte value.

So, in our first example:

- The first byte is 01010101, which equals 85 in decimal
- The second byte is 11001101, which equals 205 in decimal
- So, 85 * 256 = 21,760 + 205 = 21,965

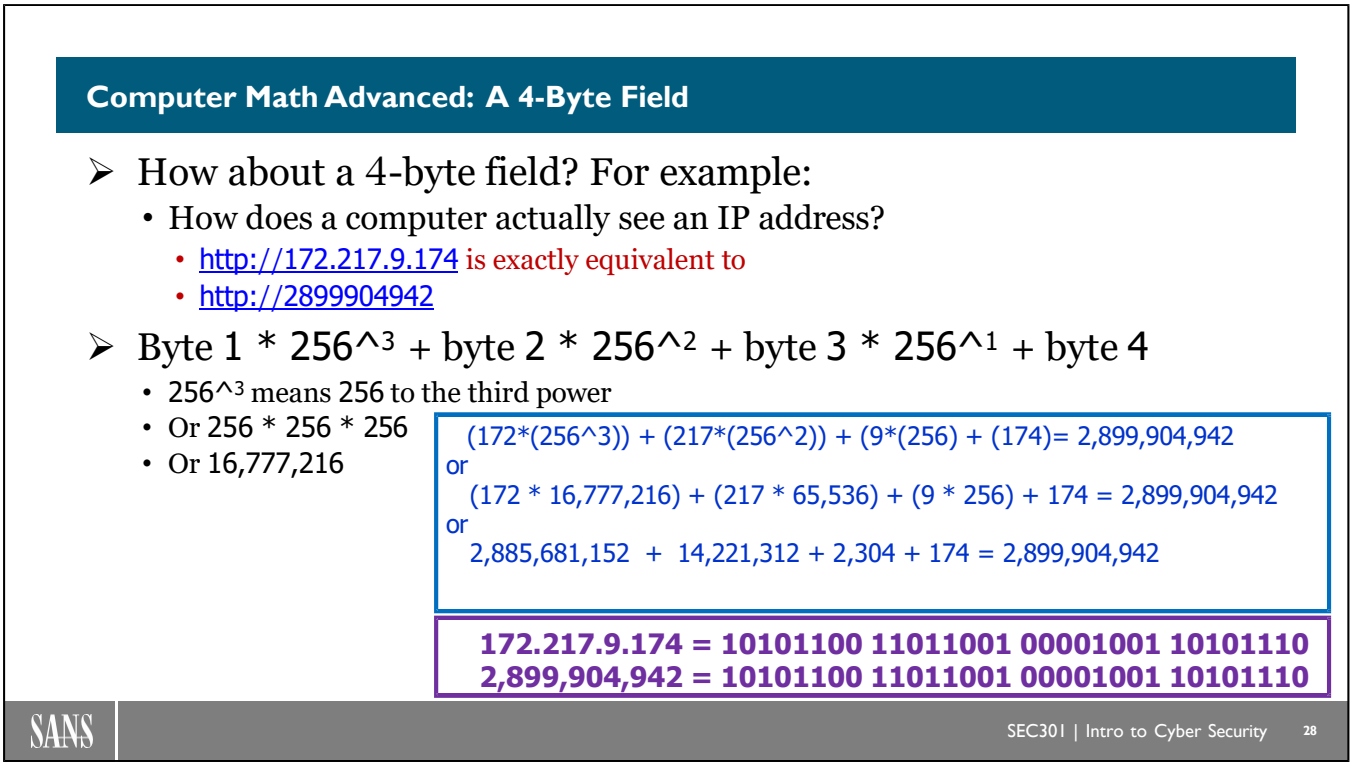
In the second example:

- The first byte is 00000001, which equals 1 in decimal
- The second byte is 11111111, which equals 255 in decimal
- So, 1 * 256 = 256 + 255 = 511

In the third example, we find out how a 2-byte field can equal less than 255:

- The first byte is 00000000, which equals 0 in decimal
- The second byte is 01010000, which equals 80 in decimal
- So, 0 * 256 = 0 + 80 = 80

In networking, there is a 2-byte field called the *Port Number*. Given this formula, you now know that the largest possible port number is 65,535 (see the last page). You also know that a port number of 80 (often used for web traffic) is actually a binary value of 00000000 01010000 and that is exactly the way a computer sees that particular number.



Computer Math Primer (3): A 4-Byte Field

What happens when we need to represent extremely large numbers? For example, what happens if we have a 4-byte field?

The formula for this is:

The first byte * 256 to the third power + the second byte * 256 to the second power + the third byte * 256 to the first power + the fourth byte.

256 to the third power is 256 * 256 * 256 = 16,777,216
256 to the second power is 256 * 256 = 65,536
256 to the first power is 256

So, it would look like:

Byte 1 * 16,777,216 + byte 2 * 65,536 + byte 3 * 256 + byte 4

An example follows on the next page.

(Note: 16,777,216 / 65,536 / and 256 are all multiples of 8.)

An example of a 4-byte field is an IP address. As you will hear repeatedly in the networking section of this course, an IP address is 32 bits or 4 bytes in length.

This example is the IP address 172.217.9.174 (which happens to be the IP address of the google.com web page). That IP address is exactly equivalent to 2,899,904,942. How do we know this? We have to use the formula from the prior page.

Remember:
256 to the third power is 256 * 256 * 256 = 16,777,216
256 to the second power is 256 * 256 = 65,536
256 to the first power is 256

172 * 16,777,216 = 2,885,681,152 (172 * 256 to the third power) plus
217 * 65,536 = 14,221,312 (217 * 256 to the second power) plus
9 * 256 = 2,304 (9 * 256 to the first power) plus
174

Or to say it another way: 2,885,681,152 + 14,221,312 + 2,304 + 174 = 2,899,904,942
Or (172*(256^3)) + (217*(256^2) + (9*256) + 174 = 1,317,631,132

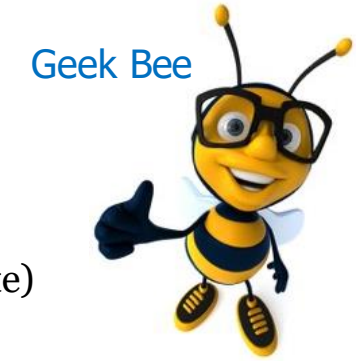
This works because in binary, the dotted decimal notation IP address and the non-dotted decimal IP address are the same (computers ignore the dots):

172.217.9.174 = 10101100 11011001 00001001 10101110
2,899,904,942 = 10101100 11011001 00001001 10101110

Do note: This is "Ninja Level" knowledge. Many IT professionals with years of experience don't realize that this is how computers actually handle IP addresses. You could easily go your entire career in IT and/or Cyber Security and never know this information. However, the question kept coming up, "How does a computer deal with really big numbers larger than 255 or 65,535? How does a computer deal with really big numbers larger than 255 (for one byte) or 65,535 (for a two-byte field)?" The answer is, you keep adding bytes and continue the formula above out to the number of bytes needed. This is the easiest way we could think of to explain and demonstrate how it works.

Beware of Geek B's

- In computing notation
  - B = bytes
  - b = bits



- 1MB = 1 Megabyte
- 1Mb = 1 Megabit (1/8th the size of a Megabyte)

- 1MB = 8Mb
  - 1MB (8,388,608 bits) = 8Mb (8 x 1,048,576 bits = 8,388,608 bits)

Beware of B's

One of the many things experienced computer people think everyone knows is the difference between uppercase and lowercase B's. There really is a significant difference. Unfortunately, the upper/lowercase can be easy enough to miss for the initiated. For the non-initiated, it simply makes no sense whatsoever.

In computing:

- B = byte (8 bits)
- b = bit (1 bit)

So for example, if you look at the specifications of a computer, you might see that it has a 500GB hard disk drive. It is vital that you make note of the capital B, since a 500Gb hard disk drive is one-eighth the size!

All page counts below are based on 1,200 characters per page. Prices valid as of 03/18/2021

Computing Size Measurements (everything is a multiple of 8)

Name	Bits	Bytes	Kilobytes	Megabytes	Gigabyte	Pages of text
Byte (B)	8	1				1 char.
Kilobyte (KB)	8,192	1,024	1			2 to 3 Para.
Megabyte (MB)	8,388,608	1,048,576	1,024	1		873
Gigabyte (GB)	8,589,934,592	1,073,741,824	1,048,576	1,024	1	894,784
Terabyte (TB)	8,796,093,022,208	1,099,511,627,776	1,073,741,824	1,048,576	1,024	916,259,689

A pallet of paper:
- 500 Sheets Per Ream
- 10 Reams Per Carton
- 40 Cartons
- 200,000 pages
- 2,200 U.S. pounds
- 1,000 Kilograms
- $1,859

Terabyte = 916,259,689 pages:
- Divide that by 200,000 pages per pallet
- That's 4,581 pallets
- 10 million U.S. pounds (5,000 U.S. tons)
- 4.5 million Kilograms (4,500 metric tonnes)
- Cost $8.5 Million

Or buy a Corsair 1TB Thumb Drive that costs $374 and weighs a few ounces (about 60 grams).

SANS SEC301 | Intro to Cyber Security 31

Computing Size Measurements

If you go to purchase a computer or hang around the IT department for any amount of time, you will hear terms such as Kilobyte, Megabyte, Gigabyte, Terabyte, and so on. These terms represent data sizes of devices such as hard drives, Random Access Memory (RAM), and other technical gear needed to make computers work.

Above, you see a chart showing the sizes from a single bit through a Terabyte. 1KB is 1,024 bytes, 1MB is 1,024KB, 1GB is 1,024MB, and 1TB is 1,024GB. The number 1024 is a multiple of 256, which is a multiple of 8; therefore, 1024 is also a multiple of 8.

As you look at the chart above, notice how this creates patterns. For example, notice that the number 1,024 appears four times and the number 1,048,576 appears three times and so on. This is not an accident or mistake. These patterns happen because there are 8 bits in a byte and, therefore, every number in the chart above is a multiple of 8 (feel free to divide any of those numbers by 8—you will notice you do not have a remainder).

The numbers above, as well as the numbers for Petabytes (PB), Exabytes (EB), Zettabytes (ZB), and Yottabytes (YB) can be found at this link: https://www.computerhope.com/issues/chspace.htm

So just how big is a Terabyte of data? 916,259,689 pages—assuming 1,200 characters per page, which is a pretty good average. So, if you wanted to put a Terabyte of data on paper

and assume 200,000 sheets of paper are in a pallet, that would take 4,581 pallets of paper. It would cost almost $8.5 million dollars and weigh 5,000 U.S. tons or 4,500 metric tonnes (yes, we are rounding the number of tons/tonnes here).

All page counts below are based on 1,200 characters per page. Prices valid as of 03/18/2019

Computing Size Measurements (everything is a multiple of 8)

Name	Bits	Bytes	Kilobytes	Megabytes	Gigabyte	Pages of text
Byte (B)	8	1				1 char.
Kilobyte (KB)	8,192	1,024	1			2 to 3 Para.
Megabyte (MB)	8,388,608	1,048,576	1,024	1		873
Gigabyte (GB)	8,589,934,592	1,073,741,824	1,048,576	1,024	1	894,784
Terabyte (TB)	8,796,093,022,208	1,099,511,627,776	1,073,741,824	1,048,576	1,024	916,259,689

A pallet of paper:

- 500 Sheets Per Ream
- 10 Reams Per Carton
- 40 Cartons
- 200,000 pages
- $1,859
- 2,200 U.S. pounds
- 1,000 Kilograms

Terabyte = 916,259,689 pages:

- Divide that by 200,000 pages per pallet
- That's 4,581 pallets
- That would cost $ 8.5 Million
- 10 million U.S. pounds (5,000 U.S. tons)
- 4.5 million Kilograms (4,500 metric tonnes)

Or buy a Corsair 1TB Thumb Drive that costs $374 and weighs a few ounces (about 60 grams).

IoT = Internet of Things – cameras, Amazon's Echo, light bulbs, etc.

What Is an Operating System or an OS?

- The software that makes a computer work
  - The central component of an operating system is the "kernel"
  - The OS and kernel define *everything* about how a computer functio ns
- Common operating system s:
  - Microsoft Windows
  - macOS
  - Linux
  - Chromebook
  - Unix
  - Apple iOS
  - Android

macOS
12% of desktops
1% of servers

Windows
83% of desktops
32% of servers
23% of IoT

Linux
1% of desktops
68% of servers
96% of web servers
72% of IoT
All Android

What Is an Operating System or an OS?

Every computer, whether it be a server, a client computer such as a desktop system or laptop, a tablet, a phone, a wireless access point, a Blu-Ray player, a smart TV, or any other type of device must have an operating system to make it work. That operating system is the low-level software that causes a computer to work the way it works.

The central component of any operating system is called the "kernel". It is the software that dictates much of how the computer does what it does. It is the core—or kernel—of the operating system.

An operating system many of us use is Microsoft Windows. It is very popular for the "client computers" such as desktops and laptops. In fact, about 82% of client computers use Windows. Many companies also use the Window Server operating system to provide services such as file servers, web servers, and so on.

The second most common operating system for desktop and laptop client computers is macOS, from the Apple company, with a 12% install base. While there is a Mac Server operating system, it is not in widespread use—though some companies certainly do use that for their server operating systems.

To say this another way, for the client desktop computer on your desk, or the laptop you carry around, Windows and macOS are, by far, the most common at 94% combined. That being said, they are far from the only operating systems in use today.

There is also the Linux operating system. While some people (mostly Information Technology (IT) people) do use this on their desktop or laptop computers, it is far more common as a server platform. For example, 96.5% of the web servers on the Internet run Linux. In addition, many companies may have Windows and/or Mac on the client computers, but run Linux on all or most of their servers. Linux also powers the 500 most powerful computers in the world.

A huge advantage of Linux is that it is completely free (Red Hat Linux you pay for, but you are paying for support, testing, etc., not for the operating system itself).

In large part, because it is free, Linux is also used almost exclusively on the "Internet of Things" or IoT devices. Your Blu-Ray player, the cameras in your house you use to watch your pets, your car, your baby monitor, and so on almost exclusively use Linux. Between its popularity as an internet server platform, and because there will be an estimated 37 billion IoT devices by 2022, this makes Linux the most common operating system in the world.

The Unix operating system was invented clear back in 1971. It is almost exclusively used as a server operating system and has been steadily dropping in popularity for years.

It should be noted that Linux and Unix are very similar in many ways. In fact, you may hear that "Linux is a Unix-like operating system." This is because the two operating systems use almost all of the same commands and function very similarly. They each have a different OS kernel, but how you use them is extremely similar.

Your phone and your tablet are also computers and, therefore, require an operating system. If you have an iPhone or iPad from Apple, the operating system is called the Apple iOS. If you have an Android-based phone or tablet, the operating system comes from Google and is called Android. In both cases, notice that the operating system on the phone and on the tablet is the same (though that may change in late 2021 when Apple puts out their Tablet iOS). Meaning that from a computing perspective, the only difference between a phone and a tablet is that one is smaller and one is larger; however, they both run the exact same OS.

By the way, the Android OS is actually a form of Linux. It uses a Linux kernel, but it has several added components. Information Technology people love to argue about whether or not Android is really a Linux and will do so for hours on end. From the author's perspective, if it has a Linux kernel, it is a Linux OS. But many will strongly disagree with that statement.4

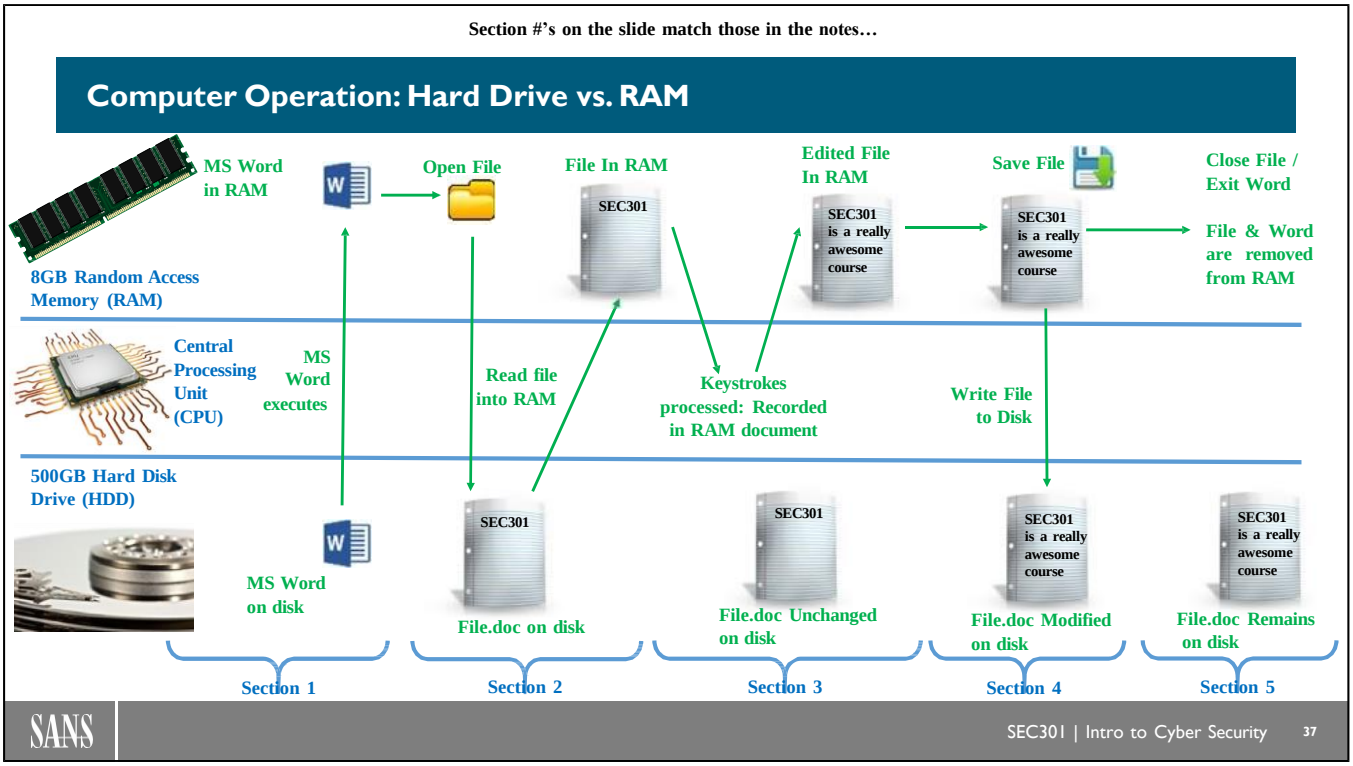
About 72% of devices run IoT devices and about 23% are run by windows. The other five percent of IoT devices are divided among 13 other operating systems.

There are other operating systems out there. This is simply a survey of some of the common OS's you might run into.

Reference for statistics:

https://www.makeuseof.com/tag/linux-market-share/

https://www.itprotoday.com/iot/survey-shows-linux-top-operating-system-internet-things-devices



Computer Operation: Hard Drive vs. RAM

A fundamental element in understanding computer functionality is understanding how information is stored and processed on a computer system. This understanding will be vital later in our discussions of functionality and security.

First, every computer has a *permanent* storage device commonly referred to as a hard disk drive (HDD) or simply hard drive or even just drive. Both software and documents are written to the hard drive and stored there.

Every computer also has a *temporary* storage location referred to as Random Access Memory, or RAM. You CANNOT run software purely off the hard drive—rather, the program is read from the hard drive into RAM and is stored there while the program is running. Likewise, you cannot process (as in edit, print, etc.) a document on the hard drive—it must also be read into RAM and processed there. This looks something like what you see in the diagram above. Each of the Section #'s on the paragraphs below match those on the slide above. The diagram is broken into three areas. The bottom portion is showing what is stored on the hard drive, the top portion is showing what is stored in RAM, and the middle portion is a greatly simplified explanation of the processing the computer is doing as it moves items from one of those areas to the other.

(The section numbers below match those on the slide above)

Section 1: First, we have the Microsoft Word application that has been installed on the tallation, the software was written onto the hard drive of

the computer. When the user executes Microsoft Word, the system reads the software from the hard drive, and a copy of the software is loaded into RAM (it is a copy—the software remains on the hard drive as well).

Section 2: When the user clicks on Open and chooses a file, that file is read from the hard drive, and a copy of the file is loaded into RAM.

Section 3: As the user is editing the file, the keystrokes are processed by the system and those keystrokes are recorded in the copy of the document that is in RAM. Notice, these changes are *not* being recorded in the copy of the file on the hard drive.

Section 4: When the user clicks on Save, then the changes to the document are written to the copy of the file on the hard drive. Until the changes are saved, the version of the document on the hard drive remains unchanged.

Section 5: Whenever the user either closes the document or exits out of Microsoft Word, the document is removed from RAM, but the modified version of the document remains on the hard drive. The version of the document on the hard drive will be whatever was last saved to the disk. Note: If the user both closes the document and exits Microsoft Word, then both the document and the Word application are removed from RAM. If the user closes the document but does not close the Word application, then the document is removed from RAM, but the Word application is still running in RAM.

> ***~~It is vital to remember~~*** for this explanation and many others to come in this course as well as throughout your use of information technology … You store data on a hard drive, but when you process it, it must be loaded into RAM. Likewise, software is stored on the hard drive of a computer, but when you execute that software, it is loaded into RAM. All processing on a computer system, *without exception*, occurs in Random Access Memory (RAM).

Note that here we show the Microsoft Word application loaded into RAM. In reality, there are many many things in RAM all at once. The operating system is in RAM, your Anti-Virus software is in RAM, any other background applications you may be using are in RAM, all software you are currently using (Word, PowerPoint, Email, games, etc.) are all in RAM. The Random Access Memory of a modern computer is a very crowded place. This is why having more RAM on a computer system today is so important.

Section #'s on the slide match those in the notes…

Computer Operation: Hard Drive vs. RAM

MS Word in RAM

Open File

File In RAM

Edited File In RAM

Save File

Close File / Exit Word

8GB Random Access Memory (RAM)

Now is the time

Now is the time for all good Men

Now is the time for all good Men

File & Word are removed from RAM

Central Processing Unit (CPU)

MS Word executes

Read file into RAM

Keystrokes processed: Recorded in RAM document

Write File to Disk

500GB Hard Disk Drive (HDD)

MS Word on disk

Now is the time

File.doc on disk

Now is the time

File.doc Unchanged on disk

Now is the time for all good Men

File.doc Modified on disk

Now is the time for all good Men

File.doc Remains on disk

Section 1

Section 2

Section 3

Section 4

Section 5

Lab Time

- LAB 2.1: Converting Number Systems

- Objective:
  - Practice converting decimal/binary/hexadecimal numbers
  - Estimated completion time: 20 minutes

This page intentionally left blank.

End Module: Computer Numbers and Math …

There are only 10 kinds of people in the world.

Those who understand binary and those who don't.

End Module: Computer Numbers and Math …

In this module, you learned the basics of decimal, binary, and hexadecimal numbering systems. If this information makes sense to you, then the joke on the slide will make sense as well.

Module 6: Networking 101

- Introduction to Network Communications
- Network Types and Terms
- Network Hardware
- Address Resolution Protocol (ARP)
- What Protocols Are

COURSE ROADMAP

- Module 5: How Computers Work
  - Lab 2.1: Converting Number Systems and Decoding ASCII
- **Module 6: Networking 101**
- Module 7: Networking 102
  - Lab 2.2: Networking

Module 6: Networking 101

In this section, we explain computer networking in the most basic possible terms. We begin by leading you through a plain English explanation of how computers communicate on a network. From there, we move into a discussion of network types, standards, and hardware. That leads us into explaining what protocols are and why we care. With that knowledge, we will be ready to understand the fundamental concepts of encapsulation, addressing, and ports.

Networking Coverage Note

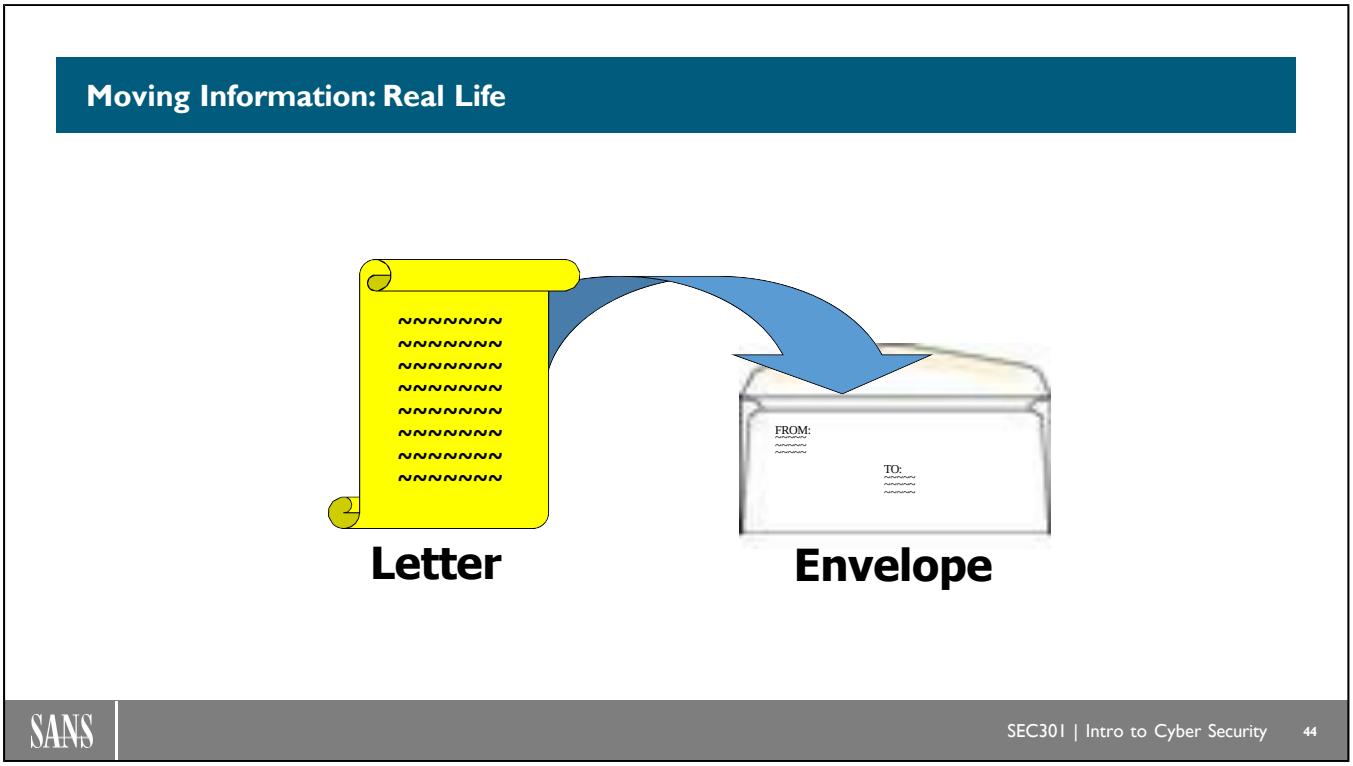
- This course will not make you an expert on networking
  - Network fundamentals is a full five-day class
  - Network expertise requires weeks of training and years of experience
    - We will not attempt to replicate that in part of one day
- What follows covers the basics you need to understand the remainder of this course
  - We will leave out a significant amount of detail
- If you need more:
  - *Networking All-in-One For Dummies, 7th Edition*
  - ISBN-13: 978-1119471608
    - A very good introduction to networking in just under 1,000 pages

Networking Coverage Note

Networking is not a trivial topic to understand. A network fundamentals class is five very full days. Anyone who can claim true expertise has attended several weeks of training. More importantly, they have several years of experience working on networks. There is no way this class can accomplish the equivalent of that in part of one day's class-time. So we won't even try.

What follows here are the bare essentials about how networks work. We will cover enough so that you can understand the remainder of this course. Anything more is beyond the scope of this class.

If your work requires you to have a deeper understanding of computer networking, a very good starting point is the book *Networking All-in-One for Dummies, 7th Edition* (ISBN-13: 978-1119471608). While some have concerns about reading a "For Dummies" book, that really is a good resource once you are done with the material in this book.



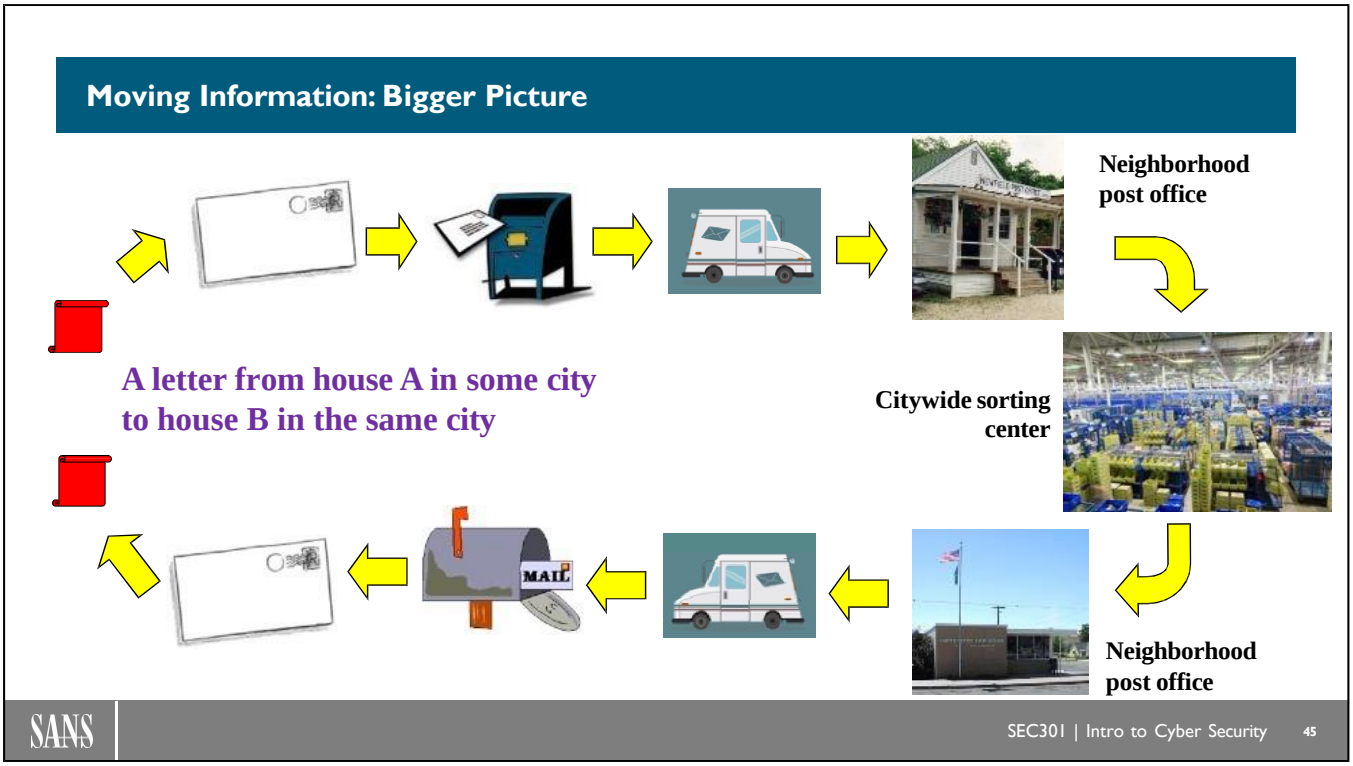
Moving Information: Real Life

The post office is a good example of how information moves from one place to another. Think about sending a letter through the Postal Service.

You write your letter and want it to go to the proper recipient. This means you need some way of telling the post office where you would like your letter to go. You, therefore, put your letter inside an envelope and seal it. You then write the *To* or destination address on the outside of the envelope.

You also need to place your own *From* or source address on the envelope. Doing so serves two purposes. First, if there is a problem with delivery and the Postal Service needs to return your letter to you, they know where it came from. Second, if your friend you are communicating with wants to write back to you, they need to know your address for their response.

We fully realize you learned all this many years ago, but we need to have some old concepts fresh and new again in our minds.

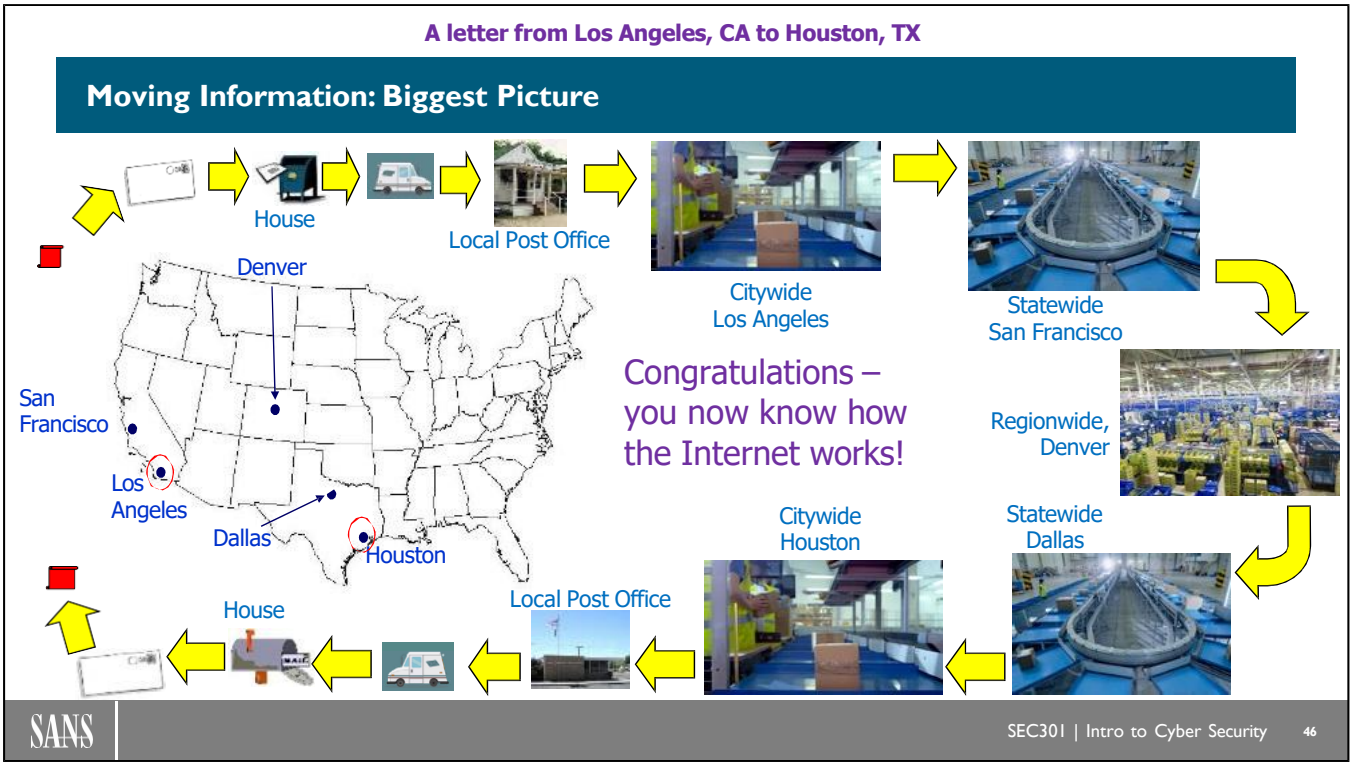


Moving Information: Bigger Picture

Let's continue to follow the analogy of the letter. After you write the letter and address the envelope, you place it into your local mailbox. From there:

- The Postal Service picks up the letter from the mailbox in its truck.
- The truck takes the letter to the local post office.
- The *To* address is not for anyone who receives mail via that local post office.
- The letter is forwarded to the citywide sorting facility.
- That facility sends the letter to a different local post office across town that serves the addressee.
- That local post office places the letter on a mail truck.
- The mail truck delivers your letter to the recipient's mailbox.
- The recipient receives the letter and reads it.

That is all straightforward. But what if the letter is not going across town? What if the letter is going across the country? The process gets a little more complicated at that point.



Moving Information: Biggest Picture

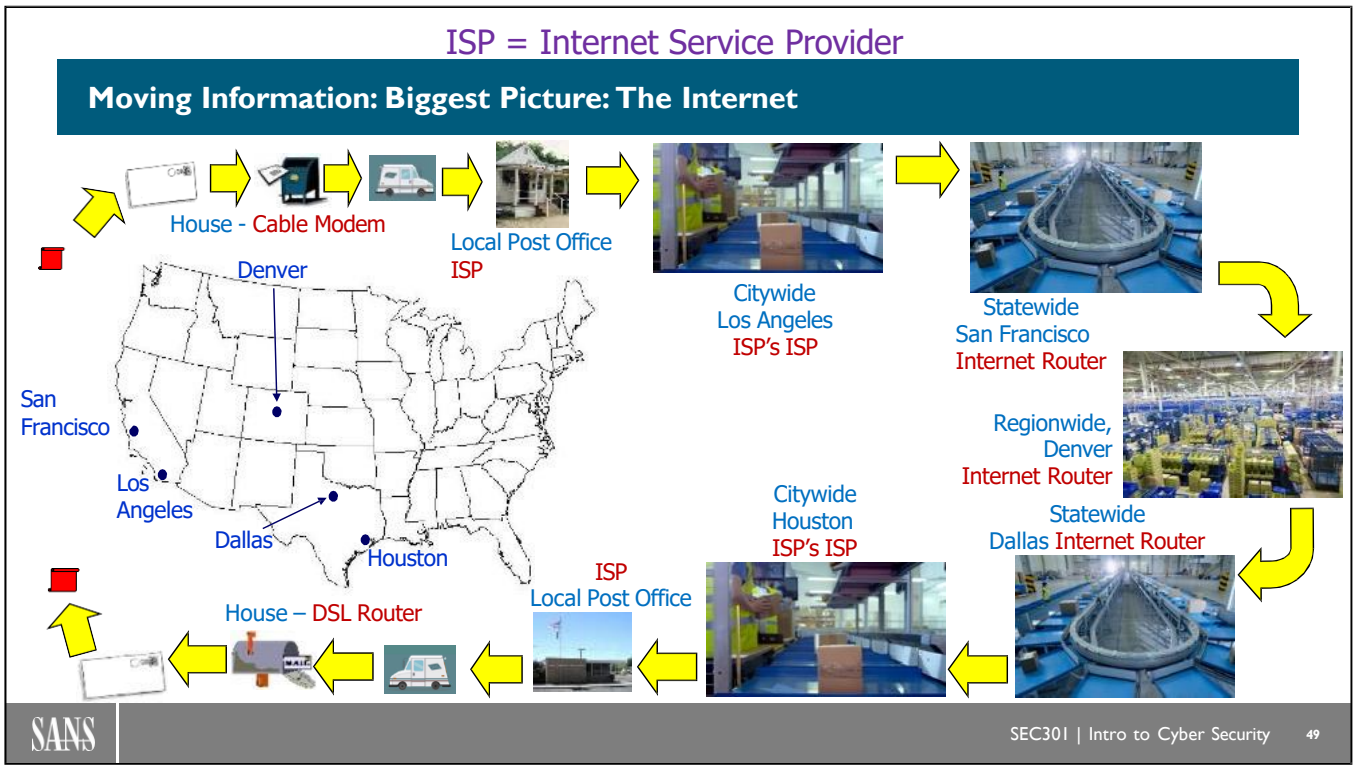
For cross-country delivery of a letter, the process has to expand significantly.

- The letter is addressed and placed in the mailbox.
- The mail truck picks up the letter and takes it to a local post office.
- The local post office sees the destination address and says, “I have no idea where that address might be. I need to forward it to my default location of the city wide sorting center.”
- The letter arrives at the Los Angeles citywide sorting center who looks at the destination address and says, “I have no idea where this address is, I will send it to my default location of the statewide sorting center." (San Francisco in this example)
- The letter arrives at the statewide sorting center who looks at the destination address and says, ”I have no idea where the destination address is so I will send it to my default location of the regional sorting center." (Denver in this example)
- The letter arrives at a regionwide sorting center who says, “I do not know where the destination address is, but I know it is somewhere in the state of Texas, therefore, I am supposed to forward the letter to the statewide sorting center in Dallas.
- The letter arrives at the Texas statewide sorting center in Dallas who says, “I do not know where this address is, but I know it is somewhere in Houston, so I should send it to the citywide sorting center in Houston.
- The letter arrives at the Houston citywide sorting center who says, “I don’t know exactly where this address is, but I know which local post office services that neighborhood, I will forward the letter there.”

- The local post office says “I don’t know exactly where this house is, but I know it is on a street serviced by a particular mail truck, so I will put the letter on that truck.”
- The mail truck driver looks at the house number, and places the letter in the correct mailbox.

A letter from Los Angeles, CA to Houston, TX

Moving Information: Biggest Picture



Moving Information: Biggest Picture: The Network

Believe it or not, the internet is similar at a conceptual level to the postal delivery we just discussed. Yes, the technical details are radically different, but the idea of "sorting facilities" looking at a "To address" to determine where to send the information to get it closer to its destination is the same.

If we send data from a house in Los Angeles, California to a data center in Houston, Texas, it all happens pretty much the same way from a logical perspective. It is, of course, very different from a technology perspective:

- The local PC (in Los Angeles) has no idea where the destination is, so it sends the message out of the house through the cable modem.
- It arrives at the sender's Internet Service Provider (ISP), who has no idea where the destination is, so they send it to their default location, the ISP's ISP.
- The ISP's ISP looks at the destination address and does not know were it is located, but knows it is not located in Los Angeles, California. The ISP's ISP sends it to their default location of an internet router in San Francisco.
- That San Francisco router sees the message is not destined for anywhere in California, so it forwards it's default location, which happens to be another internet router—let's say in Denver Colorado.
- The Denver router does not know where the destination is, but knows the destination is somewhere in the state of Texas, so it forwards the message to a router in Dallas.

- The Dallas router does not know where the final destination is, but knows the destination address is in Houston, Texas, so it forwards the message to an ISP's router in that city.
- The Houston router does not know where the final address is, but does know the destination address is for customers of a particular ISP, so forwards the data to that ISP's router.
- The recipient's ISP delivers the message to the final recipient data center.

Again, while the detail behind how mail is physically delivered cross-country and how data is delivered over the internet is quite different at the technical level, they are extremely similar conceptually. Please notice that in both examples, the mail / data went to default locations until it reached a point where someone / something knew a better place to send it. In other words, in this example, each point forwarded the information to a default location until it reached Denver, who knew to send it to Texas, where Dallas knew to send it Houston, who knew which ISP to send it to. This combination of defaults and known destinations is a very important part of how the Internet works, and a big part of our discussions later in this book.

Forwarded to default location.

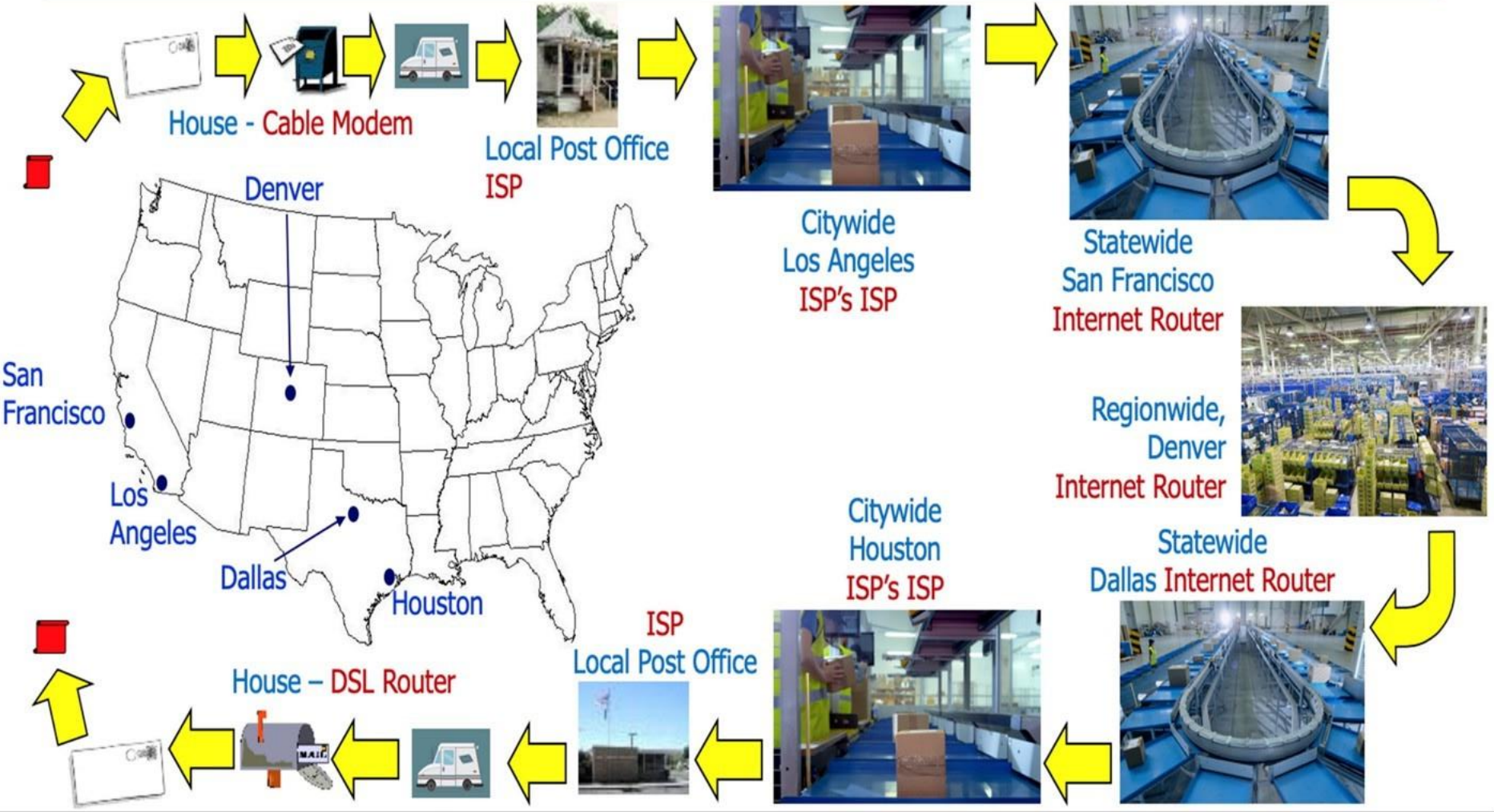
- Cable modem sent to its default of the ISP
- ISP sent to its default of the ISP's ISP
- ISP's ISP sent to its default of the San Francisco Internet Router
- San Francisco Internet Router sent to its default of Denver Internet router

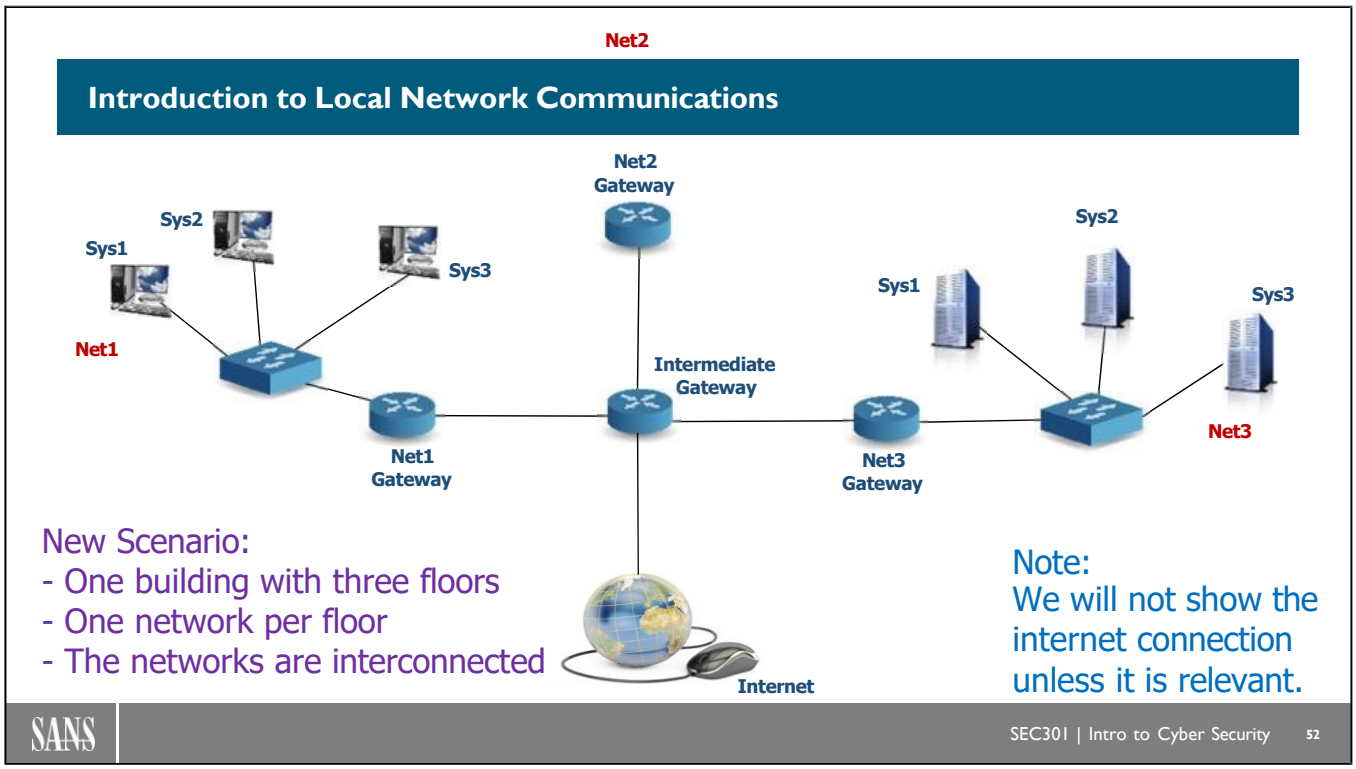
- Denver Internet router did not use a default—it knew to send it to Dallas
- Dallas knew to forward it to the ISP's ISP in Houston
- The Houston ISP's ISP knew to send it to the customer's ISP
- The customer's ISP knew to deliver it to the customer

Forwarded based on knowledge.

ISP = Internet Service Provider

Moving Information: Biggest Picture: The Internet





Introduction to Local Network Communications

We now have the basic premise down, specifically:

- Information has a destination address
- Devices on a network look at that address and make decisions on where to send the information
- The intent is to always move the data closer to its final destination

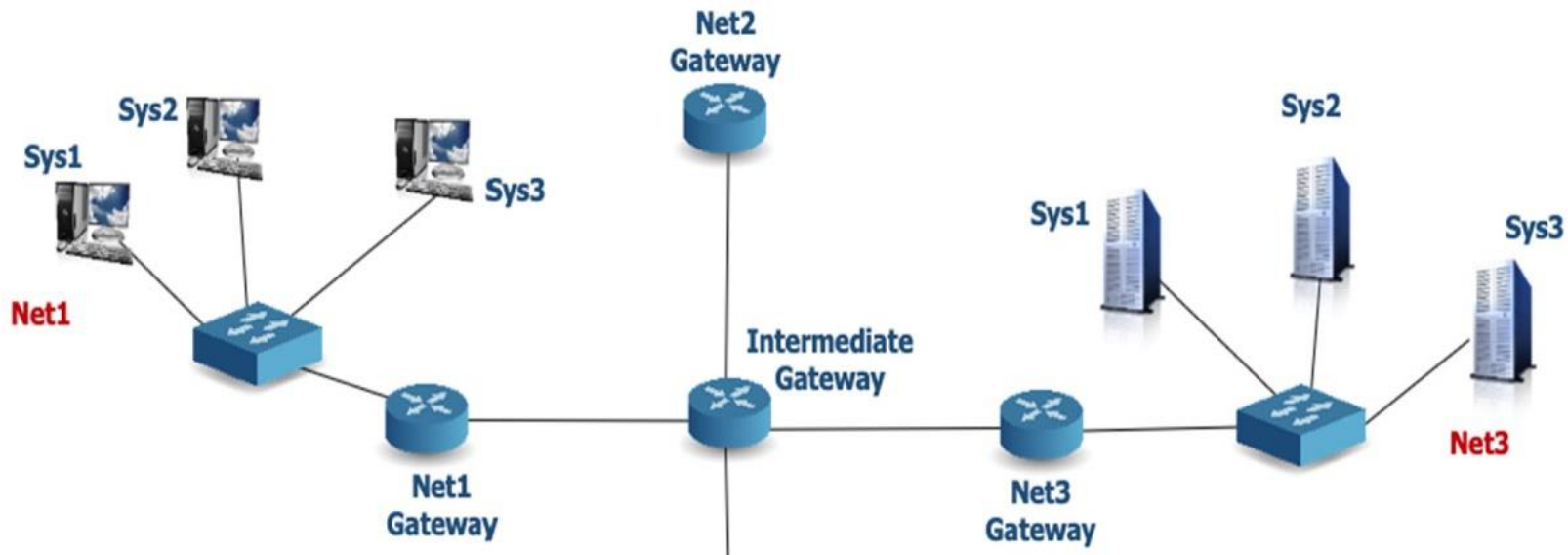
Now let's introduce a new scenario for the purpose of explaining local network communications. For this example, assume that we have one three-story building. Each floor of the building has its own network. Those three networks are interconnected via intermediate devices.

On the next several pages is a greatly simplified explanation of network communications. In this explanation, we avoid technical jargon and details that might cloud understanding. The goal here is to understand the underlying process, not the details of how that process works. We add some of those details as the course progresses.

To begin explaining the network communications process, we have the above network diagram. Notice that there are three networks depicted: Net1, Net2, and Net3. (And as stated, we have one of those networks on each of three floors of a building.) Each network has three systems (or computers) of one type or another (for space and readability, we do not show the systems on Net2 here).

Also notice that all these networks are part of a single, larger network within an organization. For simplicity, we show a single Intermediate gateway system but understand that, in reality, there could be many such systems that the data must pass through to reach a given destination. For the example here, *Net1 System2* needs to send data to *Net3 System2*. The actual process begins on the next page.

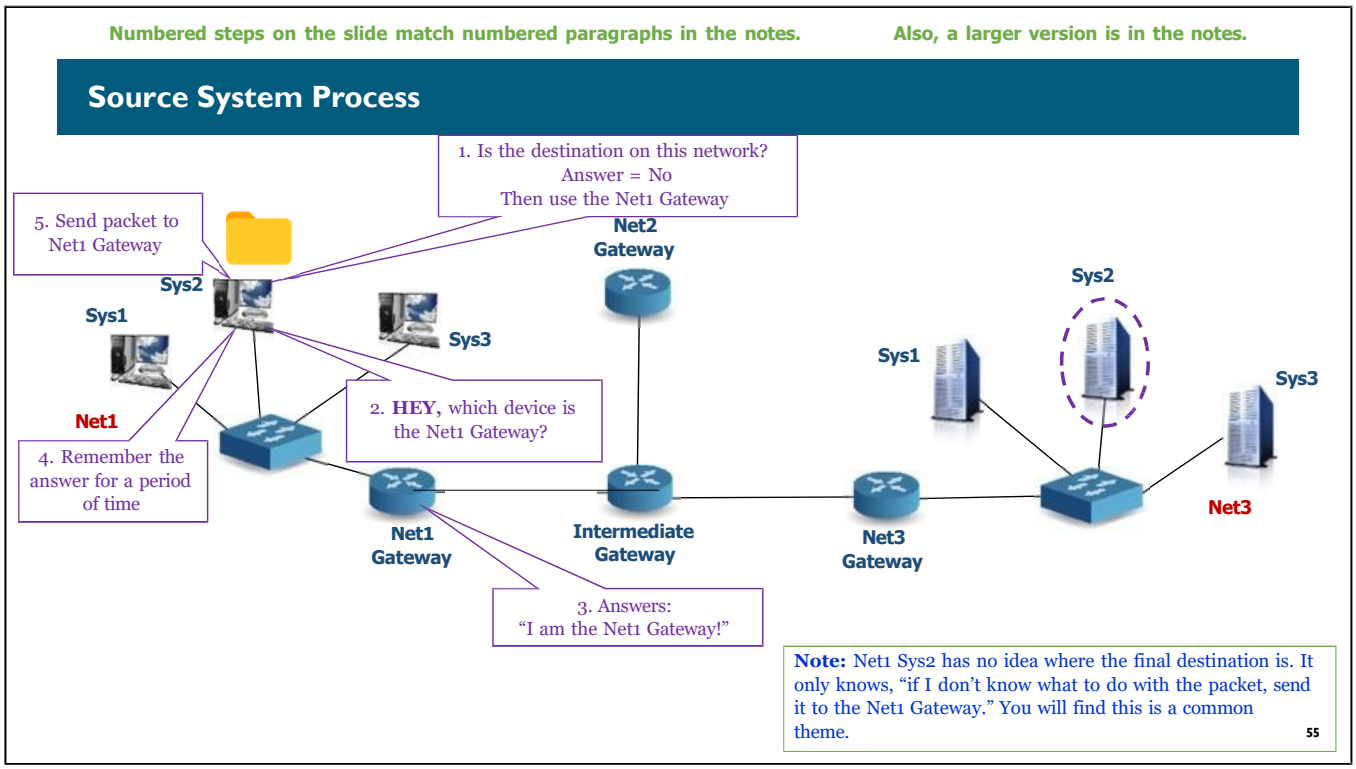
Introduction to Local Network Communications



New Scenario:
- One building with three floors
- One network per floor
- The networks are interconnected

Internet

Note:
We will not show the Internet connection unless it is relevant.



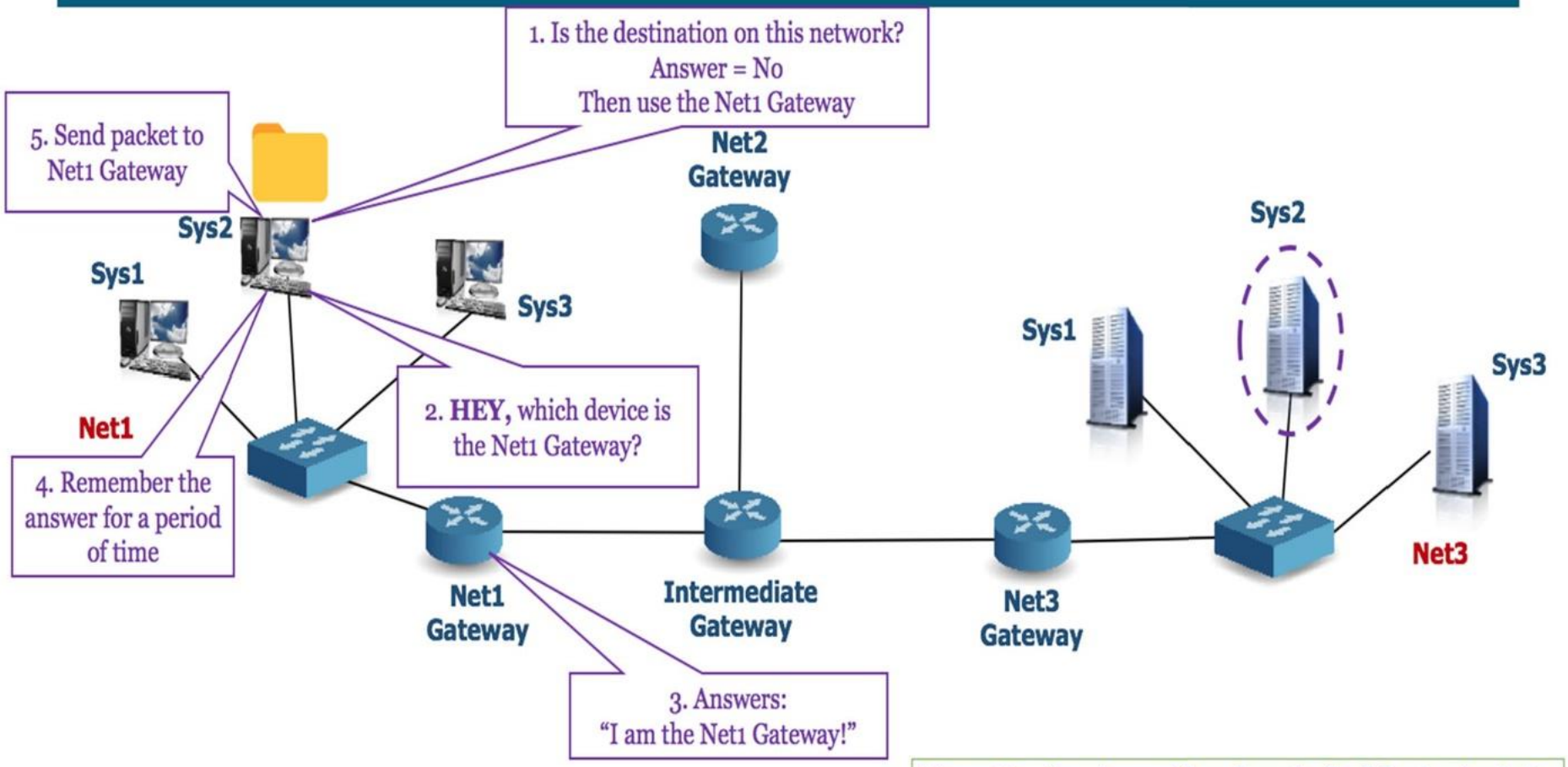
Source System Process

The process actually begins when the source system realizes it has data destined for a machine other than itself and must send it across the network. The source system (like all the systems in the diagram) has network configuration information on it. It utilizes that configuration information as well as information in the packet headers to make determinations below.

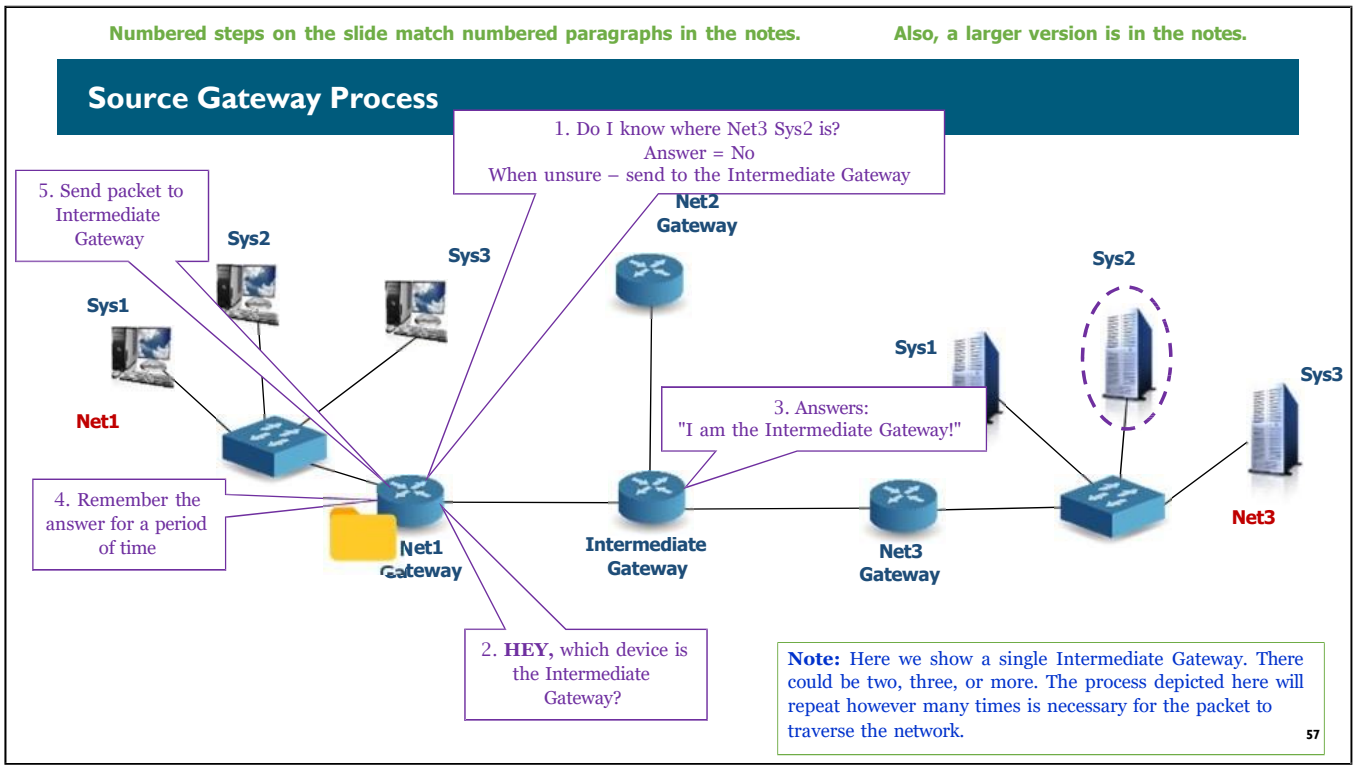
1. The source system (Net1, Sys2) asks, "Is the destination address on this network"? The answer is "No," meaning that Net1 Sys2 cannot send the packet directly to the destination. So, it knows it needs to send the packet to the Net1 Gateway.
2. In order to send the packet to the Net1 Gateway, it must know where it is physically located on the network. So Net1, Sys2 shouts out on the network, "HEY, which device is the Net1 Gateway?" The other systems on Net1 hear this shout but ignore it, since it does not pertain to them.
3. The Net1 Gateway hears the shout on the network and sends a response back to Net1, Sys2, saying, "I am the Net1 Gateway!"
4. Net1, Sys2 receives the response from the Net1 Gateway and will remember that information for a period of time.
5. With that information in hand, Net1, Sys2 is able to physically transmit the data packet to the Net1 Gateway. Note that Net1 Sys2 does not know where the final destination is. It only knows, "If I don't know where the destination is, I should send it to the Net1 Gateway."

Numbered steps on the slide match numbered paragraphs in the notes. Also, a larger version is in the notes.

Source System Process



Note: Net1 Sys2 has no idea where the final destination is. It only knows, "if I don't know what to do with the packet, send it to the Net1 Gateway." You will find this is a common theme.



Source Gateway Process

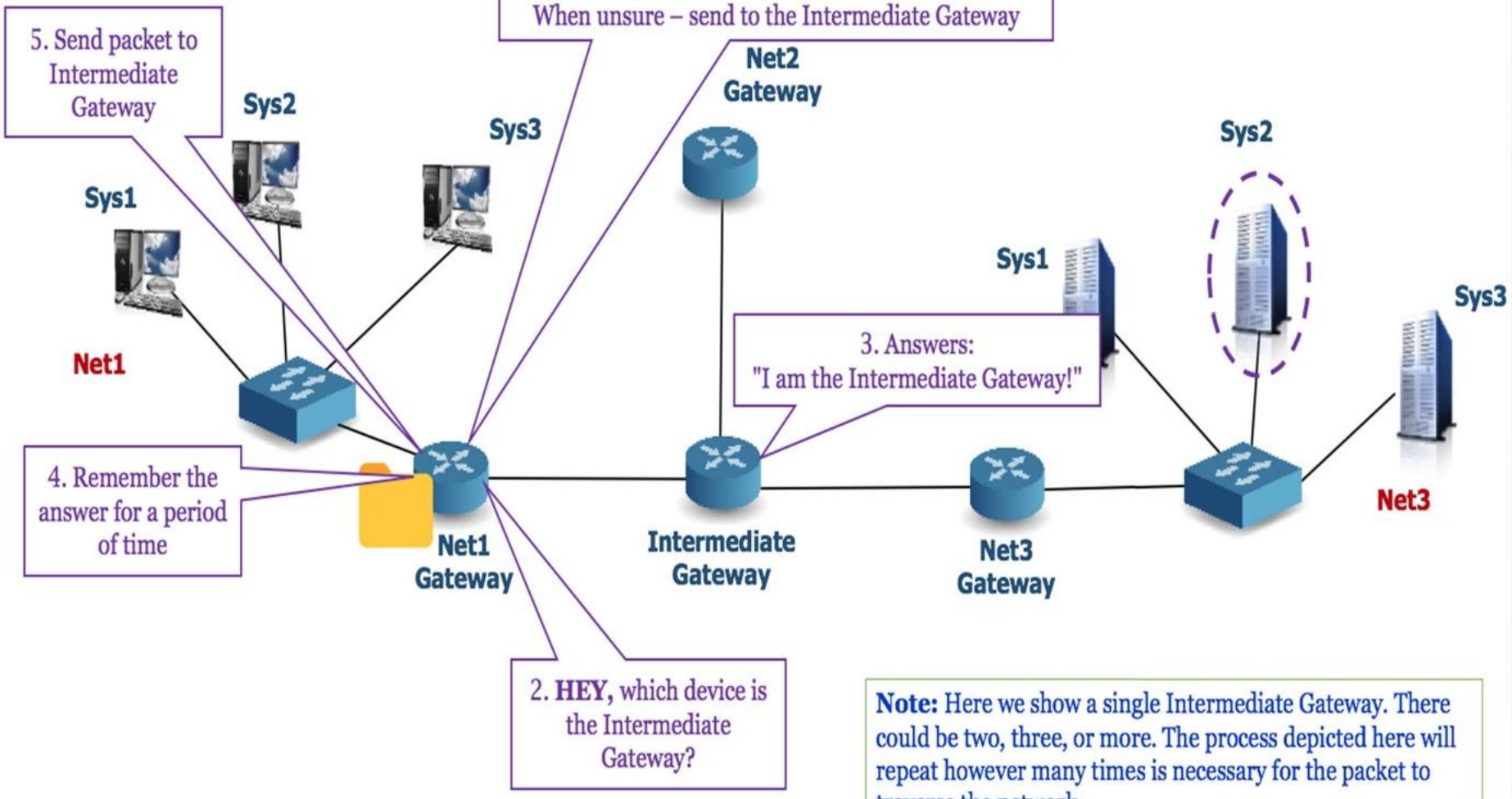
The data is now on the *Net1 Gateway* system (the Default Gateway for the source system, *Net1 Sys2*). The Net1 Gateway looks at the packet header information to determine its destination, which is *Net3 System2*. It then begins the process below:

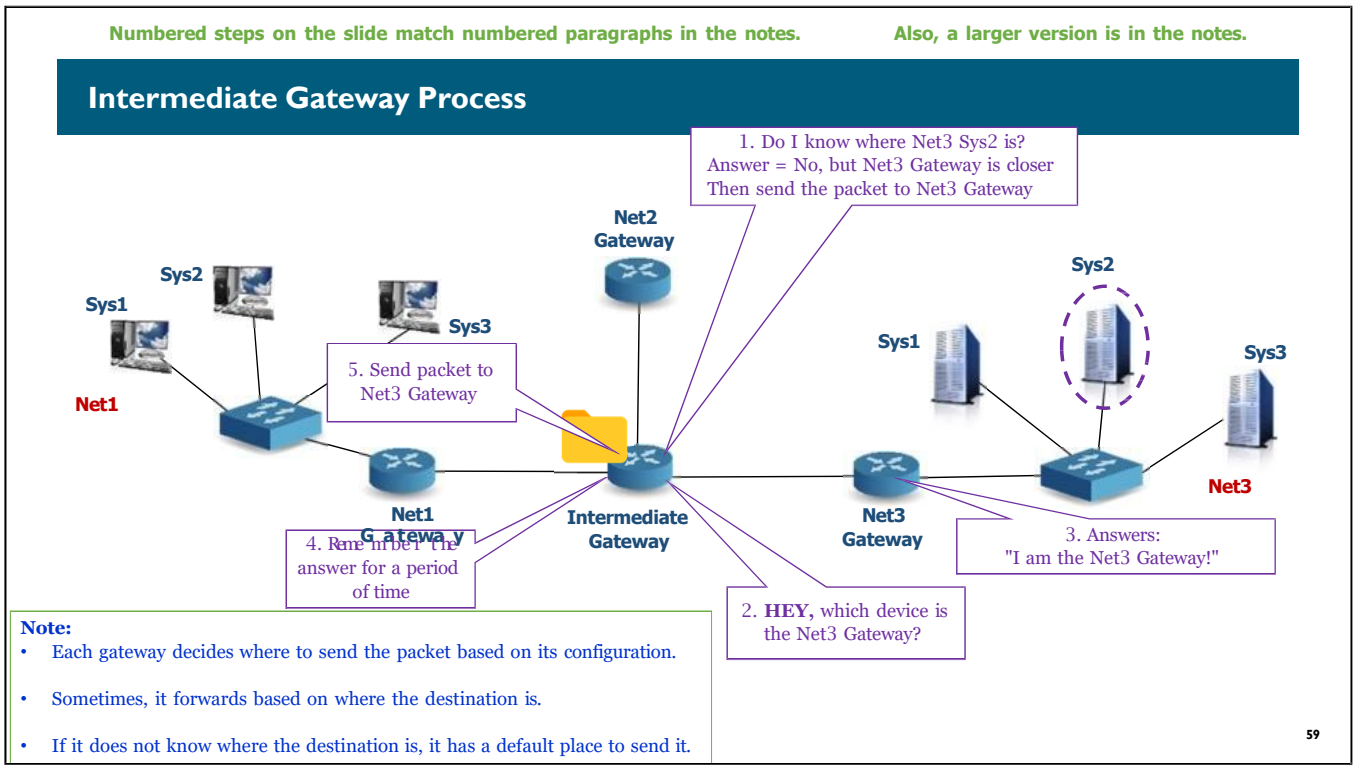
1. The Net1 Gateway asks, "Do I know where Network 3, System 2 is"? The answer is "No". When I don't know where the destination is, my configuration says to send the data to the Intermediate Gateway.
2. In order to send the packet to the Intermediate Gateway, it must know where it is physically located on the network. So, the Net1 Gateway shouts out on the network, "HEY, which device is the Intermediate Gateway?" If there were other systems on this network segment, they would ignore the shout since it does not pertain to them.
3. The Intermediate Gateway hears the shout on the network and sends a response back to Net1, Gateway saying, "I am the Intermediate Gateway!"
4. The Net1 Gateway receives the response from the Intermediate Gateway and will remember that information for a period of time.
5. With that information in hand, Net1 Gateway is able to physically transmit the data packet to the Intermediate Gateway.

As noted on the slide, there could be any number of intermediates. This process repeats as many times as necessary for the data to reach its destination.

Numbered steps on the slide match numbered paragraphs in the notes. Also, a larger version is in the notes.

Source Gateway Process





Intermediate Gateway Process

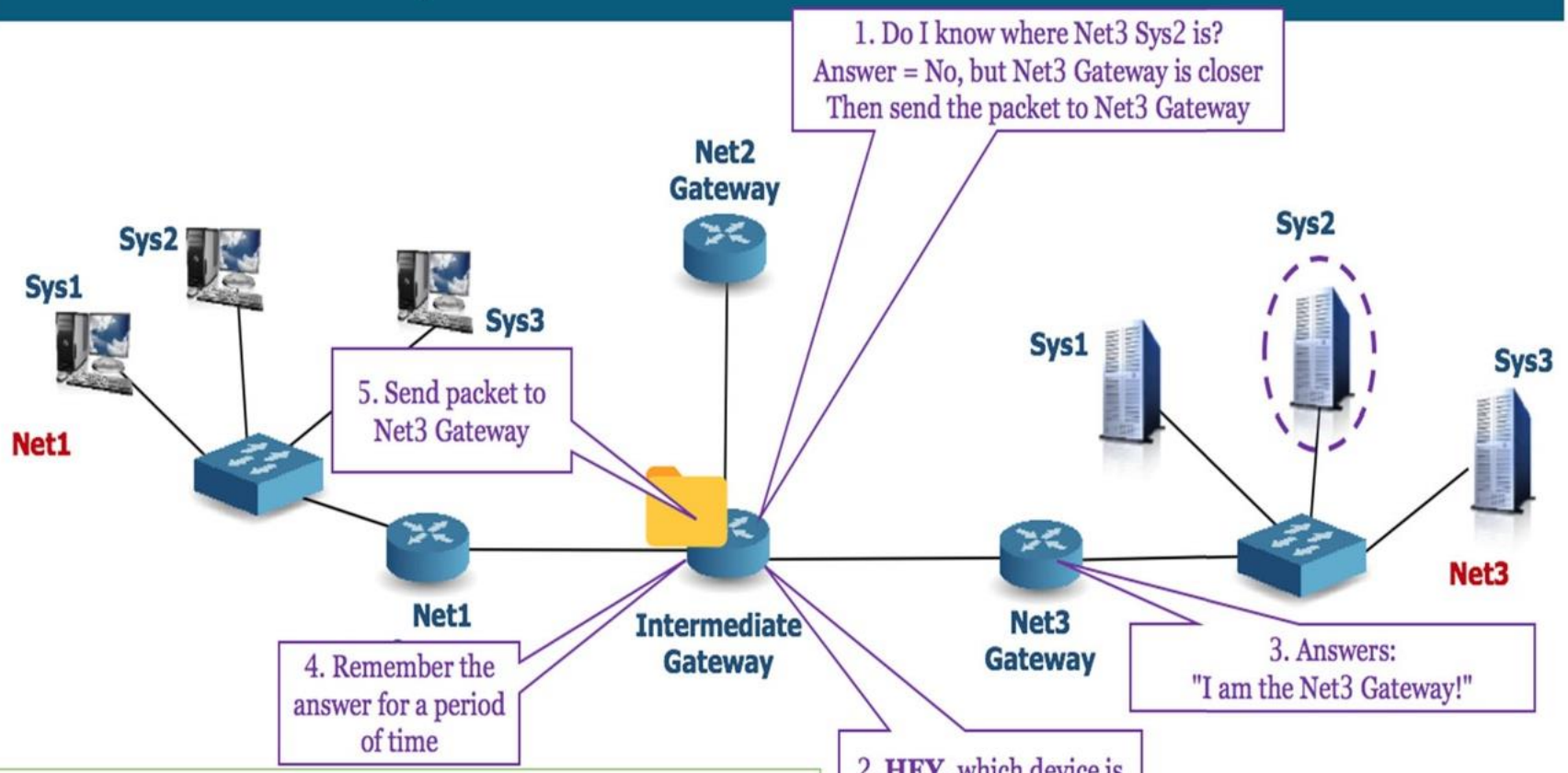
With the data at the *Intermediate Gateway*, the process continues.

1. The Intermediate Gateway asks, "Can I go direct to Net3, System2?" The answer is "No," but my configuration says Net3 Gateway is closer. So, send the packet to the Net3 Gateway.
2. In order to send the packet to the Net3 Gateway, it must know where it is physically located on the network. So, the Intermediate Gateway shouts out on the network, "HEY, which device is the Net3 Gateway?" If there were other systems on this network segment, they would ignore the shout, since it does not pertain to them.
3. The Net3 Gateway hears the shout on the network and sends a response back to Intermediate Gateway saying, "I am the Net3 Gateway!"
4. The Intermediate Gateway receives the response from the Net3 Gateway and will remember that information for a period of time.
5. With that information in hand, Intermediate Gateway is able to physically transmit the data packet to the Net3 Gateway.

As noted on the slide, you have probably noticed that this seems highly repetitive. Fortunately, the electrons on the copper that makes all of this happen are moving at just a tiny fraction below the speed of light, so it is still a very quick process.

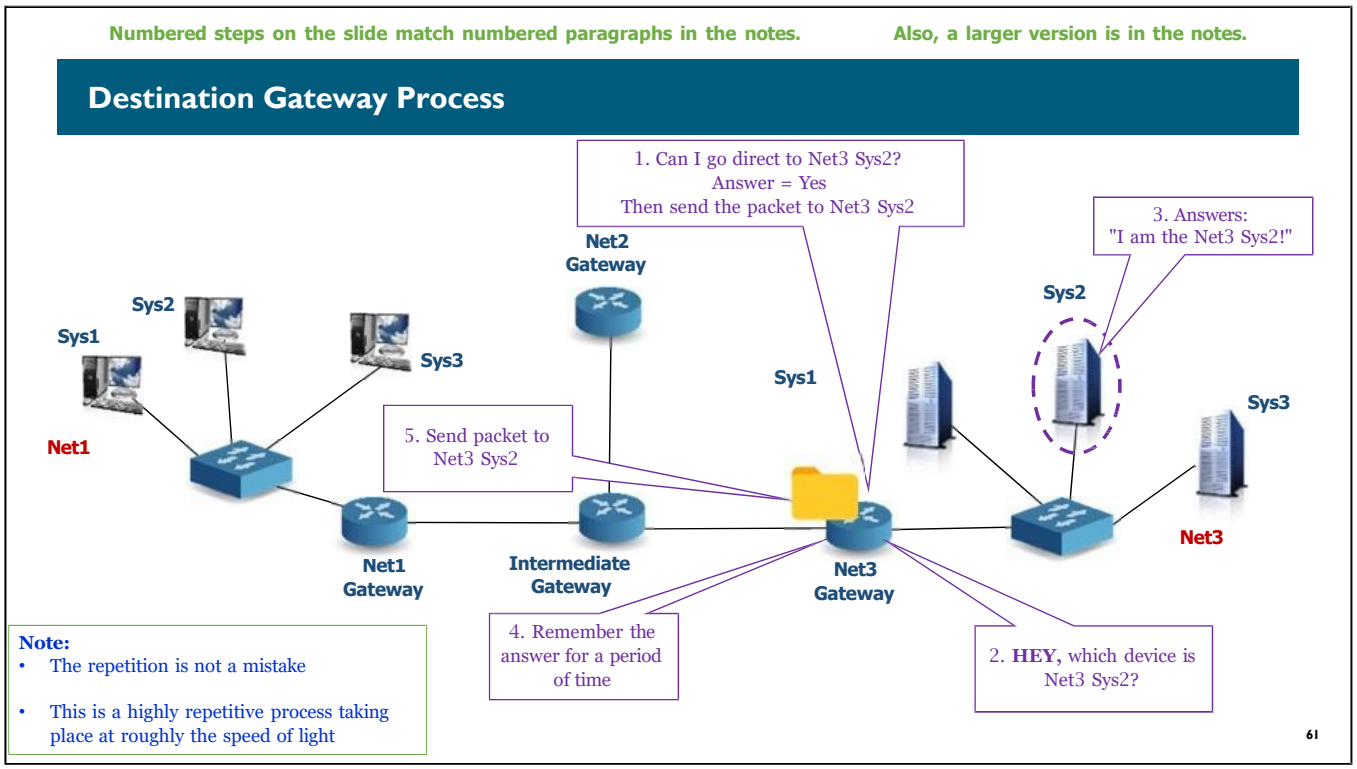
Numbered steps on the slide match numbered paragraphs in the notes. Also, a larger version is in the notes.

Intermediate Gateway Process



Note:

- Each gateway decides where to send the packet based on its configuration.
- Sometimes, it forwards based on where the destination is.
- If it does not know where the destination is, it has a default place to send it.



Destination Gateway Process

When the packet arrives at the Net3 Gateway, the process repeats as follows:

1. The Net3 Gateway asks, "Can I go direct to Net3, System2?" The answer is "Yes." So, send the packet to Net3 Sys2.

2. In order to send the packet to the Net3 Sys2, it must know where it is physically located on the network. So, the Net3 Gateway shouts out on the network, "HEY, which device is the Net3 Sys2?" Net3 Sys1 and Sys3 hear the shout, but they ignore it since it does not pertain to them.

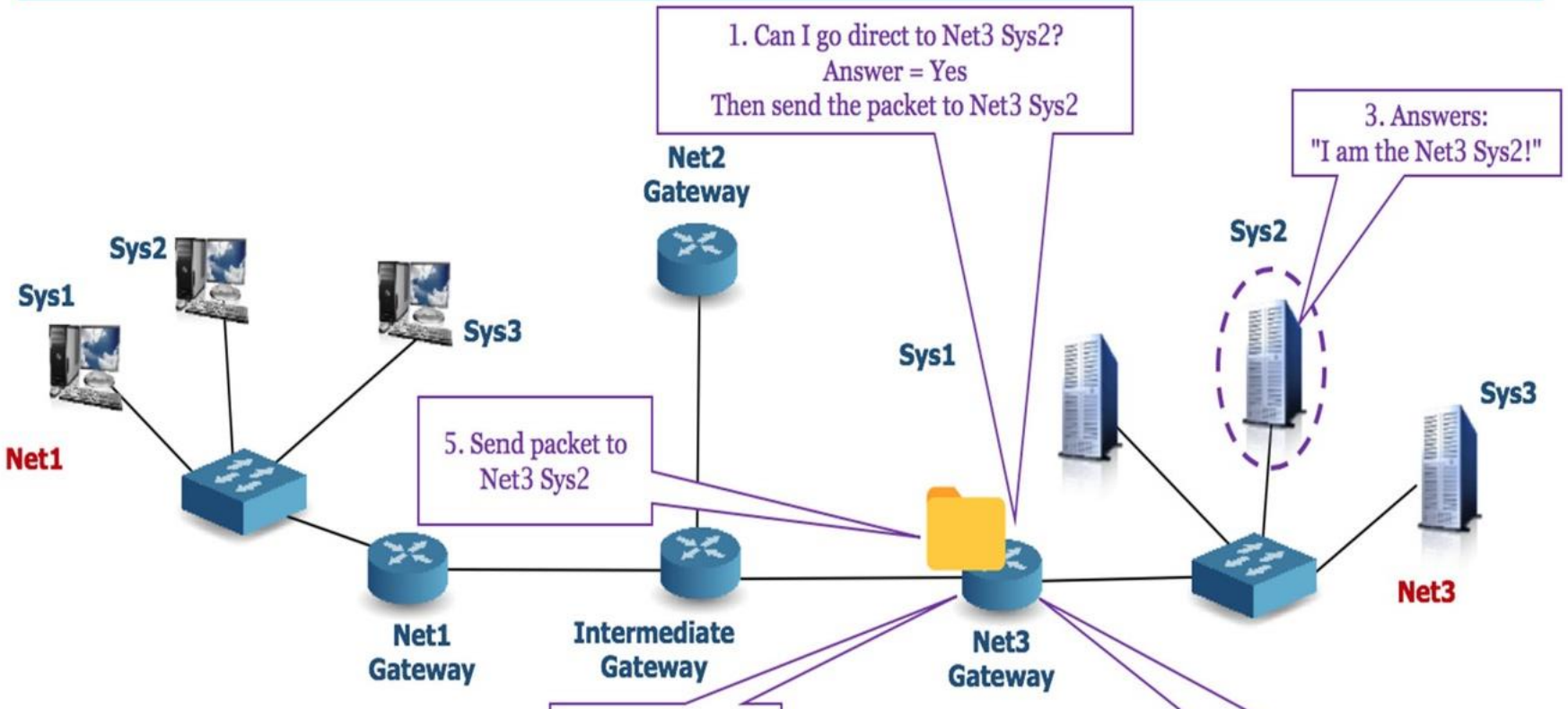
3. The Net3 Sys2 hears the shout on the network and sends a response back to Net3 Gateway saying, "I am the Net3 Sys2!"

4. The Net3 Gateway receives the response from Net3 Sys2 and will remember that information for a period of time.

5. With that information in hand, the Net3 Gateway is able to physically transmit the data packet to the Net3 Sys2. The recipient has now received the data and processes it.

Numbered steps on the slide match numbered paragraphs in the notes. Also, a larger version is in the notes.

Destination Gateway Process

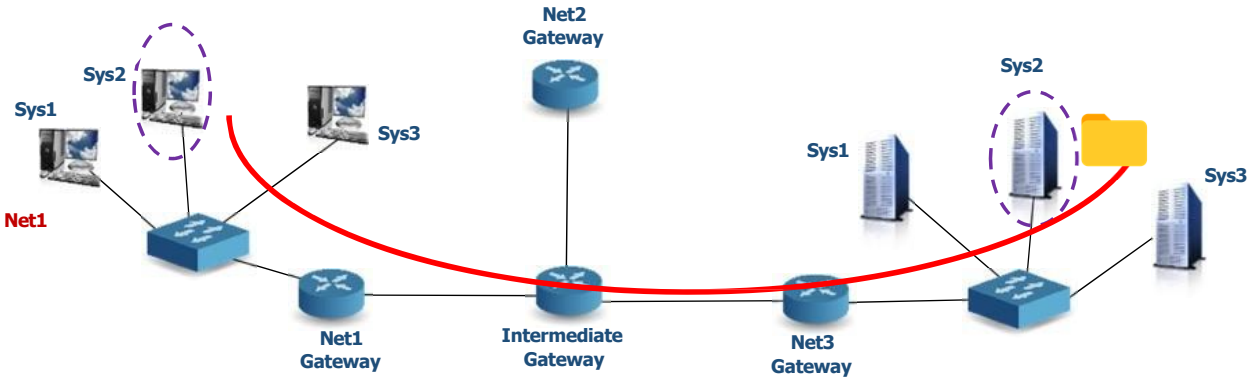


Note:

- The repetition is not a mistake
- This is a highly repetitive process taking place at roughly the speed of light

A larger version is in the notes.

Response Process



Response: The process repeats in the opposite direction. The same questions are asked by each system, the same "shouts" occur, systems remember the answers, and the packet physically transfers from device to device.

BUT, since each system remembers the answer to the "shout," further communications do not require the shouting so long as it happens within the timeframe that those answers remain in the memory of each device.

Response Process

There may be one more step in the network communications process. In many cases, the destination system returns an acknowledgment to the source, letting the source system know it received the data. Sometimes this does happen, and sometimes it does not. It depends on the type of traffic exchanged.

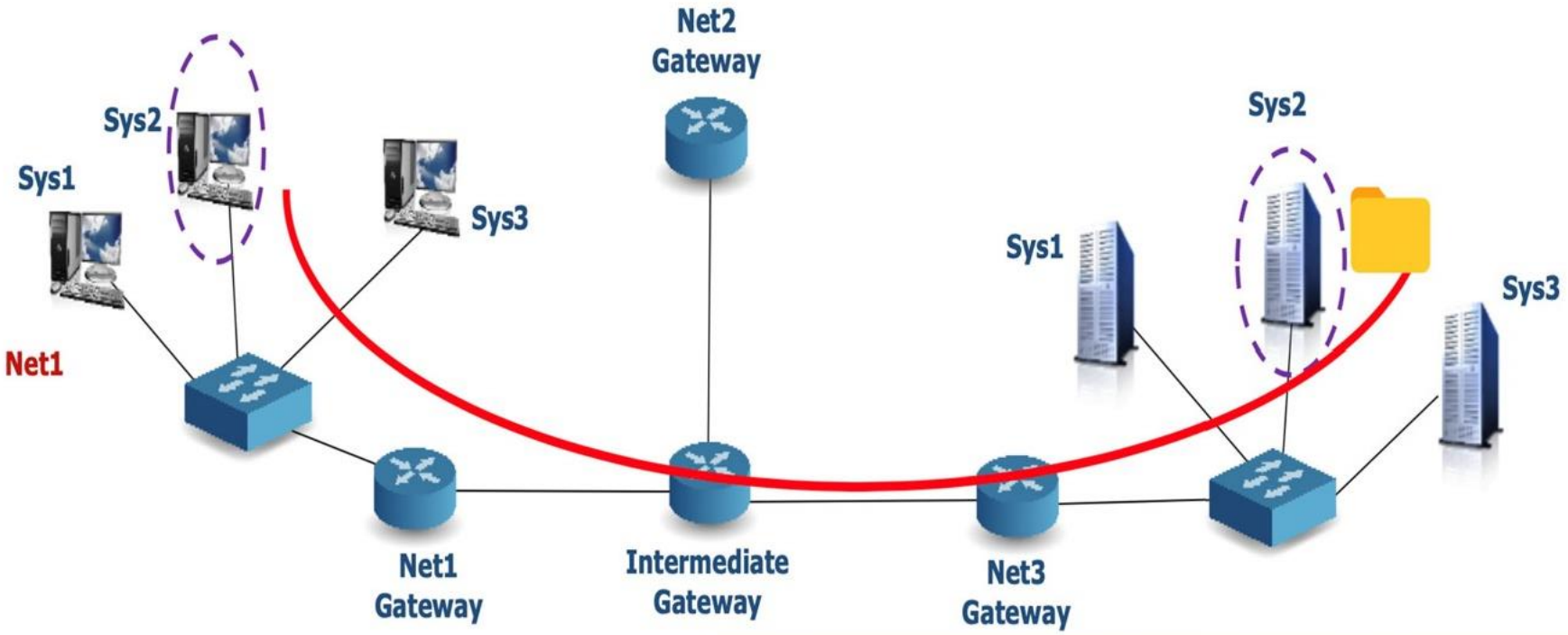
Note that when the destination sends acknowledgments, the network communications process repeats in the opposite direction. *Net3 Sys2* asks if it can send directly to Net1 Sys2. No, it can't. Send the packet to the Net3 Gateway. Shout on the network, "who is the Net3 Gateway?" The Net3 Gateway answers back and Net3 Sys2 remembers the answer for a period of time.

This exact process continues through the Intermediate Gateway, to the Net1 Gateway, and on to Net1 Sys2. There is no difference from what we have already seen, except that everything is happening in the opposite direction.

However, once the communication occurs in each direction, the "shouting" for "who is the gateway," etc. becomes unnecessary. Each device remembers the answer for a period of time. So long as communications continue during that period of time, there is no need to keep asking those questions.

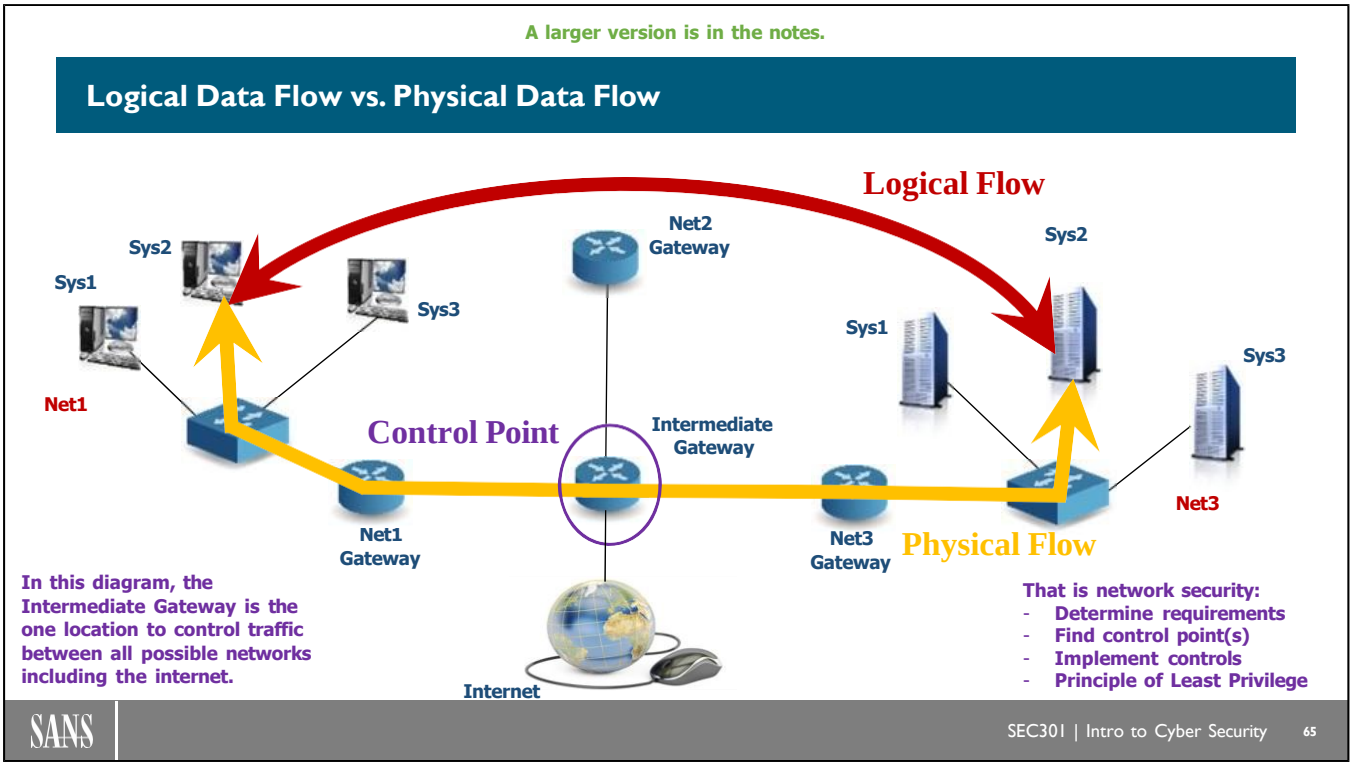
A larger version is in the notes.

Response Process



Response: The process repeats in the opposite direction. The same questions are asked by each system, the same "shouts" occur, systems remember the answers, and the packet physically transfers from device to device.

BUT, since each system remembers the answer to the "shout," further communications do not require the shouting so long as it happens within the timeframe that those answers remain in the memory of each device.



Logical Data Flow vs. Physical Data Flow

Whether you are network operations or network security, when you look at a network, there are two issues you always think about: The logical flow of data and the physical flow of data.

In the example we just went through, Net1 Sys2 needed to communicate with Net3 Sys2. This requirement is a logical flow of data. For that communication to get from the source to the destination, it had to follow the lines and pass through the gateways, etc. This path is the physical flow of data.

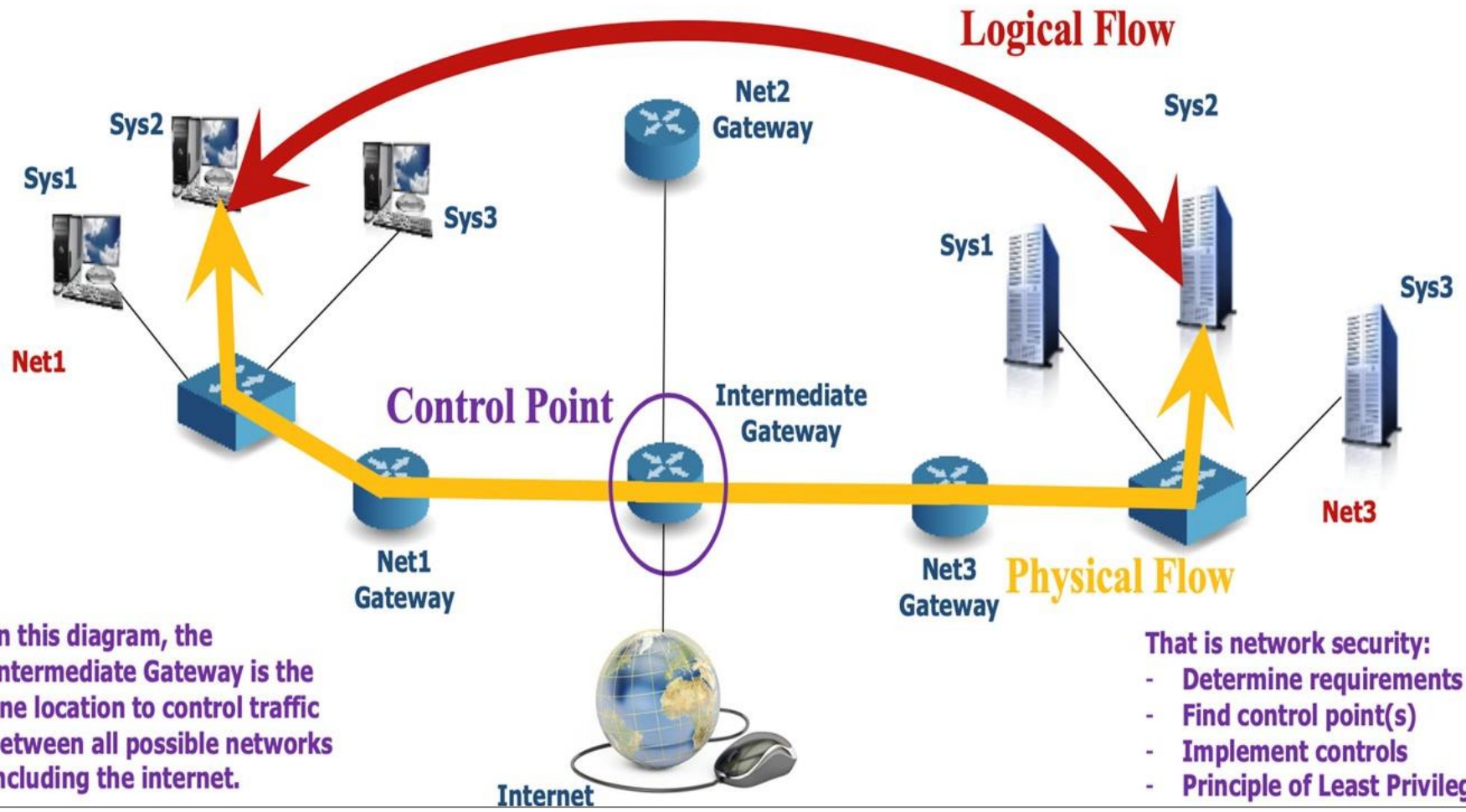
Network operations staff look at the logical flow of data to determine where traffic has to be able to flow on the network. They then look at the physical path to determine if it is stable. For example, are all the cables plugged in, are any of the wires broken, are the devices configured to know where to send data next, and so on.

Network security also looks at both logical and physical flow, but from a different perspective. We also identify the requirement that Net1 Sys2 has to be able to communicate with Net3 Sys2. But we may even go a step further and determine precisely what kind of communication is required. For example, let's say that Net3 Sys2 is a Web server. So now we could specify that Net1 Sys2 must be able to communicate with Net3 Sys2 specifically for web traffic. (Remember the Principle of Least Privilege: "Everyone can do everything they are supposed to be able to do, and nothing more.")

So, with that requirement in mind, we can then go to the physical flow to determine a point where we can control that traffic. At that point, we can put in place mechanisms such as firewall rules that allow everything necessary, but nothing more. In our example diagram above, the Intermediate gateway is the ideal place to put in firewall rules that will enable Net1 Sys2 to communicate with Net3 Sys2 specifically for web traffic and nothing else.

A larger version is in the notes.

Logical Data Flow vs. Physical Data Flow



In this diagram, the Intermediate Gateway is the one location to control traffic between all possible networks including the internet.

That is network security:

- Determine requirements
- Find control point(s)
- Implement controls
- Principle of Least Privilege

Networks

- What is a network and why do I care?
  - *"Two or more computing devices connected together in some way to facilitate communication and exchange of data."*
  - Preferably rapid, efficient, reliable, and secure
  - Much more efficient than ***Sneakernet***
- Size, connection method, and reach do not matte r; it's still a network
- Network operating systems: An OS that has networking built in
- Thick Client and Thin Client networks

Networks

It may sound simplistic to ask, "What is a network and why do I care?" It really isn't. Too many people say they want a network because everyone else has one. They never stop to think about why they want or need that network. The bottom line is that you want a network to support your business needs, whatever those happen to be. You want your computers to help you accomplish your mission.

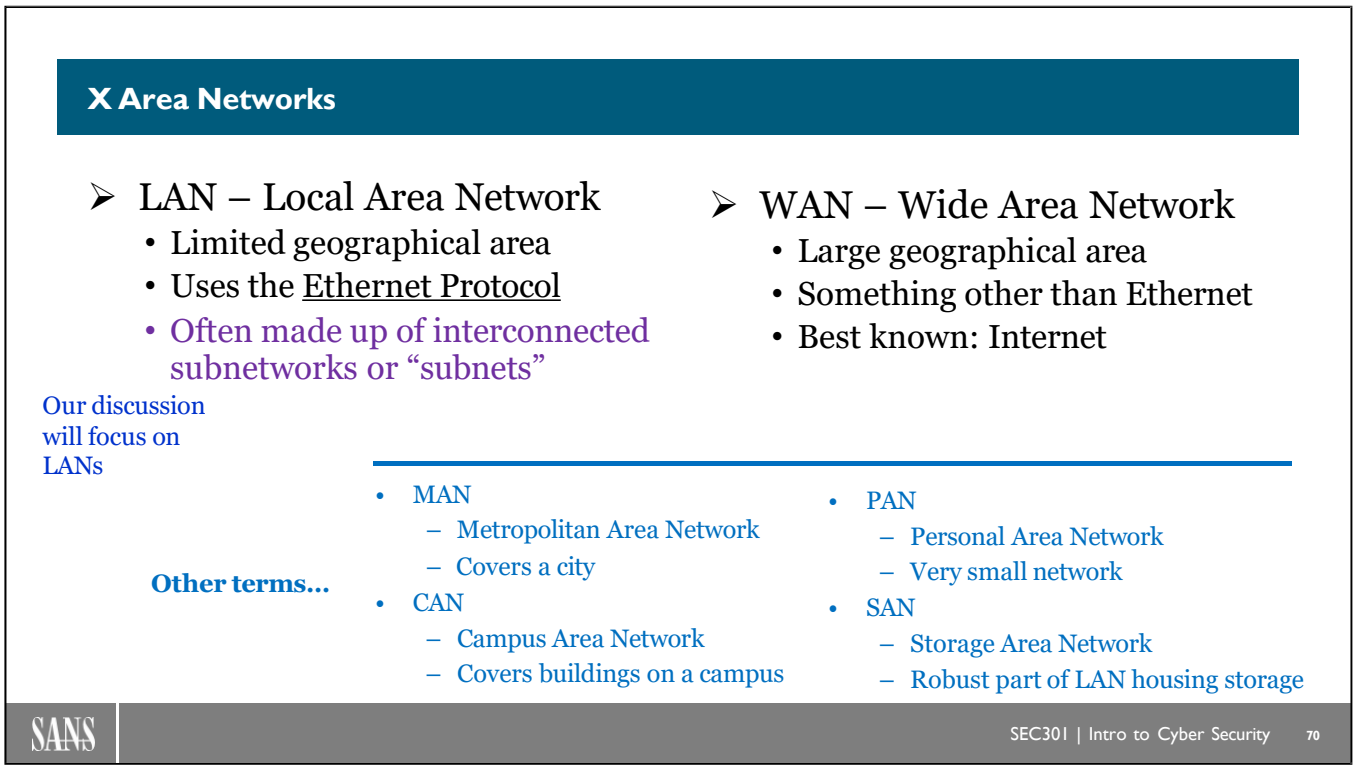
That explains why we want a network, but it doesn't define what a network actually is. For our purpose, the definition of a network is, "Two or more computing devices connected together in some way to facilitate communication and exchange of data." The term *computing devices* can be extremely broad: Desktop PCs, servers, printers, tablets, cell phones, as well as network-specific devices such as routers, switches, hubs, and a ton of other gadgets. The key to this definition is that these devices connect together so that they can communicate, preferably in a rapid, efficient, reliable, and secure way.

With that network in place and functioning properly, we achieve the exchange of data without resorting to the old standby, traditionally referred to as Sneakernet. That is the process of putting data on a thumb drive, CD, or other storage medium and carrying it down the hall (or across the country) to whoever needs it, that is, using your sneakers to transport data, hence the term Sneakernet. (Yes, that really is a term that has been around in networking for many, many years.)

Note that, given the preceding definition, things such as size, connection type, and reach become irrelevant. The network may be hardwired, wireless, or some combination of the two. It may have two systems or a million systems, and it may reach no more than a few feet across the room, or it may reach every corner of the globe. The definition covers all these scenarios and more, so they are all still networks. Note that we will discuss many of these scenarios later in this course.

In almost all cases, there is one or more network operating systems making the network function. The network operating system runs on the devices connected to the network and handles the communication and data exchange. Microsoft Windows, UNIX, and Macintosh are examples of network operating systems because they contain software components that handle networking functions when connected to a network. Devices from the company Cisco use software that is also an example of a network operating system that runs on the devices that interconnect those endpoint computers.

Some networks have Thick Clients, whereas others have Thin Clients (and yes, some have a little of both). The difference between the two is actually simple. Thick client networks have PCs that run their own applications and often store their own data. Thin client networks have "dumb terminals" that must run their applications on distant servers connected to the network (but note that "distant" in this case could mean a foot or two away). Thin client systems rarely have data storage capability, relying instead on file servers. You can put file servers on thick client networks if desired. In that case, the thick client usually runs its own software but stores files on the file server. This is a common scenario.



X Area Networks

There are many types or categories of networks in use today with a variety of names. In general, the name of the category describes the geographical area it covers. The naming convention is X Area Network with the X describing the geographical area or, sometimes, the type.

Note: In reading through the descriptions of the various X Area Networks, keep in mind that none of this is written in stone. There are many variations on these basic themes.

A Local Area Network (LAN) typically covers a limited geographical area such as an office or building. A LAN can consist of one, two, or more smaller networks connected together to create the larger LAN. Doing this makes the network more efficient and easier to manage. When you do that, you call the smaller networks "subnets" (or subnetworks). Several subnets connect together to create a larger network called a LAN. We spend almost all our discussion in this course talking about LANs.

As the name implies, a Wide Area Network (WAN) covers a large geographical area such as a state or a country. Generally, a company uses a WAN to connect LANs at various geographical locations.

One way to tell the difference between a LAN and a WAN is by looking at the technology used to connect them together. (Though again, this is not a certain method.) If a network uses a technology such as Ethernet (discussed later), it is a LAN. If a network uses leased line technology such as T1s, T3s, or some sort of fiber backbone, it is most probably a WAN.

There are other X Area Network types that we mention here just for completeness. You may very well hear these terms being used in an IT shop.

We refer to a network that covers a city as a Metropolitan Area Network (MAN). Historically, MANs consisted of fiber-optic backbones run throughout the area. A business connected to the MAN via a T1, T3, or similar technology. From the MAN, the business not only gained access to other businesses connected to the MAN but also to the internet at large. Recently, some metropolitan areas have begun deploying Wireless MANs using wireless network technologies we will discuss later.

A Campus Area Network (CAN) does what the name implies: It connects all the buildings on a campus. The campus may be a university, a business park, a research complex, or a medical center, to name a few. Whatis.com defines a *campus* as "a physically contiguous association of locations such as several adjacent office buildings."[1] According to the same site, a *campus area network,* therefore, is "… a computer network that interconnects Local Area Networks (LANs) throughout a limited geographical area, such as a university campus or corporate campus."

A Personal Area Network (PAN) interconnects information technology devices within the range of an individual person. For example, a person traveling with a laptop, a smartphone, and a portable printer might interconnect them in a hotel room, creating a PAN. Business travelers have done this for years using small portable hubs. Today, Bluetooth and Wi-Fi wireless technologies commonly form Wireless Personal Area Networks, or WPANs. In general, the difference between a wireless LAN and a WPAN is that a LAN of any kind usually serves multiple users, whereas a PAN of any kind provides services for an individual.

Although most X Area Network category names identify the geographical area of the network, the term Storage Area Network (SAN) does not. A SAN is typically a small portion of a LAN (or WAN, or CAN, or whatever) that houses multiple storage devices such as high-end hard drives, optical drives, and backup tape systems. Because users on the rest of the network are accessing and storing data to and from a SAN on a continual basis, it is a *very* busy area of the network. Because of this, it is also usually the fastest, most robust area of the network.

Again, it is important to note that the above definitions and explanations are generalities at best. People throughout the network community are mixing and matching these terms at will. The best advice is when you are having a conversation with people and they start using one of these terms, make sure you know what definition they are using. Otherwise, it is extremely easy to become extremely confused in an extremely short amount of time.

Reference

[1] http://whatis.techtarget.com/definition/campus

https://www.internetlivestats.com/

The Internet

- 1969: Advanced Research Projects Agency (ARPA)—ARPANET
  - Intended for research and government use
  - Original design included survival of nuclear attack
- In 1995 *The Net* went commercial
  - Users grew from 45M in 1995 to 1B+ users in 2005
  - Now connects over 4 billion users
    - 48.4% in Asia, 21.8% in North and South America, 19% in Europe, 9.8% in Africa

"A worldwide system of computer networks (a network of networks) in which systems at any one location can potentially access information at any other location if they have permission to do so."

The Internet

Everyone knows what the internet is, but can you actually define it? It is not as simple as it sounds because *The Net* is so many things all at once. The best definition we could come up with is: "A worldwide system of computer networks (a network of networks) in which systems at any one location can potentially access information at any other location if they have permission to do so."

The internet began its life in 1969 as a project of the U.S. government's Advanced Research Projects Agency (ARPA). Its original name was ARPANET. The original intent of ARPANET was to interconnect universities and research organizations with the U.S. government and military. The idea was to facilitate communication between all the people doing research for the government (or people doing research that the government might want to use). The intent of ARPANET was to support government and (primarily) military research. Because the Cold War and its threat of nuclear attack were so prevalent in 1969, they designed the network from its inception to withstand major system outages such as those that would result from a nuclear attack. Today, the threat of nuclear attack has diminished, but the legacy of a survivable, self-healing network provides us with an extremely robust internet.

By the early 1990s, so many organizations had connected to what was now known as the internet and so much of the data was not research-related, it became clear that the internet was on its way to becoming a commercial venture. In 1995, the internet went commercial when purely commercial interests connected to the internet for the first time. In short order, the internet exploded into The Net that we know today.

By 1995, the internet had grown beyond anyone's imagination to 45 million users. Ten years later, in 2005, the internet user population was just more than 1 billion. Ten years after that, by the beginning of 2015, the internet had more than 3 billion users. Those who put together the original ARPANET truly had no idea where this would eventually go.[1]

The internet has become an integral part of life in America and other developed countries. Seemingly, every company and many individuals have their own website. It is hard to imagine watching television for any length of time, reading a newspaper or magazine, or even just listening to a conversation where something about the internet doesn't intrude into the dialogue.

Some very interesting statistics about the internet, its population, how much time spent on the internet by country (hint, the U.S. is nowhere near #1), etc. can be found at: https://wearesocial.com/blog/2018/01/global-digital-report-2018

Reference

[1] https://www.internetlivestats.com/

Map of Internet fiber backbone in the U.S. ➔

The Internet Versus TCP/IP Versus Web

- The terms internet and web are often used interchangeably
  - "I'm on the internet ..."
  - "I'm on the web ..."
- Internet = The highway
- TCP/IP = The truck (or transport)
- Web = Just one type of cargo
  - The transport does not care what the cargo happens to be

SANS SEC301 | Intro to Cyber Security 74

The Internet Versus TCP/IP Versus Web

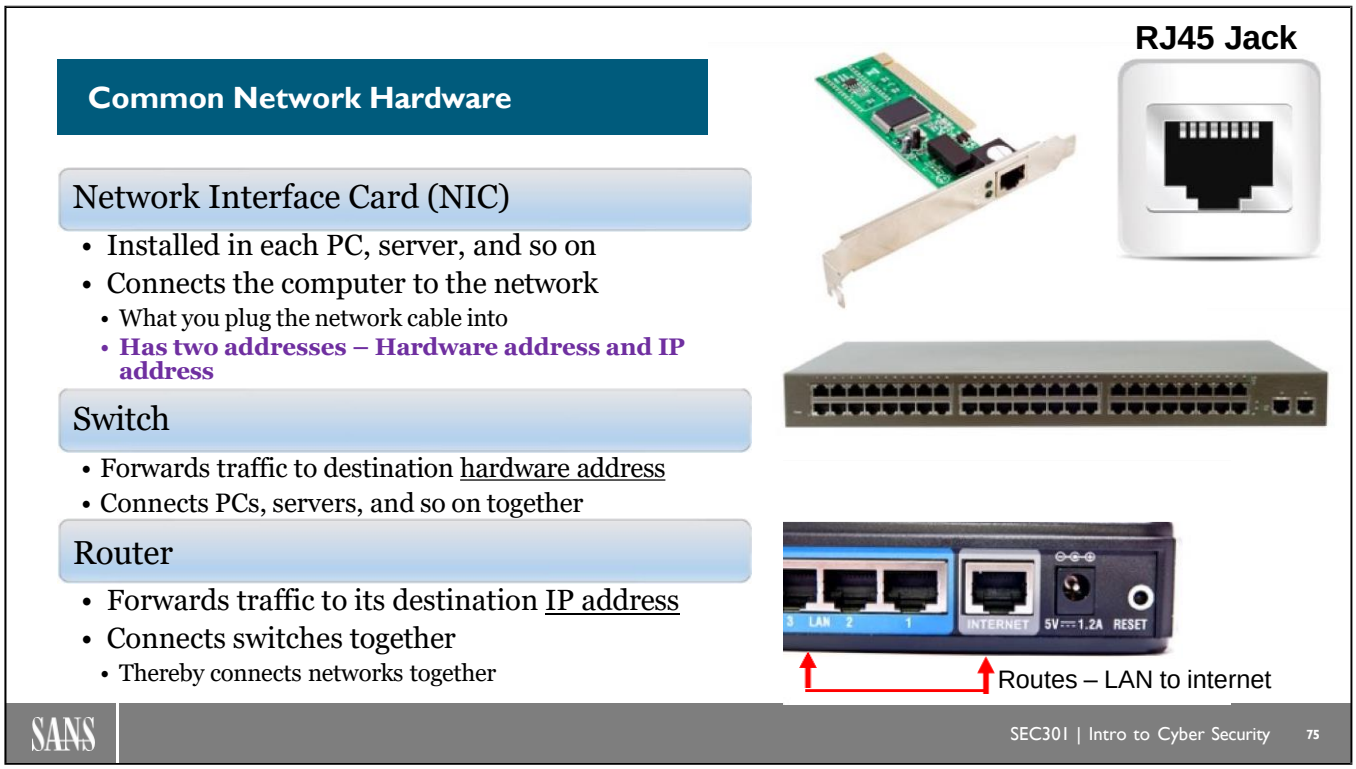
Many people interchange the terms internet and web: "On the internet" and "On the web" mean the same thing in conversation. In reality, the web is just one application of many used on the internet.

A good analogy is found in the picture above:

- The internet is the highway. It is an interconnected set of pathways over which vehicles can move. Similarly, the internet is an interconnected set of pathways over which we move data.

- On a highway, one of the ways that we move large amounts of cargo is by using a truck or transport. On the internet, the transport is better known as a family of protocols called TCP/IP (Transmission Control Protocol / Internet Protocol). Note that in networking, we would never use the word "truck" for this, but we do indeed call these "transport protocols".

- Just like the truck in the picture does not really care what kind of cargo it is hauling, TCP/IP does not care what the cargo is either. It is a set of protocols designed to move ones and zeros around from place to place to facilitate communication and exchange of information. Web is one type of cargo. Email, of course, is anther. And of course there are many others such as file transfer, voice over IP, video such as Netflix, and the list goes on. TCP/IP goes on delivering the ones and zeros that make all of that up.



Common Network Hardware

Every device connected to a network has a *Network Interface Card (NIC)* installed in it. This is what the network cable physically connects to on that device (or on a wireless network—it is what sends and receives the wireless signals that let you communicate on that wireless network). The NIC handles the transmission and receipt of data across the network. Each NIC has a unique hardware address called a Media Access Control address or MAC address (assuming it is an Ethernet network, which is our primary concentration here). As you will see in later discussions, other systems send data to the MAC address.

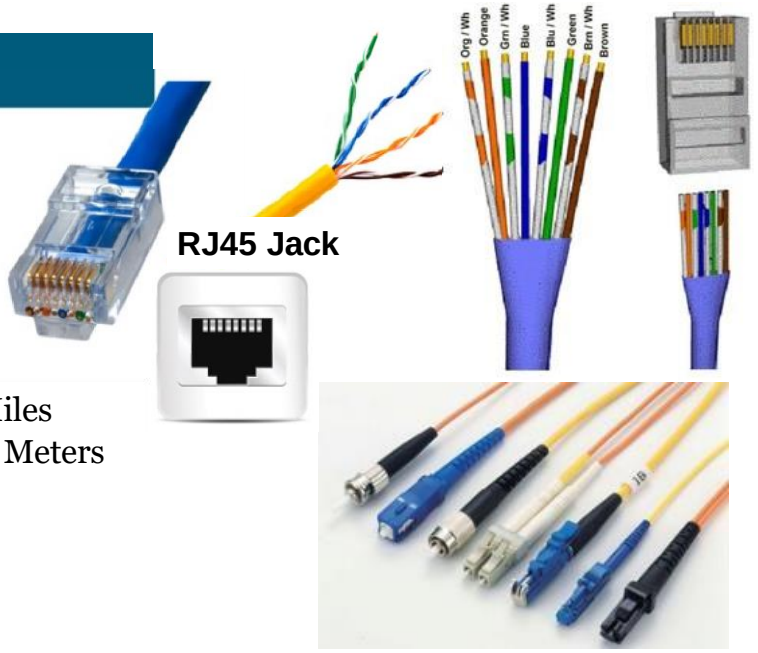
A *switch* is the central device of a modern star topology network. It is a device that directs traffic to the recipient based on the hardware (MAC) address. A switch is the central device in a star topology and sends the traffic only to the computer system where the recipient should reside. The switch maintains a list of the NIC MAC addresses for each node connected to it. When the switch sends data to a system, it sends it only to that one system.

Routers forward data from one network to another based on the destination IP address. Every router has at least two interfaces (NICs), each connected to a different network. Each piece of data traversing networks contains the IP address of the destination. A router maintains a routing table that says, to reach systems on network X, send the data out interface #1, but to reach systems on network Y, send the data out interface #2, and so on. In other words, a piece of network traffic enters the router on one of its interfaces; the router looks at the destination IP address, and the data leaves the router on another of its interfaces. It is, therefore, routing the traffic, hence the name router.

These three pieces of network hardware are found on virtually every network today.

Communications Media

- Copper cable:
  - Twisted Pair Cable
    Slang: Ethernet Cable
- Fiber-Optic cable:
  - Single mode: Laser—40 Miles
  - Multi-mode: LED—2,000 Meters
- Wireless: Covered later



Communications Mediums

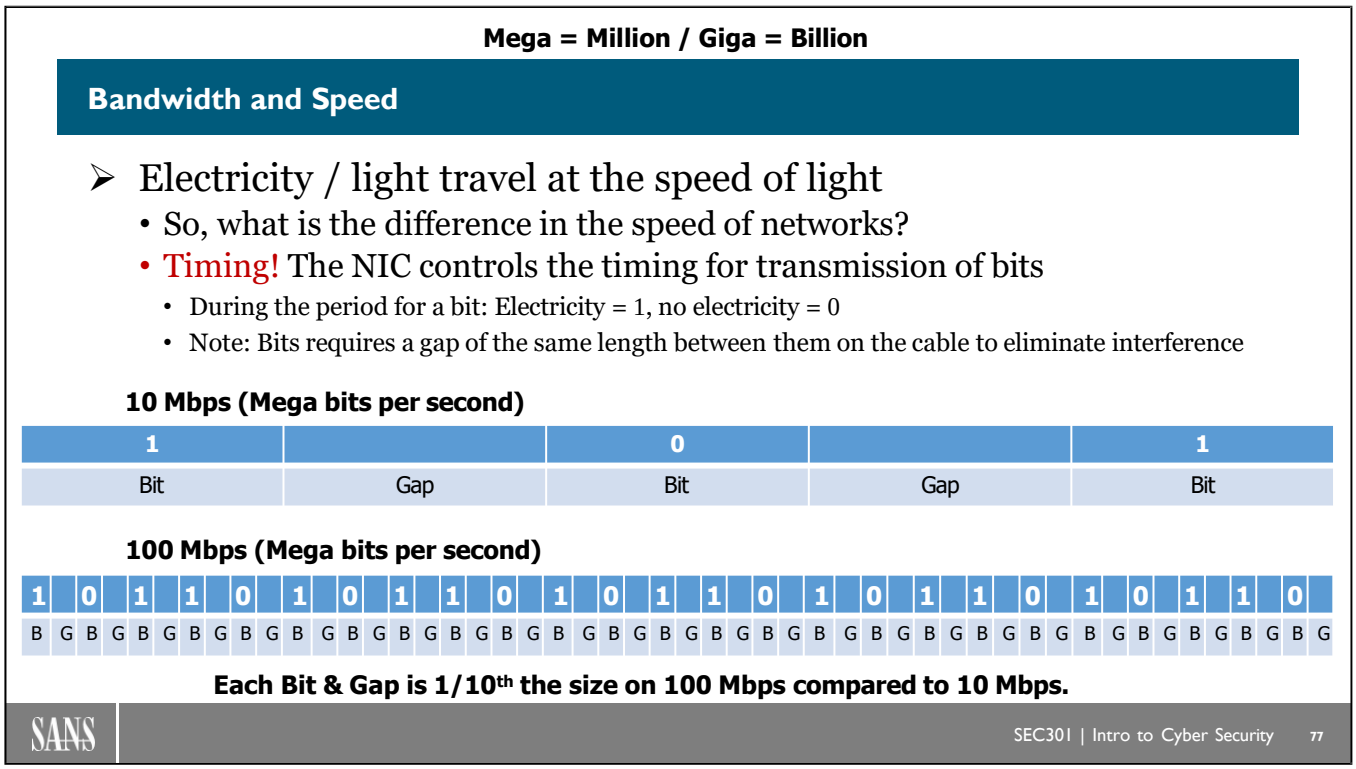
Twisted pair cable is what we use to build almost all LANs today. This type of cable often builds star topologies. The name does a good job of describing this type of cable. Typically, an outer insulation layer binds four pairs of copper wires twisted together. The more twists per foot, the better the cable's category rating because the better it protects against electromagnetic and radio frequency interference.

The absolute standard for connecting twisted pair cable to devices is the RJ45 plug and jack. This is always what is used.

A *fiber-optic cable* transmits information as light impulses along a glass or plastic wire or fiber. Fiber can carry more data, faster, and farther than copper cable. There are two types of fiber cable. Single-Mode fiber-optic cable utilizes laser transceivers to send data up to 40 miles without boosting the signal. Multimode fiber uses Light-Emitting Diode (LED) technology to transmit data up to 2,000 meters (1.24 miles).

As you can see above, there is no standard connector for fiber-optic cable. This is just one of many factors that lead to fiber being less common on corporate LANs.

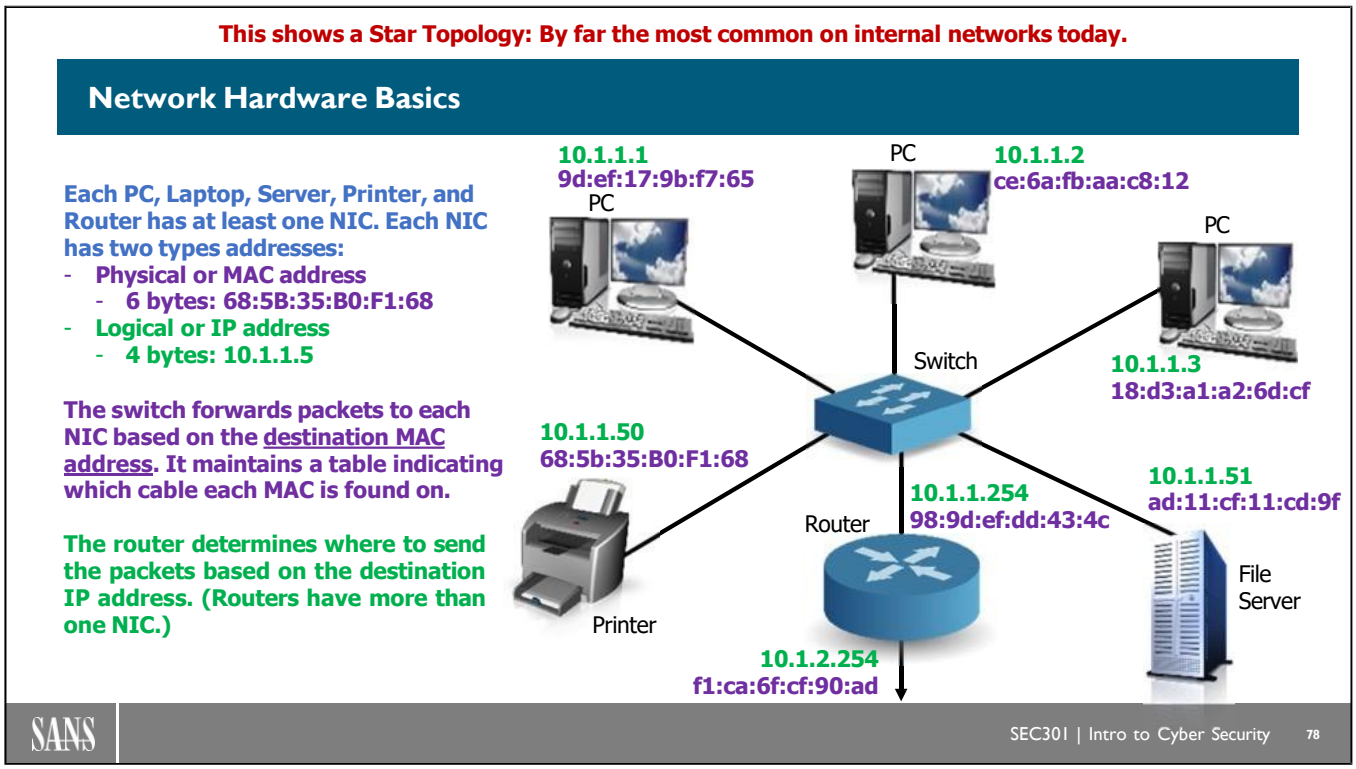
Wireless is also an option as the communications medium. These can overcome many of the problems that physical cable faces, such as getting the cable physically from point A to point B. However, you should also keep in mind that security is almost always a greater issue with wireless systems. There is an entire module devoted to wireless networking coming later in the course.



Bandwidth and Speed

Data moves across copper cable as an electrical impulse and physically travels at the speed of light regardless of the "speed" of the network. The difference is all in the timing. The NIC takes care of the timing: There is a period of time for a bit, followed by a period for a gap, followed by a period for a bit, and so on. Literally, if there is electricity during the period of time for a bit, that is a binary 1; if there is no electricity during that period, it is a binary zero. The difference between a 10Mbps (10 million bits per second) and a 100Mbps (100 million bits per second) is that on the faster network the period of time for the bit and for the gap are 1/10th the size of those periods on the slower network. So, you can physically fit more bits on the network in the same period of time.

The reason for the gap between bits is that the periods of time we are discussing here are extremely finite. On most networks today, they are measured in milliseconds (one-thousandth of a second) and on some networks, these times are measured in nanoseconds (one-billionth of a second). The gap between the bits keeps one bit from interfering with the next bit.



Network Hardware Basics

The word topology comes from the Greek *topos*, meaning place. In general usage, it is the description of the layout of any given locality. In networking, a *topology* is a drawing or schematic of how the network connects together.

There are several types of network topologies. A star topology occurs when all systems on the network connect to a central device, usually a switch. In this topology, you indirectly connect every node to every other node via that central device. Star topologies work well when nodes are at scattered points. In addition, it is easy to add nodes by plugging them into the central device. A cable failure in a star topology affects only the one workstation. The obvious point of failure is the central device. If it goes down, the network is down. *This is by far the most common LAN topology deployed today, and the one we concentrate on here*.

Network Interface Cards (NICs) have two addresses: A physical (MAC or Media Access Control) address and a logical (IP or Internet Protocol) address. The MAC address is 6 bytes and represented in Hex, with each byte separated by either a colon (:) or a dash (-). The IP address is 4 bytes and is represented in Dotted Decimal Notation, where each byte is separated by a dot.

The switch that all the devices of a star topology plug in to maintains a table with each device's MAC address and which port (or cable) that device is found on. The switch forwards packets to the correct device based on that MAC address.

Routers have at least two NICs. They maintain a routing table that indicates to the router which interface to forward packets out of based on the destination IP address.

This shows a Star Topology: By far the most common on internal networks today.

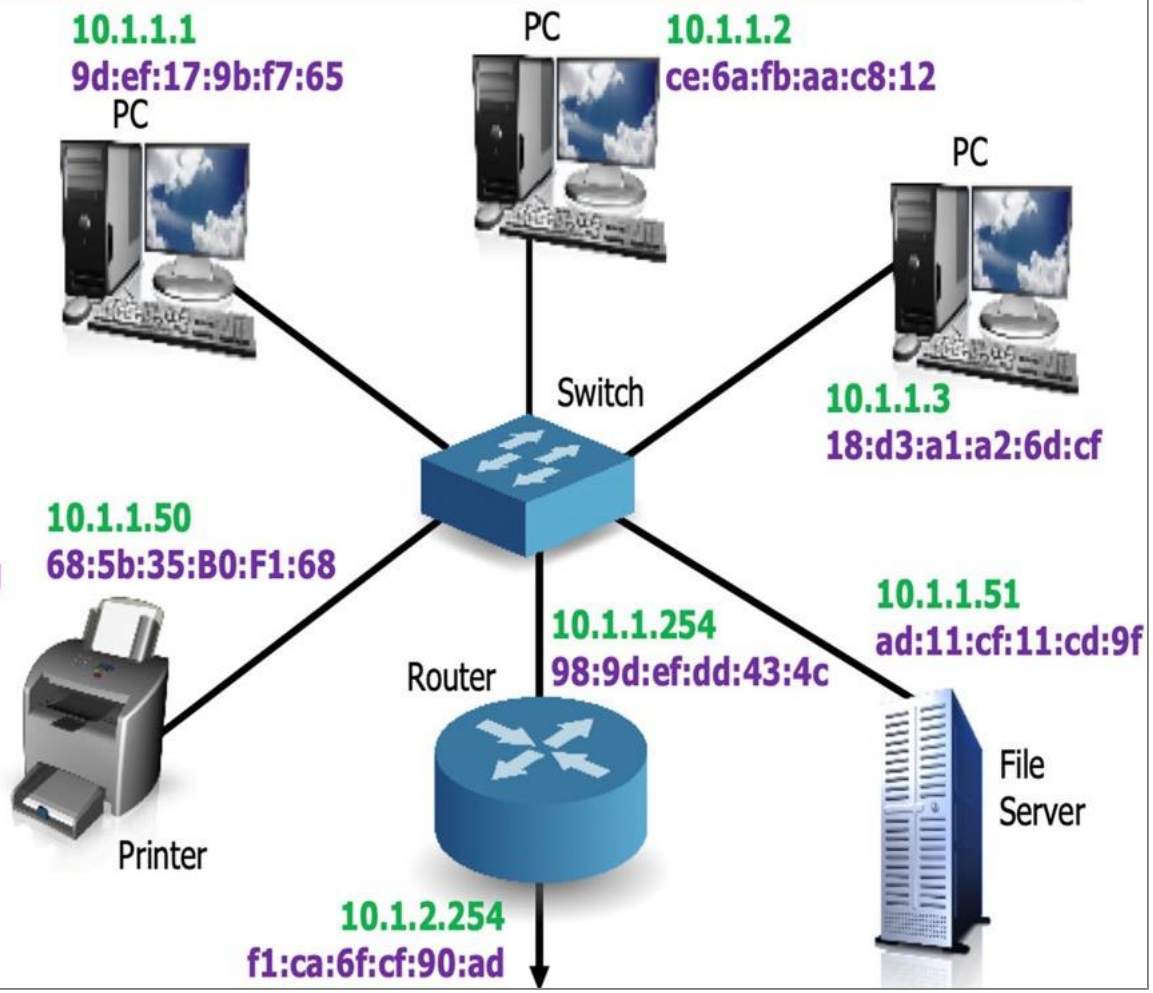
Network Hardware Basics

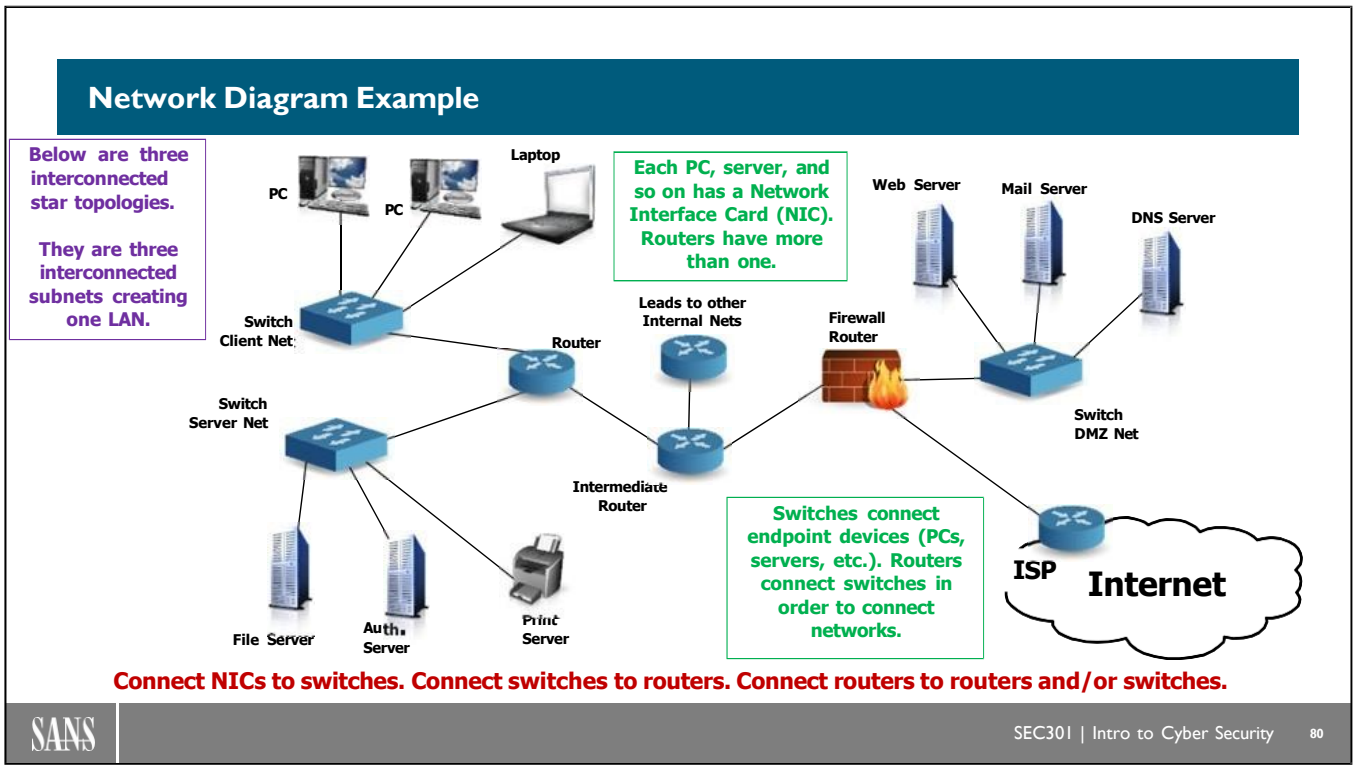
Each PC, Laptop, Server, Printer, and Router has at least one NIC. Each NIC has two types addresses:

- Physical or MAC address
  - 6 bytes: 68:5B:35:B0:F1:68
- Logical or IP address
  - 4 bytes: 10.1.1.5

The switch forwards packets to each NIC based on the destination MAC address. It maintains a table indicating which cable each MAC is found on.

The router determines where to send the packets based on the destination IP address. (Routers have more than one NIC.)





Network Diagram Example

Here, we see a simple network diagram. In the upper left corner, we see the client network containing devices that our users would utilize. In the lower left corner, we see the server network containing various types of servers that would be accessed by the clients. In the upper right corner is the Demilitarized Zone (DMZ), which is the public access area of our network. This is where we place servers that the public can access. In the bottom right corner is the connection to the ISP, who routes our traffic to the internet. And of course, in the middle, we see a firewall protecting the internal network (the client and server networks) from attacks coming from the internet. The firewall is allowing only appropriate traffic in to the DMZ, such as web traffic to the web server, for example.

Notice that the networks labeled Client Network, Server Network, and DMZ Network are three-star topology networks. You would commonly refer to each of them as a subnet. When you connect them together as you see above, you are creating the LAN for this organization. In this case, there are three subnets interconnected by the routers to create the organization's network or LAN.

The important points to remember are:

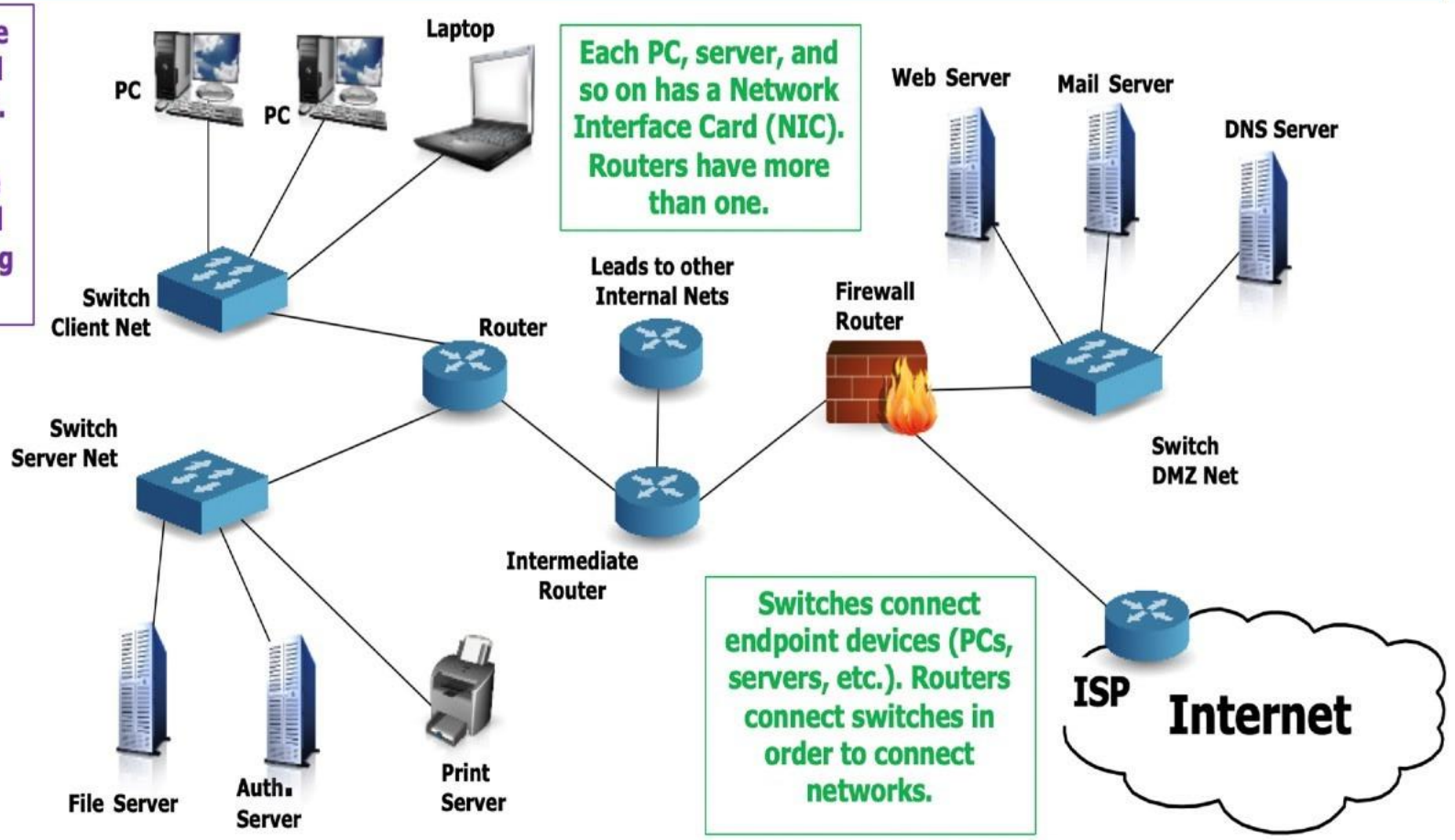
- Each PC, server, laptop, or other network connected device has a Network Interface Card (NIC) that connects it to the network.
- Routers have more than one NIC. The network traffic enters the router via one NIC and is routed out another NIC. (On routers, they are often simply referred to as interfaces.)

- Notice that both the device labeled Router and the device labeled Firewall have three interfaces each, indicating that firewalls must also be routers.
- We connect endpoint devices such as PCs, servers, and printers to switches.
- We connect switches to routers, which connect the networks (or subnets) together.

Network Diagram Example

Below are three interconnected star topologies.

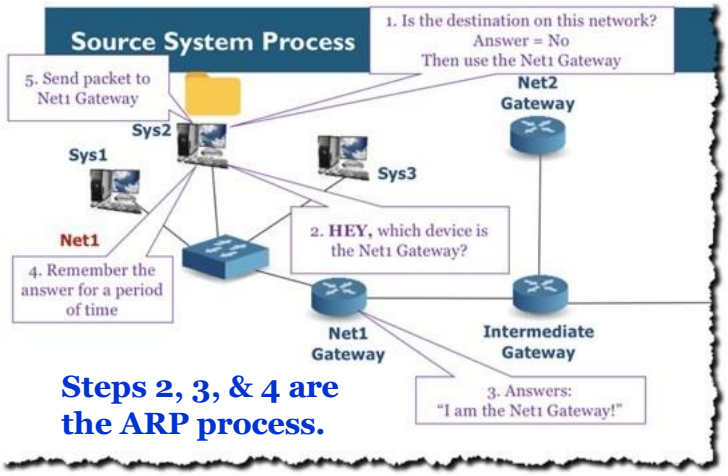
They are three interconnected subnets creating one LAN.



Connect NICs to switches. Connect switches to routers. Connect routers to routers and/or switches.

Address Resolution Protocol (ARP)

- In IPv4, there is no correlation between the MAC and the IP
- You need a protocol to resolve one to the other
  - Uses an IP address to obtain a MAC address



Data routes to the IP address, but physically transmits *only* to the MAC address.

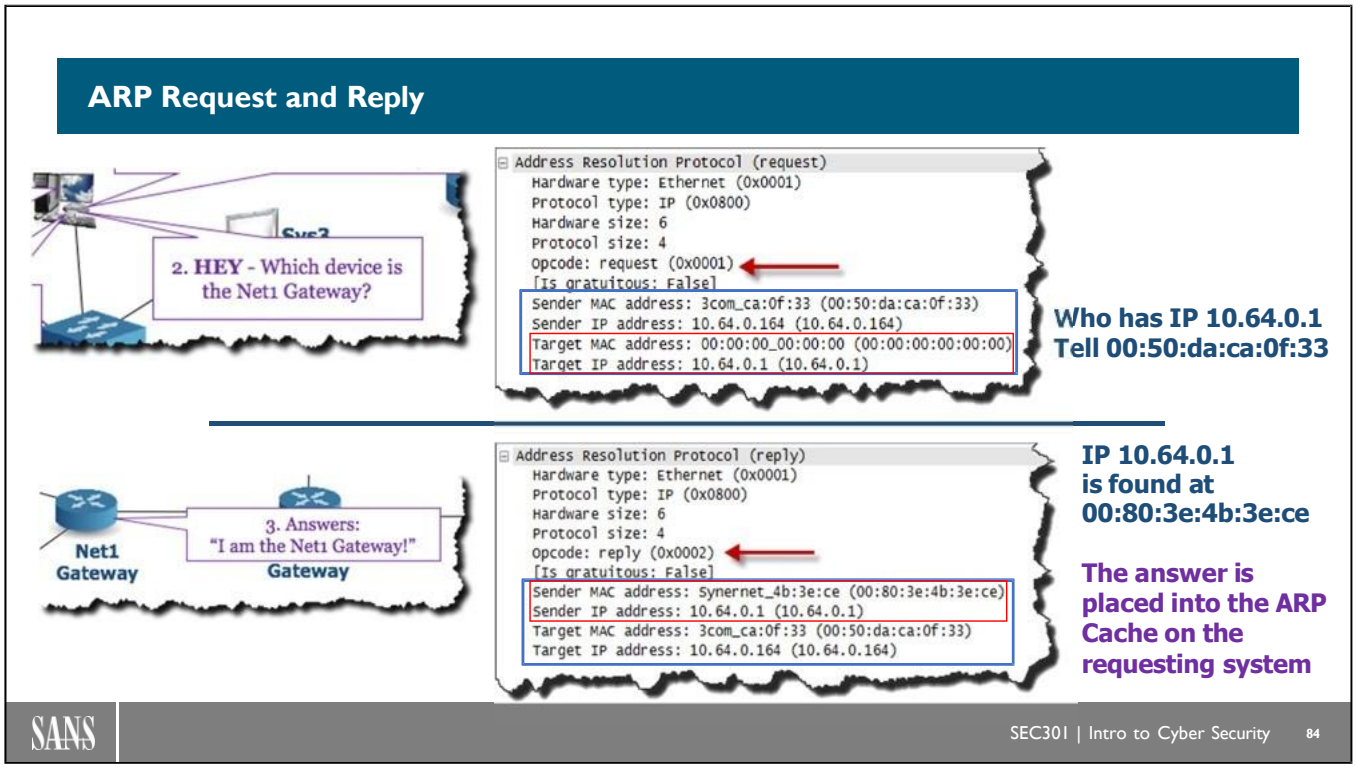
Address Resolution Protocol (ARP)

You have now seen two types of addressing: Hardware addresses (known as MAC addresses in Ethernet), and IP addresses. In IP version 4, there is no correlation between these two addresses. This means that if you have one of the addresses, you need a mechanism by which you can resolve the other (or determine what the other is). This is exactly what the Address Resolution Protocol (ARP) accomplishes.

Data routes to the IP address, but physically transmits *only* to the MAC address.

Let's say you have a packet destined to a web server on the internet. That means that the destination IP address in the packet is the IP address of that web server. In order for the packet to make it to the web server, it has to pass through the Net1 gateway (a router) in the diagram above. As noted above, the destination IP address is used to determine where to send the packet next. But to actually transmit it to the next device, you use the MAC address.

So above in step 2, the transmitting computer does not actually shout "HEY" on the network. In reality, it sends a broadcast that will be seen by all devices connected to Net1. The broadcast asks, "Who has the IP address assigned to the Net1 Gateway—tell the MAC address of Sys2." All devices on Net1 see this broadcast, but those that have a different IP address ignore it. Only the Net1 Gateway system replies back to Sys2 in step 3 saying, "Here is my MAC" address. In Step 4, Sys2 places this information into its ARP Cache (or sometimes called an ARP Table) that it maintains in RAM. In this way, Sys2 remembers the answer for a period of time and does not have to re-ask the question for a while.



ARP Request and Reply

Earlier, we saw Net1 Sys2 ask "Who is the Net1 Gateway?" and we saw the gateway answer that question. What did that actually look like?

We use the Wireshark network analysis tool to see the network traffic and fields of the packet. Here, you see that there are a few more fields contained in the actual ARP packet. We want to keep things simplified in our explanations here, so we are discussing only the most vital fields.

In the request, you can see

- Sender MAC: 00:50:da:ca:0f:33
- Sender IP: 10.64.0.164
- Target Mac: 00:00:00:00:00:00
- Target IP: 10.64.0.1

In the reply, you can see the sender and target are reversed and the sender MAC provides the answer we are looking for:

- Sender MAC: 00:80:3e:4b:3e:ce
- Sender IP: 10.64.0.1
- Target MAC: 00:50:da:ca:0f:33
- Target IP: 10.64.0.164

When a computer receives a reply to its ARP request, it places that information into its ARP cache. In computer lingo, a *cache* is a temporary storage location in memory. So, we are saying that the system that receives the ARP reply remembers that answer for a period of time. How long the answer is remembered depends on the operating system (and can be configured to a different value if need be), but suffice it to say, the computer does not need to ask for the distant system's hardware address again for the next several minutes.

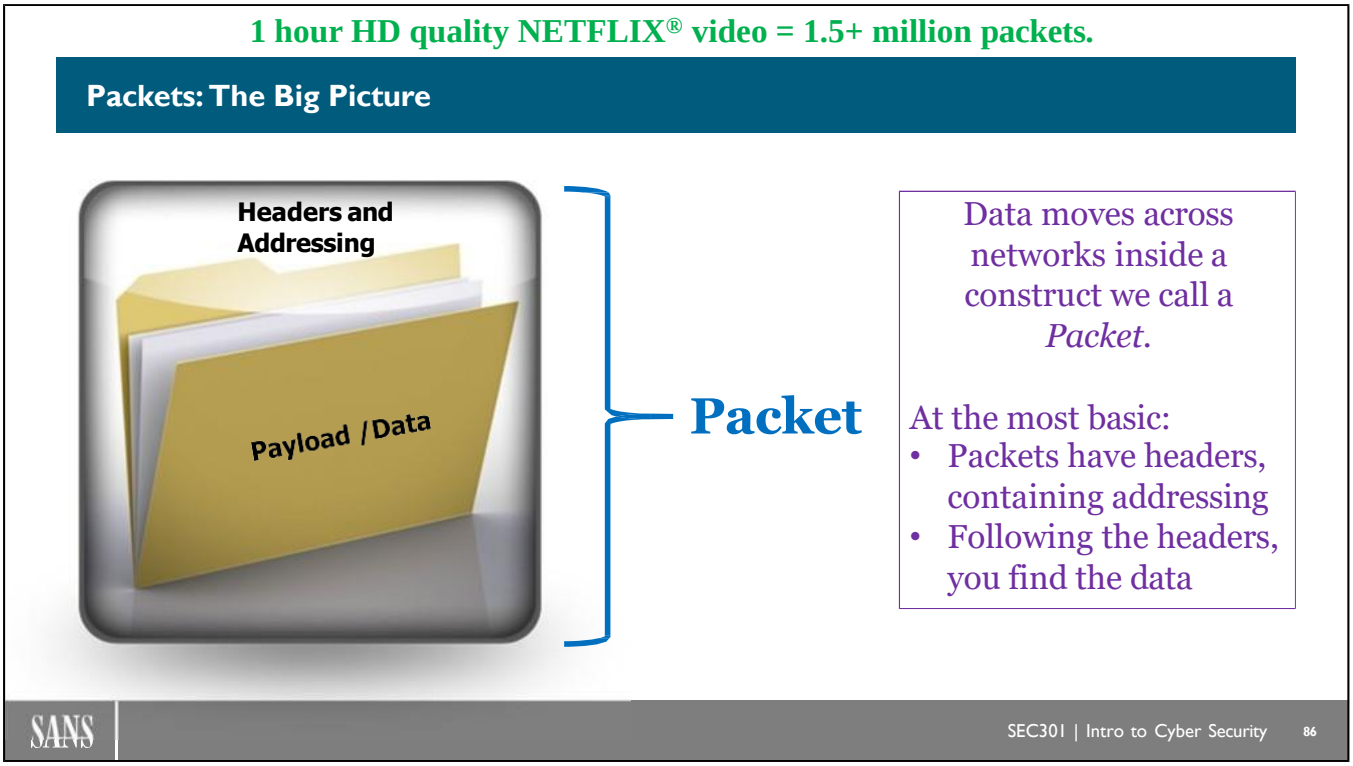
The command-line command **`arp -a`** shows you the contents of the ARP cache. The ARP cache shows the IP address and the hardware address. The command works identically on both Windows and Mac, but the output looks a little different. (Note that on Windows, if the request is sent but the response is not yet received, the ARP table shows the IP address, and all zeros for the hardware address, but you have to include the **v** (for verbose) switch on your command: **`arp -av`**)

Windows **`arp -a`** output:

```
C:\Users\Keith>arp -a

Interface: 172.16.148.137 --- 0xa
  Internet Address      Physical Address      Type
  172.16.148.2          00-50-56-e3-d9-7d     dynamic
  172.16.148.254        00-50-56-f2-24-13     dynamic
  172.16.148.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  226.178.217.5         01-00-5e-32-d9-05     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

```
keithpalmgren — -bash — 80×24
Last login: Thu Oct  5 13:57:01 on console
Keiths-iMac:~ keithpalmgren$ arp -a
? (10.37.129.255) at ff:ff:ff:ff:ff:ff on vnic1 ifscope [ethernet]
? (10.211.55.255) at ff:ff:ff:ff:ff:ff on vnic0 ifscope [ethernet]
? (169.254.46.201) at 84:89:ad:90:a4:2c on en1 [ethernet]
? (192.168.7.1) at 14:22:db:83:4a:ad on en1 ifscope [ethernet]
? (192.168.7.37) at ac:9b:a:24:bc:da on en1 ifscope [ethernet]
? (192.168.7.44) at 38:60:77:40:d8:ea on en1 ifscope [ethernet]
? (192.168.7.61) at (incomplete) on en1 ifscope [ethernet]
? (192.168.7.255) at ff:ff:ff:ff:ff:ff on en1 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en1 ifscope permanent [ethernet]
? (224.0.0.252) at 1:0:5e:0:0:fc on en1 ifscope permanent [ethernet]
? (224.224.224.224) at 1:0:5e:60:e0:e0 on en1 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en1 ifscope permanent [ethernet]
broadcasthost (255.255.255.255) at ff:ff:ff:ff:ff:ff on en1 ifscope [ethernet]
Keiths-iMac:~ keithpalmgren$
```



Packets: The Big Picture

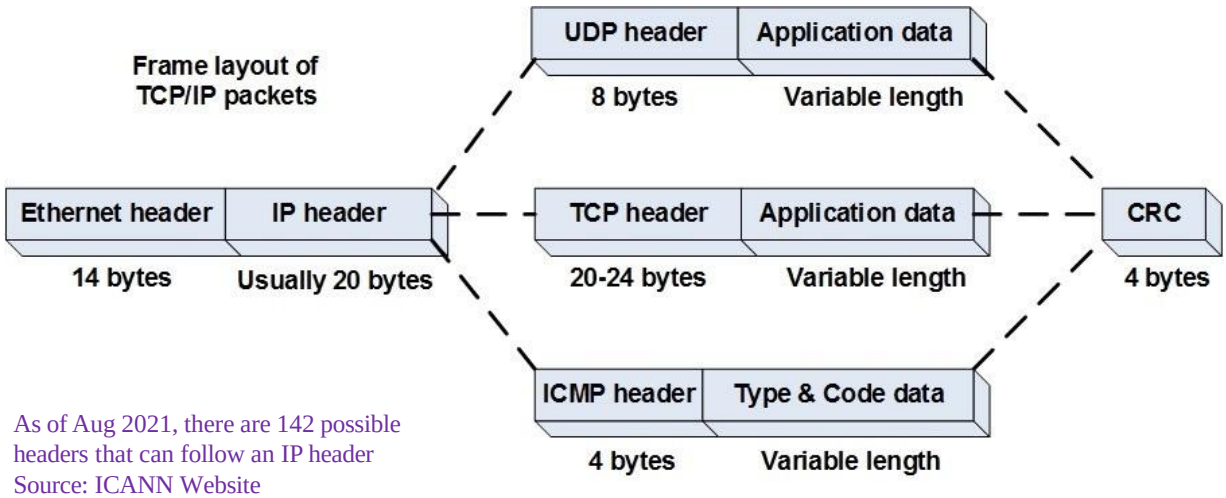
In the Postal Service analogy, we talked about a letter inside an envelope. In networking, we use the same sort of mechanism. However, instead of calling the containers of data an envelope, we refer to them as *packets* (or sometimes *data packets*).

A data packet has two basic parts: The *headers* and the *payload* (or *data)*. The header is the part of the packet that contains information about the packet: Where it came from, where it is going, and the kind of information it contains. The payload is the data or the information the packet is carrying. Note that in most cases, when you send something across a network, it does not travel in a single packet. Sending an email message, a file, or a web page could require many packets (literally up to thousands, tens of thousands, millions, and more).

The packet reveals many things about the information it contains. For example, by examining a packet header, you can determine:

- Where a packet came from (though this could be changed for a number of reasons)
- Where a packet is going
- The type of information the packet contains
- Whether a packet is the start of a new transmission or the continuation of an existing transmission
- Where in the information stream a packet belongs

Packet Anatomy



As of Aug 2021, there are 142 possible headers that can follow an IP header
Source: ICANN Website

Packet Anatomy

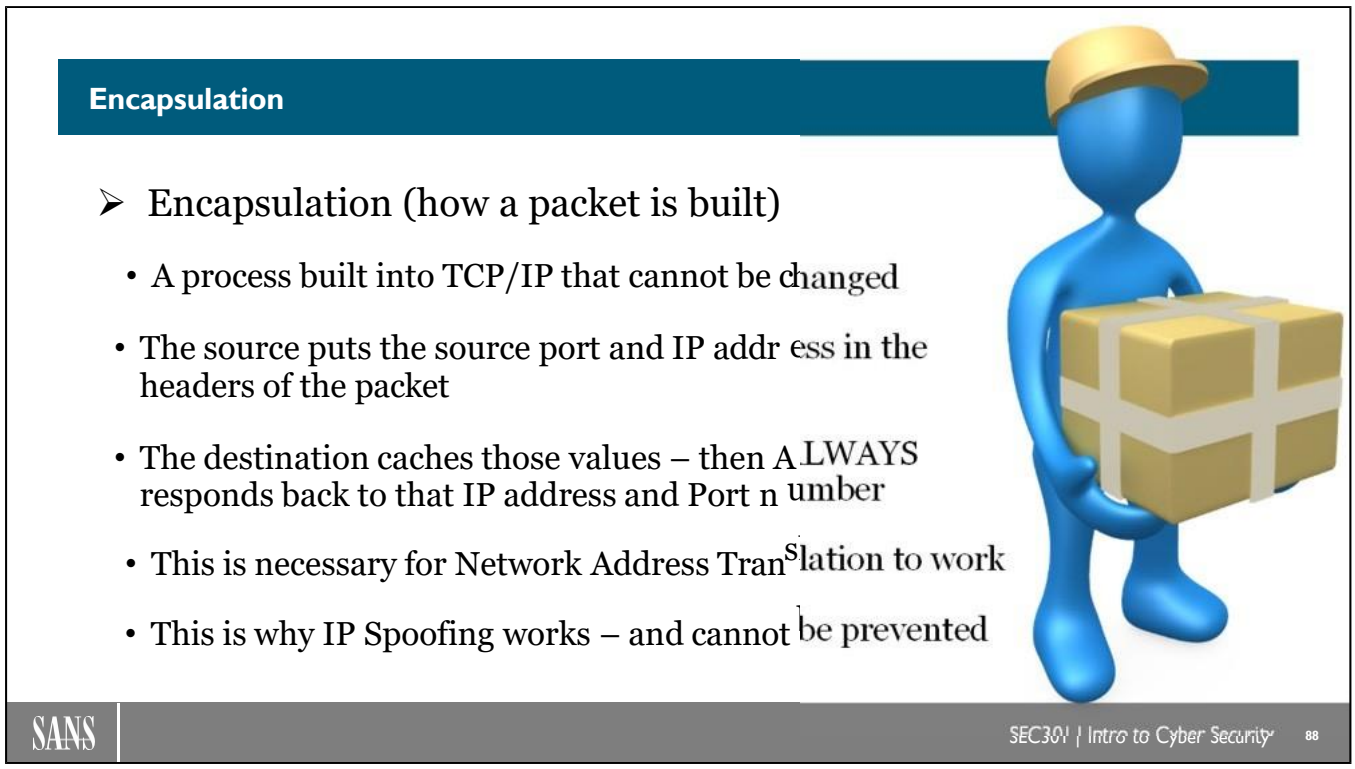
A TCP/IP packet consists of several parts. You see those parts listed in this diagram, but notice that for different types of packets, some of the parts change.

In short, each packet on an Ethernet network begins with an Ethernet header. That is followed by an IP header. At the end is a Cyclical Redundancy Check, or CRC (a number generated by a mathematical formula). The recipient of the packet runs the calculation again; if the calculation result equals the CRC, then the packet has not been garbled. Those three elements are static for all Ethernet packets.

Following the IP header, you typically see one of three types of headers depending on the type of packet. You may see a User Datagram Protocol (UDP) header, a Transmission Control Protocol (TCP) header, or an Internet Control Message Protocol (ICMP) header. We discuss those three protocols later. In all cases, following that header, you find the data.

To transmit this packet, the computer must first create this packet. That brings us to our discussion of encapsulation.

For a list of possible headers that can follow the IP header, see:
https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml



Encapsulation

One of the most important single concepts to understand about networking is that of *encapsulation*. This is the process within a computer that builds the packet for transmission across the network. Note that encapsulation is an integral part of how these protocols work. You cannot change or modify the functionality of encapsulation without breaking TCP/IP and in doing so, breaking the internet and virtually all other networks in existence today.

As the computer builds the packet, it places information in the headers. For example, let's say the computer is creating a TCP packet:

- The TCP header would contain the source and destination TCP ports
- The IP header would contain the source and destination IP addresses

When the next computer receives the packet, it takes it apart. As it removes the IP header, it caches the source IP address into a buffer. As it removes the TCP header, it caches the source port number into a buffer. When that computer builds a responding packet, it will always, and without exception, use what was the source port and IP address and the destination port and IP address.

In other words, responses to a packet ALWAYS return back to the IP and port they say they came from. This is necessary for Network Address Translation to work. This is also why IP spoofing works. An incoming packet can say it came from anyplace – and that is exactly where you will respond to, even if it is not the actual source.

SANS

Definition of Protocol
The OSI Reference Model (Stack)
The TCP/IP Reference Model (Stack)
Mapping OSI to TCP/IP

What Protocols Are

This page intentionally left blank.

Communication Protocols

- ***Rules governing communications on a network:***
  - How does a computing device find and understand the information it needs to act on?
- Defined in Request For Comment (RFC) documents
  - Defines format of headers, etc.
  - Allows computers to find what they are looking for
  - For example: Where is the destination IP address field?
    - RFC 791, page 10 spells out exactly were the destination IP is and how it is formatted every single time in every packet created by every computer in the world!

Communication Protocols

Networks are made up of various types of computing devices. Clearly, the computer on your desk is a computer, as is the server you connect to so you can retrieve a file or print a document. Less obvious, routers and switches are types of computers as well. Yes, they are special-purpose computers, but they are computers nonetheless.

When you hear a definition for the term protocol that says something like, "The rules governing communication on a network", it is a literal definition. When a computer creates a packet of data, there are rules the computer must follow when formatting that packet. For example, where do you place the destination IP address and how long is it? Every device in the world creates the packets the exact same way. Therefore, they all know how to find what they are looking for when they receive a packet of data. Again, if we all put the destination IP address in the same place, then we all know where to find that destination IP address. A simple fact about computers: They have no organic brain. Computers are incapable of making intuitive decisions. They can do what their software says they can do and nothing more. There can be no ambiguity in the instructions provided or in how data is formatted.

The rules for formatting communication packets are found in Request For Comment documents, or RFC's. These documents spell out every detail of how a protocol will work; how information is formatted, what it means, how it will be processed, etc. For example, the RFC for the IPv4 protocol is RFC 791. Page 10 of that document very specifically shows the destination IP address always begins at the 16th byte of an IP header in a packet and that the address will be exactly 32 bits [or 4 bytes]. Every computer in the world knows this and does it the same way.

From a security perspective, note that protocols dictate how to make networks work. They do not dictate how to make networks work securely. As long as both sides of the communication use the same protocol and use it properly, the communication will be successful. Unfortunately, simply using the protocol properly does not ensure that a transmission will be secure; in fact, many of the security problems we have today are a result of problems inherent with protocols. While some protocols, if properly implemented, do provide for security, many protocols do not. This is especially true of early protocols from the 1970's and 1980's. It is also true of some protocols being created today.

NOTE: Always check the publication date of RFC documents. If they are published on April 1st (April Fool's Day), they are a joke and should not be taken seriously.

Reference

https://en.wikipedia.org/wiki/Communication_protocol

https://en.wikipedia.org/wiki/April_Fools%27_Day_Request_for_Comments

Types of Protocols

- Network protocols:
  - Ethernet
  - Internet Protocol (IP)
- Application protocols:
  - Simple Mail Transfer Protocol (SMTP)
  - Hypertext Transfer Protocol (HTTP)
- Security protocols:
  - IPSec (for VPNs)
  - SSL and TLS (Secure Socket Layer and Transport Layer Security)

Types of Protocols

In the world of computers and networks, protocols handle everything from the moment your fingers type a character into the keyboard to the moment a computer on the other side of the planet receives your letter. The major types of protocols we use are as follows:

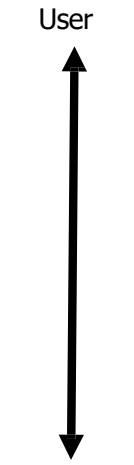
- **Networking protocols:** These are the protocols we use to move information around our networks. Network protocols govern how information is formatted on the network, how different network hardware and software deal with that information, and what to do if information does not fit the proscribed format. Network protocols generally deal with the movement of bits of information from one place to another. Examples of network protocols include the Internet Protocol (IP) for moving packets around a network and the Transmission Control Protocol (TCP) for tracking the order and delivery of packets.

- **Application protocols:** Application protocols deal with the movement of information between applications (for example, programs). They can also deal with the interface between human beings and the programs we use to store and process information. For example, the File Transport Protocol (FTP) defines how files can be moved between computers in an organized manner.

- **Security protocols:** These are the protocols that systems use to securely exchange information between them, ensuring the confidentiality and integrity of that information. Examples of security protocols are IPSec (for creating Virtual Private Networks [VPNs]) and applications such as HTTPS (HyperText Transfer Protocol Secure for secure web) and SSH (Secure Shell).

OSI layers 8 & 9 = Politics and Funding – A joke, but one you often hear referenced.

The OSI Reference Model (Stack)



7	**Application**	Manage Requests from Applications
6	**Presentation**	Code Conversion and Compression
5	**Session**	Establishes Sessions and Authentication
4	**Transport**	TCP and UDP Protocols
3	**Network**	Internet Protocol, Routing, ICMP
2	**Data Link**	Hardware Addressing
1	**Physical**	Cable/Media, Transmission, Timing

The OSI Reference Model (Stack)

Another common protocol *stack*, and one of the most widely known, is the International Organization for Standardization (ISO) Open Systems Interconnect (OSI) Reference Model (also commonly referred to as the OSI stack). The OSI model divides network communications into seven layers:

- **Physical layer:** This layer handles transmission across the physical media. This includes such things as electrical pulses on wires, connection specifications between hardware, voltage and current, and so on.

- **Data link layer:** This layer connects the physical part of the network (such as cables and electrical signals) with the abstract part (such as packets and data streams). It also creates the headers and validation information that are attached to packets.

- **Network layer:** This layer interacts with the network address scheme and connectivity over multiple network segments. It describes how systems on different network segments find and communicate with each other.

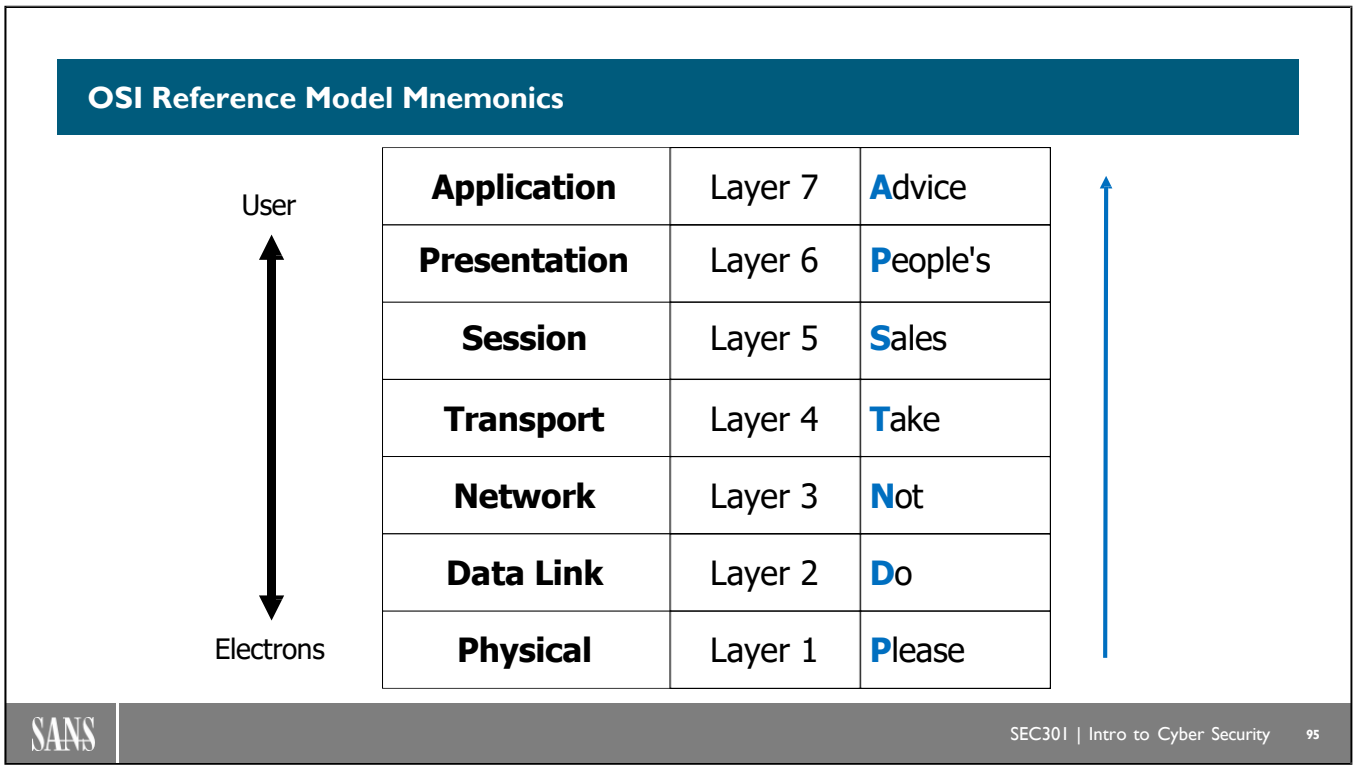
- **Transport layer:** This layer actually interacts with your information and prepares it for transmittal across the network. This is the layer that ensures reliable connectivity from end-to-end. The transport layer also handles the sequencing of packets in a transmission.

- **Session layer**: This layer establishes and maintains sessions between systems. If you want to think of a session as a connection, you are not far off the mark. This layer negotiates the session, sets it up, maintains it, and ensures that information exchanged across the connection is in sync on both sides. This layer is also where you find the authentication (login/logoff) functions.

- **Presentation layer:** This layer ensures that the data sent from one side of the connection is received in a format that is useful to the other side. For example, if the sender compresses the data prior to transmission, the presentation layer on the receiving end decompresses it before the receiver can use it. It prepares the compressed data for presentation to the application layer.

- **Application layer:** This layer interacts with the application to determine whether network services are required. When a program requires access to the network, the application layer manages requests from the program to the other layers down the stack.

In all protocol stacks, whether you are talking about OSI, as we are here, or TCP/IP, which we are about to see, the lower layers are closest to the wires. The upper layers are closest to the end user. The following screenshot shows the layers in the OSI stack, as well as some handy mnemonics to help you remember them.



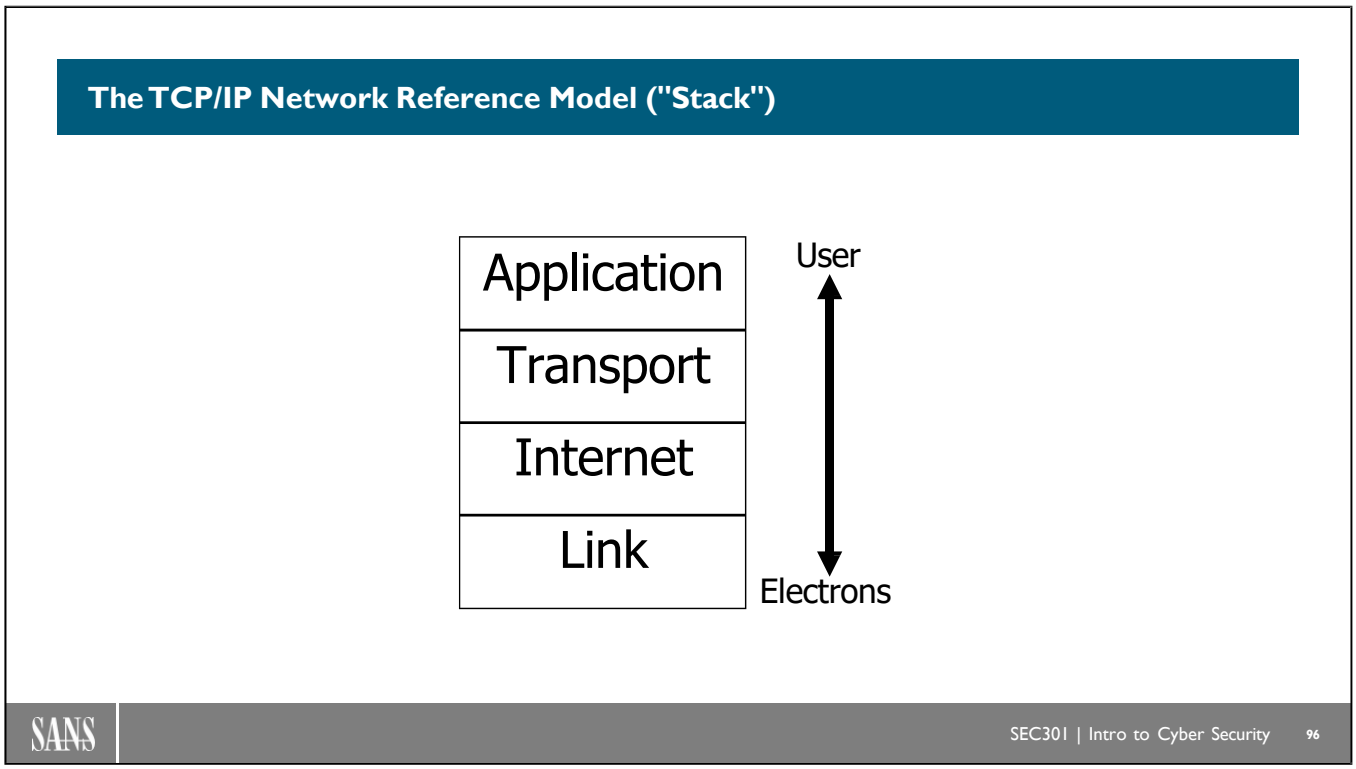
The OSI Reference Model Mnemonics

It is often necessary to remember the names of the layers of the OSI stack. Mnemonics can help greatly with this. There are MANY such mnemonics out there. If you already have one you have memorized, that is great. If you don't, then here is one you can use.

We prefer to remember the layers from the bottom (Layer 1) up, so "Please Do Not Take Sales People's Advice" works in that case. Because that tends to be good advice, we have used that one for many years.

Some people prefer to remember the layers from the top (Layer 7) down, so "A Person Should Try New Drinks Periodically" works as a mnemonic.

The thing about mnemonics—they have to be easy to remember in order to do you any good. Each person finds different mnemonics work better for them than others do. We have given you two here and hopefully one of them will work for you. If not, a quick Google search will find at least a dozen more.



The TCP/IP Network Reference Model ("Stack")
One of the earliest stack models was the *TCP/IP Network Reference Model*. The TCP/IP stack has four layers: The application layer, the transport layer, the internet layer, and the link layer.[1,2]

Note**:** Some reference books alternatively refer to the internet layer as the network layer.

Following are the definitions of the TCP/IP layers, from bottom to top:

- **Link layer:** This layer defines how to access a specific network topology; for example, Ethernet, Token Ring, and so on.

- **Internet layer:** Sometimes called the network layer, the internet layer defines how datagrams are formatted and handles the routing of data through the network. Examples of internet layer protocols include IP Version 4, IP Version 6, and the Internet Control Message Protocol (ICMP).

- **Transport layer:** This layer provides end-to-end data delivery service. This is the layer that assembles packets and sends them to the internet layer for processing. Examples of transport layer protocols include TCP and UDP.

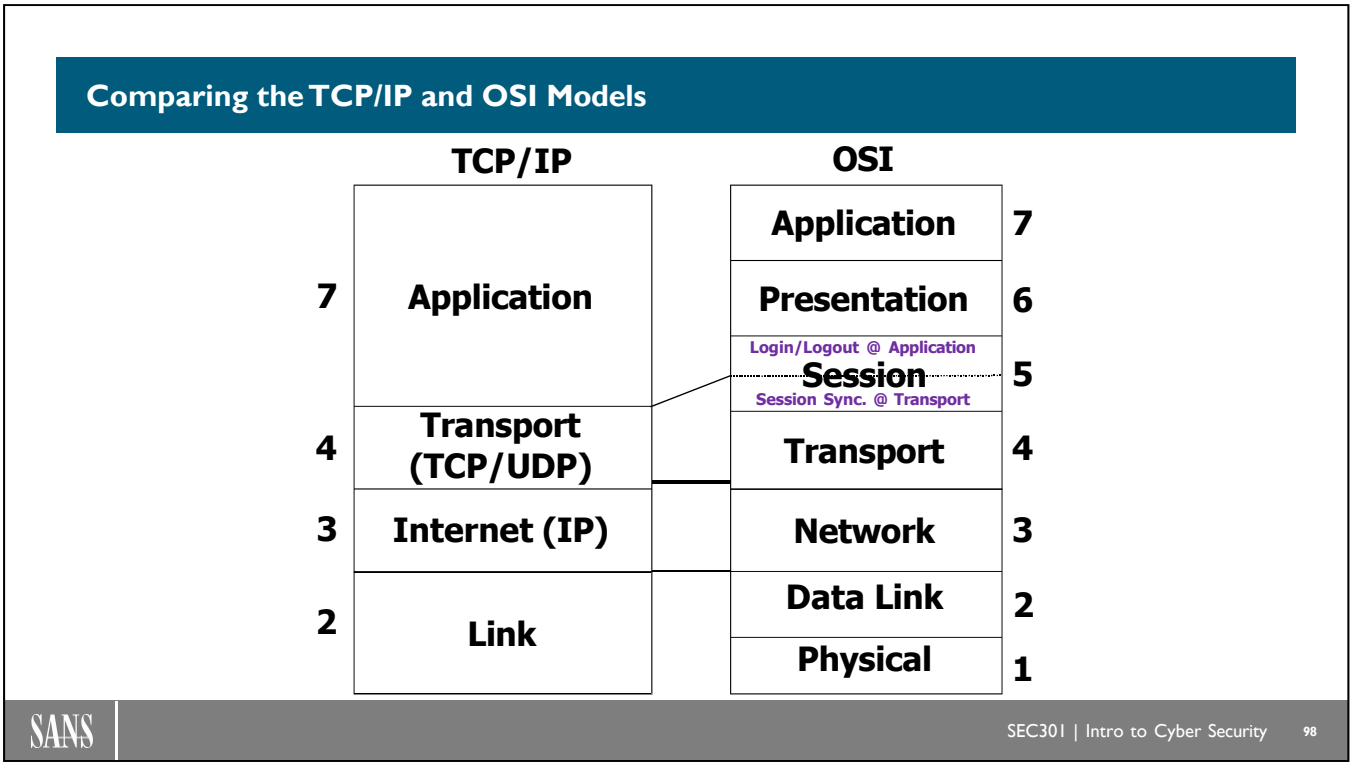
- **Application layer:** This layer consists of application programs and serves as the network interface into user applications. Examples of application layer services are Telnet, FTP,
and DNS.

With most stack models, including the TCP/IP model, the lower layers are the closest to the physical network (wires, electrons, and so on). The upper layers are closest to the ultimate *end user* of the data carried on the network.

References

1 *TCP/IP Illustrated*, Volume 1, *The Protocols*. W. Richard Stevens p. 1–5.

2 https://tools.ietf.org/html/rfc1122



Comparing the TCP/IP and OSI Models

No direct correlation between the OSI model and TCP/IP model is possible. Because one is a seven-layer model and one is a four-layer model, they obviously won't line up. To complicate matters further, some of the layers from each model don't always begin and end at the same place. A loose comparison is possible, however.

Note that while there are four layers in the TCP/IP model, the IT community numbers them in closer accordance with the OSI model. Specifically, they are numbered Layers 2, 3, 4, and 7 as you see in the slide above.

- The TCP/IP link Layer 2 matches the OSI Layers 1 and 2 in that these layers perform the same functions.
- TCP/IP Layer 3, the Internet or IP layer, matches directly to OSI Layer 3, or the network layer.
- TCP/IP Layer 4 matches all the functions of OSI Layer 4 and some of the functions of OSI Layer 5. Specifically, OSI Layer 5 handles session synchronization as well as items such as login and logout. In the TCP/IP model, Layer 4 (Transport) handles session synchronization, but not the login/logout services.
- TCP/IP Layer 7 handles the login/logout services of OSI's Layer 5, as well as all the functions of OSI Layers 6 and 7.

Module 7: Networking 102

- IP version 4
- The Default Gateway
- Routing Basics
- Internet Control Message Protocol (ICMP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Dynamic Host Configuration Protocol (DHCP)
- Domain Name System (DNS)
- Network Address Translation (NAT)

COURSE ROADMAP

- Module 5: How Computers Work
  - Lab 2.1: Converting Number Systems and Decoding ASCII
- Module 6: Networking 101
- **Module 7: Networking 102**
  - Lab 2.2: Networking

Module 7: Networking 102

This module continues our discussion of network communications by taking a closer look at what occurs at Layers 1 through 3 of the OSI stack. This module explains how computers communicate with each other using packets, how those packets are assembled, and how the packets are routed between computers (perhaps on different networks). We also look at the error notification mechanism that is called ICMP.

The Internet Protocol (IP)

- Provides IP addressing and packet routing:
  - Movement of packets from source to destination
  - Packet fragmentation and reassembly
- A *Best Effort* protocol
  - No guaranteed packet delivery
  - Has error *notification*
  - Not error *correction*

The Internet Protocol (IP)

Although a number of network protocols (for example, Internet Protocol (IP), Appletalk, Novell (IPX), and DECNET) are used today, we examine the Internet Protocol (IP) because it is the protocol used for communication on the internet. The Department of Defense developed IP as a way to link dissimilar computers across a network. From its humble beginnings, it has grown to be the "mother of all protocols" on the internet.

IP is designed to handle basic packet management on a network, including delivering packets to specific network addresses, routing packets between networks, and even splitting packets into smaller pieces, and then reassembling them again at their destination.

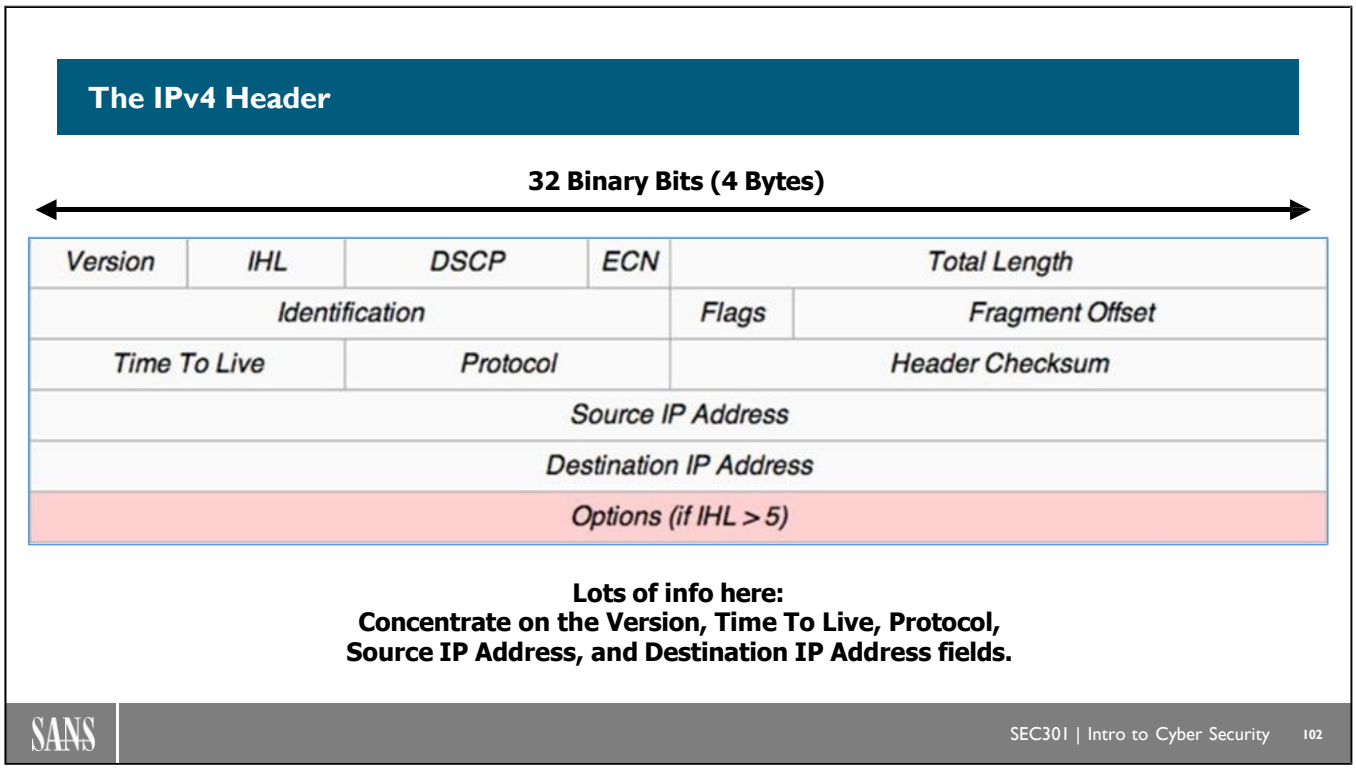
Because it focuses solely on packet delivery, IP lacks some of the features you might expect in a network protocol. For example, IP does not actually guarantee the delivery of a packet. The IP protocol lacks mechanisms that track packets to make sure they arrive at their destination safely. Also, IP does not guarantee the order of packet delivery; if three packets are sent to the same destination over an IP network, there is a chance that packet 3 might reach the destination before packet 1. You might think this makes IP unhelpful for "real" networking, but remember back to the discussion on how stacks work. Each piece of the stack does one task and leaves everything else for some other part of the stack. IP is no different; it concentrates solely on packet movement and lets other protocols worry about niceties such as delivery guarantees and packet order.

A packet, however, cannot tell you everything. For example, it cannot tell you:

- Who *really* sent the packet
- Whether the packet's contents have been altered during transmission

Because packets are the basic building block of network communications, you can tell that they will be important to our discussion of network security. Many attacks are the result of deliberate packet manipulation, selective packet sequencing, and packet misinterpretation. By understanding how packets work, and how the various fields in a packet can be used (and misused), you can begin to see the security implications of even the slightest variation in their correct usage.

In addition, many of the defensive mechanisms we commonly use, such as firewalls and intrusion detection/prevention devices, work by analyzing and reporting on packet anomalies. To properly understand the information that these mechanisms give us, we need to understand the data that is fed into them.



The IPv4 Header

This slide is the graphical representation of an IP header. Obviously, in reality, packets are not split up into nice, neat rows and columns like this; they are just streams of electrical energy representing ones and zeros. But the figure helps us to see how they are logically organized into groups of bits called *fields*, with each field having a particular meaning to the packet. Let's take a look at the overall structure of a packet and then look at some of the important fields individually.

Each row in the structure represents 32 bits (or 4 bytes) of data. Every 8 bits can be grouped into a *byte* or *octet*, and each byte contains two 4-bit chunks called *nibbles*. As we showed previously, a packet is divided into the header and the payload. The header in an IPv4 packet has a minimum size of 20 bytes, taking up five rows in the structure diagram, from the IP Version field to the Destination IP Address field.

Let's take a closer look at some of the header fields.

Version: The first nibble, or 4 bits, in the header is the IP Version field. This indicates the version of IP the packet is based on. The only legal values are 4 (for IPv4) or 6 (for IPv6). Any other value in this field indicates a malformed packet and should be dropped by your routers. When the version value is 4, the rest of the packet header is decoded (or the data is arranged), as shown in the graphic.

Time-To-Live: The *Time-To-Live* field, more commonly referred to as *TTL*, indicates the maximum number of transfers (or hops) from router to router that the packet can take on the network. Each time a packet gets to a new router, the TTL field is decremented by one. When the value in this field reaches zero, the packet is dropped, and a message is sent back to the originating computer indicating that the Time-to-Live was exceeded. The maximum allowable value for the TTL field is 255, meaning that a packet can take no more than 255 hops around the network before being dropped. For normal traffic, this is more than enough hops. IP limits the number of hops to prevent packets from wandering around the network indefinitely. This can be due to misconfigured routers or some packet routing anomaly.

Protocol: The *Protocol* field indicates the subprotocol that this packet is encapsulating. Recall that IP is concerned only with packet delivery. If an application needs to add intelligence to the packet delivery system, for example, packet order checking or reliability, it needs to use some other protocol on top of IP. The Protocol field indicates the protocol that this IP packet is encapsulating. Some common decimal values for the Protocol field follow:

1: Internet Control Message Protocol (ICMP)
6: Transmission Control Protocol (TCP)
17: User Datagram Protocol (UDP)
50: Encapsulating Security Payload (for IPSec)
51: Authentication Header (for IPSec)

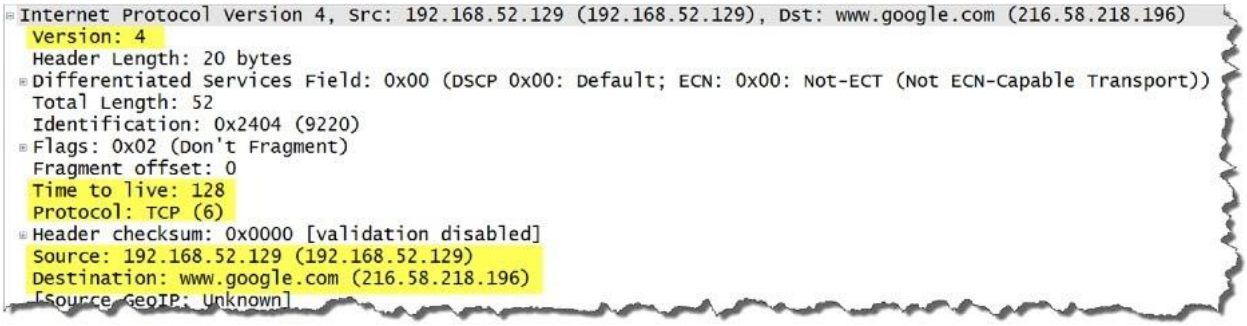
Source and Destination IP Address: The *Source Address* is the alleged address of the system where the packet originated (we say "alleged" because it is possible to spoof this address). The *Destination Address* is the location of the system where the packet is destined and is found in bytes 16 through 20.

Each of these addresses is a 32-bit number. If you do the math, that translates to almost 4.3 billion unique addresses (2^32 or 4,294,967,296 to be precise). This may seem like a lot of addresses, but in reality, it wasn't enough. The IPv4 address space was exhausted in February 2011.

Reference
http://www.ietf.org/rfc/rfc791.txt

IPv4 Header: Another View

- Screen shot from the Wireshark tool:
  - Utility that captures packets from the network
  - You can see the fields exactly as described



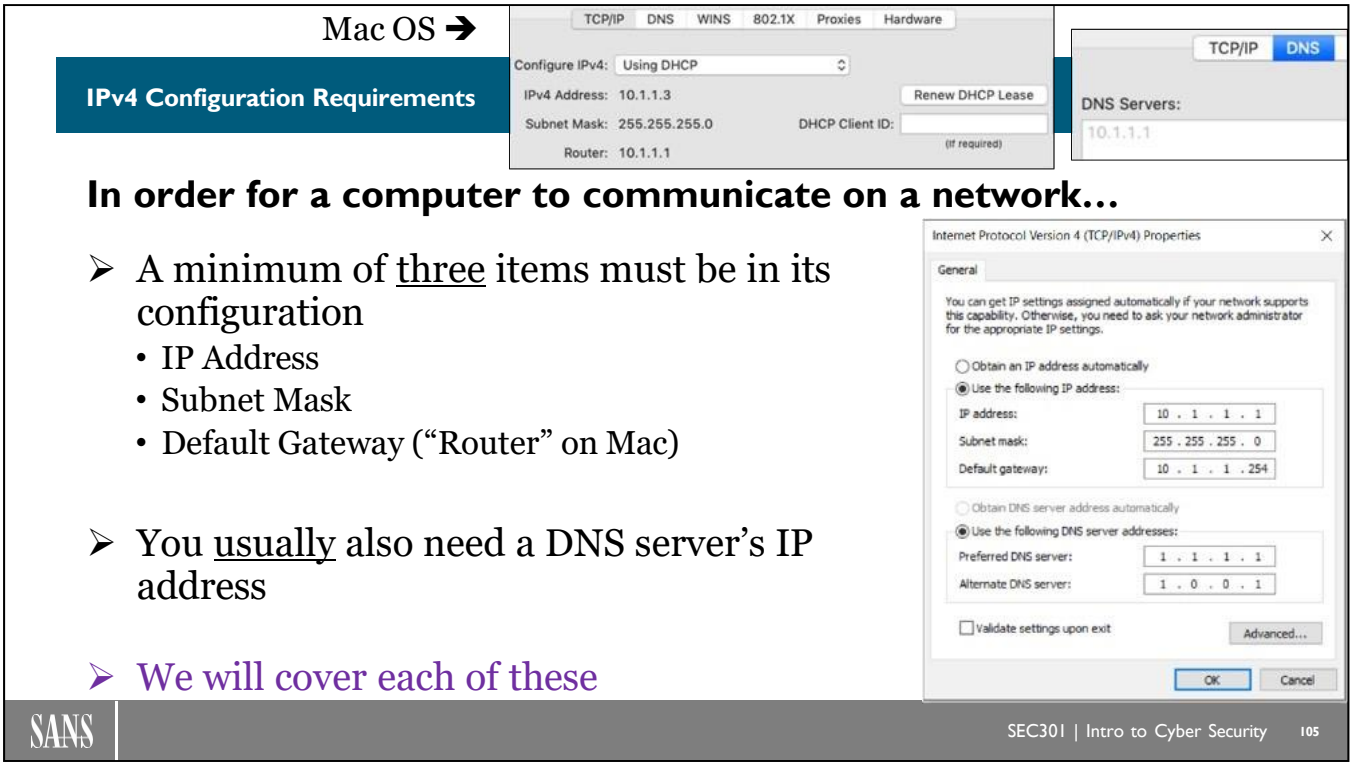
The fields above have been highlighted for clarity; the Wireshark tool does not highlight them as you see here.

IPv4 Header: Another View

This screen shot is from a network analysis tool called Wireshark. It can capture a copy of every packet that goes by on the network and show them to you for analysis.

Here you can see the Version, Time-To-Live, Protocol, and both IP address fields clearly.

- The version is set to 4 (meaning this is an IP version 4 header).
- The time to live is set to 128 (meaning this packet can go another 128 hops before being dropped).
- The protocol is set to 6 (meaning the next header in this packet will be a TCP header).
- The source address is 192.168.52.129 (listed twice because it is unresolvable via DNS).
- The destination address is 216.58.218.196 (which we see resolves to www.google.com).



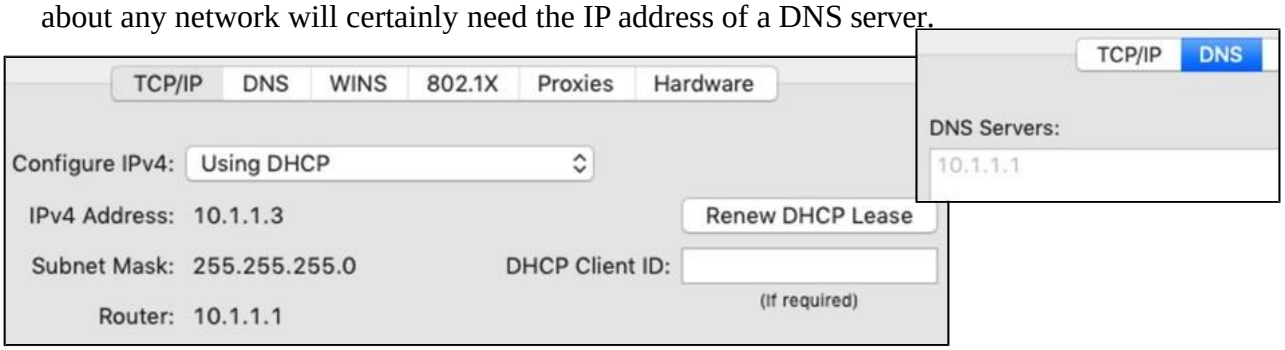
IPv4 Configuration Requirements

For a computer to communicate on a network, there is a minimum of three configuration settings required. Specifically, the configuration must include:

- IP Address
- Subnet Mask
- Default Gateway

So far, we have only discussed the IP address, and even then, just in a general sense. In the pages to come, we will cover each of these settings and why they are necessary.

In almost all cases, you also have to have a DNS server's IP address in the configuration if you want to get much done. That setting is not technically a "requirement" since on small networks, you would be able to work without a DNS server. In today's modern world, just about any network will certainly need the IP address of a DNS server.



IP Addressing (32-Bit)

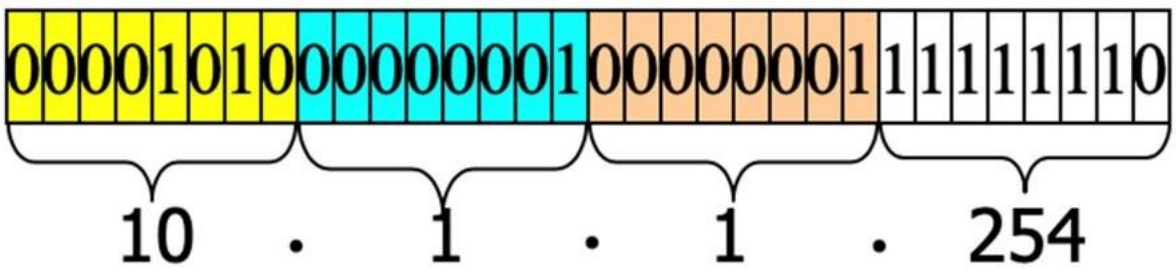
- IPv4 has a *32-bit* field for network addresses:
  - 32 bits are 4 bytes, also called 4 "octets"
- Uniquely identifies a computer on the network
- Ipv4 addresses written in "dotted decimal notation" for humans
  - Computers see everything in binary

IP Addressing (32-Bit)

Let's think back to the letter and post office analogy we used in our encapsulation discussion. We indicated that we needed to put a *To* address on the letter so that the post office would know where the letter was being sent. We included a return address (or *From*) address so the post office would know where to return the letter back to if there were a delivery problem. IP networks use the same concept. Each device on an IP network has an address or a method of uniquely identifying each device so that the protocols and network equipment can receive and deliver packets to the correct place.

Version 4 of the IP protocol has a 32-bit address field, meaning that all addresses are stored as 32-bit binary values. From earlier discussion, you know that 32-bits equals 4 bytes of 8 bits each. With IP addresses in particular, you will often hear the phrase “octet” – as in “an IP address is 4 octets” or “look at the third octet”. Please remember that the words ”byte” and “octet” are synonymous.

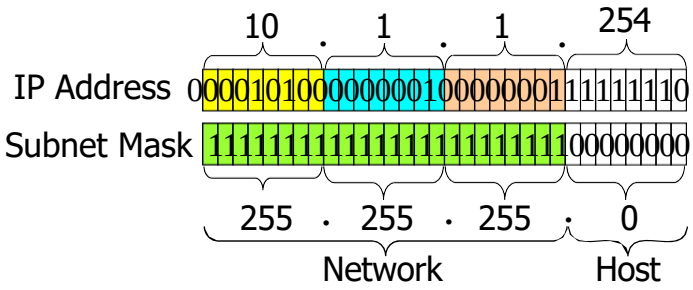
We represent IP addresses in “dotted decimal notation” of four decimal values separated by dots. Of course, computers see everything in binary 1’s and 0’s.



You can also use "slash notation" such as /24 – Meaning the first 24 bits are 1's

Introducing Subnet Masks

- Each IP address has two components:
  - Network Portion and Host Portion
    - Which network am I on? And, which computer I am on that network
  - Subnet Mask Defines the network and Host Portion
  - The mask here is 255.255.255.0



Network Address = 10.1.1.0

Given the mask of 255.255.255.0 & the first three octets of 10.1.1…

All computers with an IP beginning with 10.1.1 are on the same network.

The fourth octet tells which computer it is on that network.

Introducing Subnet Masks

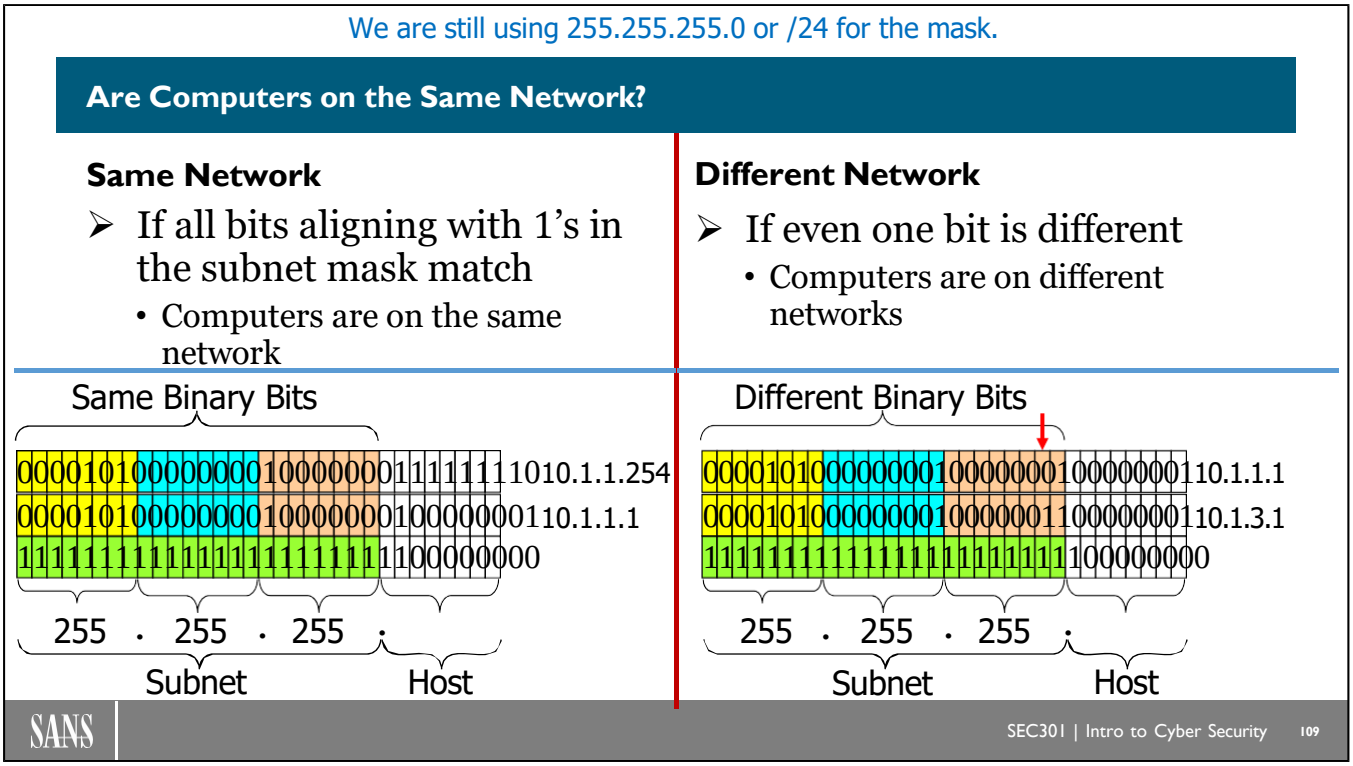
An IP address has two components: A network portion on the left, and a host portion on the right.

In other words, part of the IP address tells us which subnet our computer is on. Another portion of the address tells which computer (or host) on that subnet we are. We are saying "a portion" because the amount of the IP address that applies to the subnet and the hosts can change based on the subnet mask.

The subnet mask (also known as a *netmask* or even simply as the *mask*) identifies how many of the bits comprise the subnet portion of the address; the remaining bits comprise the host portion. For any bit in the IP address that the corresponding bit in the subnet mask is 1, the system includes that address bit as part of the network portion of the address. If the bit in the mask is a 0, that portion of the address denotes the host on that network.

In the slide, notice the subnet mask is 255.255.255.0. Because a byte with a decimal value of 255 equals eight 1's in binary, it means that the first three octets of the subnet mask are 1's. Therefore, the first three octets of the IP address denote which subnet the computer is on. The last octet denotes which host it is on that network. Because the IP address in this example begins with three octets of 10.1.1 and the mask begins with 255.255.255, this means any computers with an IP address beginning with 10.1.1 are all on the same subnet.

Historically, when configuring systems, we always entered the network mask using dotted decimal notation such as what you see above: 255.255.255.0. A newer convention is to enter the mask in "slash notation"—meaning you can express the mask by simply saying /24 in the example above. /24 means that the first 24 bits of the mask are 1's and any remaining bits are 0's.



Are Computers on the Same Network?

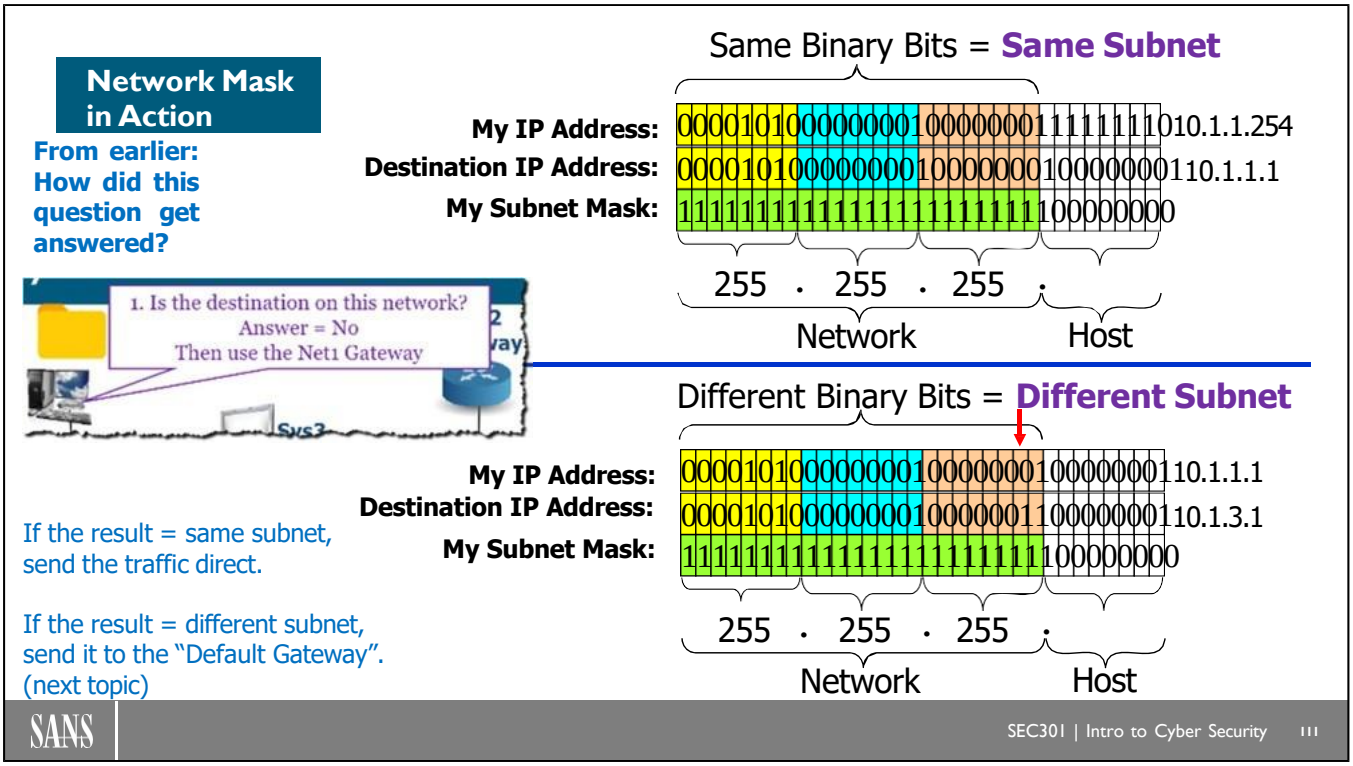
Your computer uses three values to determine if the destination computer is on the same network or a different one. Specifically, it looks at its own IP address, the destination address, and the subnet mask. If all bits in the source and destination address that correspond to 1's in the mask match, then the source and destination are on the same network. If even 1 binary bit is different, they are on different networks.

In both examples above, to determine if the source and destination are on the same network, we have to start by examining the mask. Notice, in both cases, we have a subnet mask of 255.255.255.0 or /24. Either way we write this, it means that the first 24 bits of the mask are all set to 1's. Therefore, we will compare the first 24 bits of the two IP addresses to determine if they are on the same network.

In the example on the left, notice all 24 bits are the same, so we know those two IP addresses are on the same network. In the example on the right, notice the binary bits are different. Specifically, the 23rd bit in the top line is 0, but it is a 1 in the bottom line (making the third octet a 1 above and a 3 below). That difference means the two IP addresses are on different networks.

We know this sounds simple and you are probably wondering why you don't just compare the dotted decimal numbers. In the example above you have different third octets, so they are different networks. In this example, it really is that easy. But know that not all subnet

masks are this easy to work with. What if you had a subnet mask of 255.255.252.0. That would give you the third octet in the mask of 11111100—meaning the mask does not always end at the dots. Yes, there are reasons this is necessary. This all means that the only way to make a certain determination of same subnet, or even a different one, is to put it in binary as we have here.



Network Mask In Action

In an earlier lecture, we showed Net1 System2 asking the question, "Is the destination on this net?" and the answer to the question was "No." But how could Net1 Sys2 answer that question?

It used the process described in the last few pages. Net1 Sys2 compared:

- The subnet mask in its own configuration
- Its own source IP address
- The destination computer's IP address

If all the bits of the two IP addresses that align with the 1's in the subnet mask match, the destination is on the same network, so send the traffic directly. If any binary bits in that section of the two IP addresses are different, then the destination is on a different subnet and the computer forwards the packet to the Default Gateway.

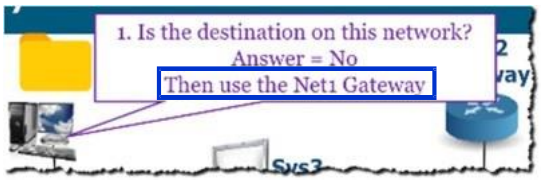
Wait—what is a "Default Gateway"? Simple—it is our next topic!

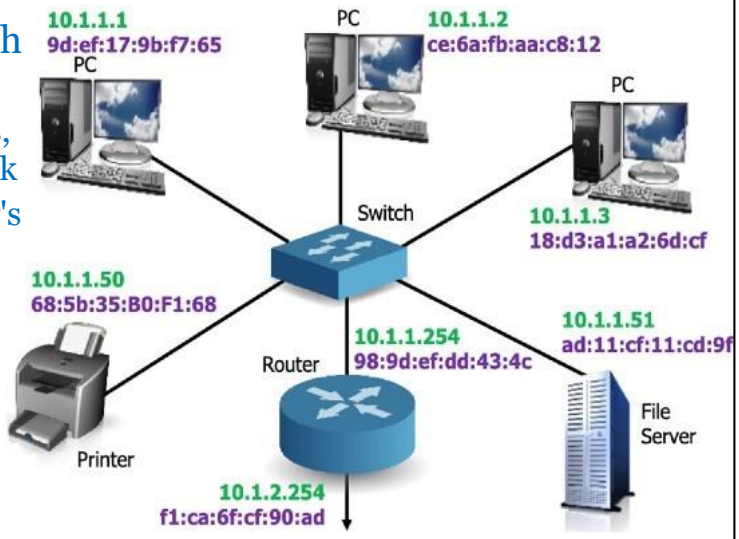
The Default Gateway (1)

Default Gateway = 10.1.1.254

The router that traffic goes through to enter or leave a network:

- Computer looks at its own address, destination address, & subnet mask
- Is the destination on the computer's own network?
- If "No", send the packet to the gateway





The Default Gateway (1)

Now that we know what happens when a computer decides that an address is not on its local network, the next question becomes, "What happens if the packet is going elsewhere?" That leads us to the discussion of the Default Gateway.

Every network has one entry and exit point: One place in which everything coming into the network or going out of the network must pass. This point is called a *router*. The devices inside the network see it as their Default Gateway.

To understand Default Gateway, you have to remember that each network node (PC, server, router, printer, and such) is a router. Your PC has the capability to route packets, meaning that there is something on your PC called a routing table. In that routing table, there must be at least one entry. That entry is for the Default Gateway.

This entry is also sometimes referred to as "the gateway of last resort" because it tells your PC, "If you don't know what else to do with a packet, send it here. Let that device figure it out."

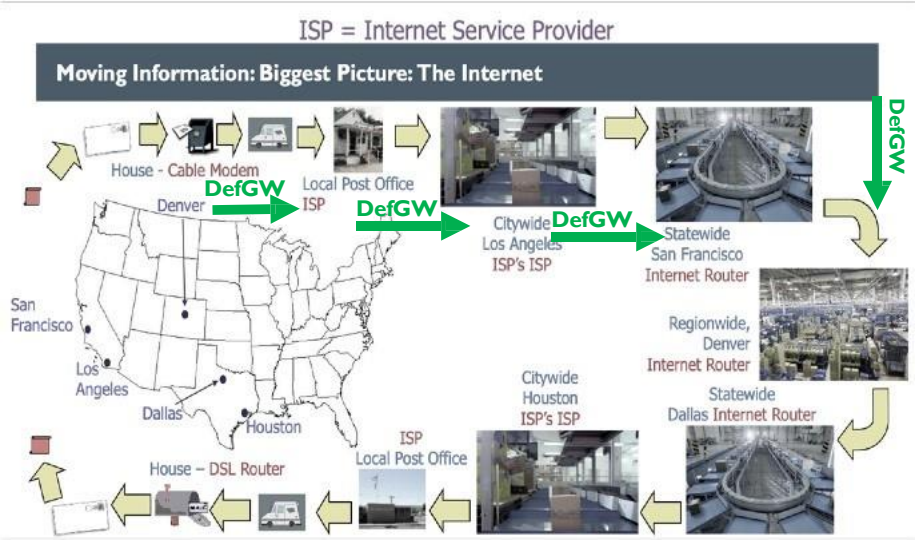
The Default Gateway (2)

PC's, Servers, Printers, etc. all have Default Gateways.

Routers also use Default Gateways. In the diagram from earlier, the Default Gateways are:

- **Cable modem = ISP**
- **ISP = ISP's ISP**
- **ISP's ISP = San Francisco router**
- **Dallas router = Denver router**

The Denver router sends the packet to Dallas based on its routing configuration. How that works is coming up.



The Default Gateway (2)

Let's clarify the Default Gateway concept a little bit. First, you should understand Default Gateways are used both on your internal network and on the internet. Every PC, server, printer, and any other device on your network uses a Default Gateway anytime that device needs to send data packets outside its own network. Routers also utilize Default Gateways any time they don't know what else to do with a packet.

This concept continues to the internet. When a router does not know what to do with a packet, it will always send the packet to that router's Default Gateway. To revisit the diagram from earlier, we see that:

- The device labeled Cable Modem has a Default Gateway of a router at the ISP
- The ISP's router has a Default Gateway of a router at the ISP's ISP in Los Angeles
- The ISP's ISP router has a Default Gateway of the San Francisco Internet Router
- The San Francisco Internet Router has a Default Gateway of the Denver Internet Router

In our example here, the Denver Internet Router would have some idea of how to forward the packet to its destination based on routing information in its configuration. How that works is our next topic.

The Default Gateway (3)

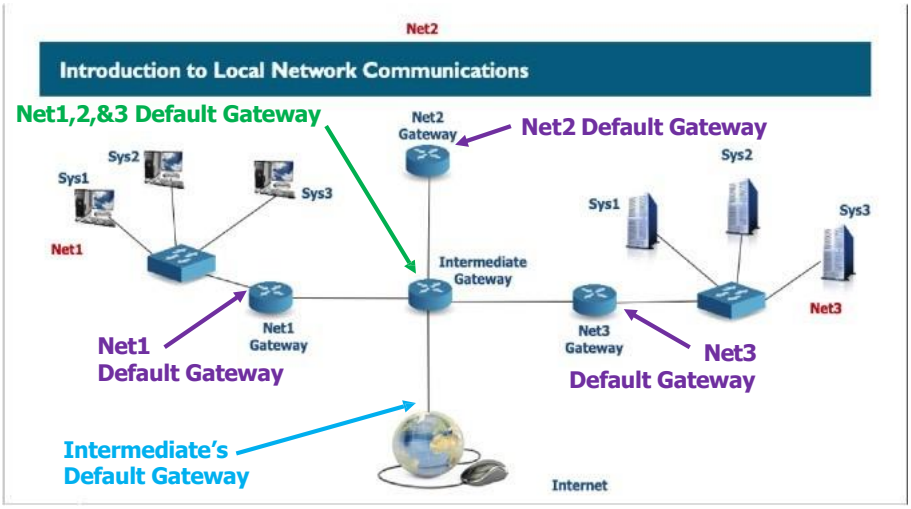
Revisiting another example from earlier – The network covering a three-story building

In this diagram, Default Gateways include:

- Net1 = Net1 Gateway
- Net2 = Net2 Gateway
- Net3 = Net3 Gateway
- Intermediate = ISP

Net1, Net2, & Net3 Gateways all use the Intermediate Gateway as their Default Gateway

The Intermediate uses the Internet Service Provider (ISP)



The Default Gateway (3)

We probably have the Default Gateway concept down pretty well by now. However, just for completeness of explanation, let's revisit one more earlier diagram; the one of the three-story building. You may recall there is one network on each of three stories of a building. Network 1, System 2 needs to send information to Network 3, System 2.

In this diagram, networks 1, 2, and 3 would each use their own network gateway (e.g. Net1 Default Gateway is the Net1 Gateway). Then, all three network gateways (Net1, Net2, and Net3 Gateways) would use the Intermediate Gateway as their default.

The Intermediate Gateway utilizes routing to determine where to send packets. Again, that is our next topic.

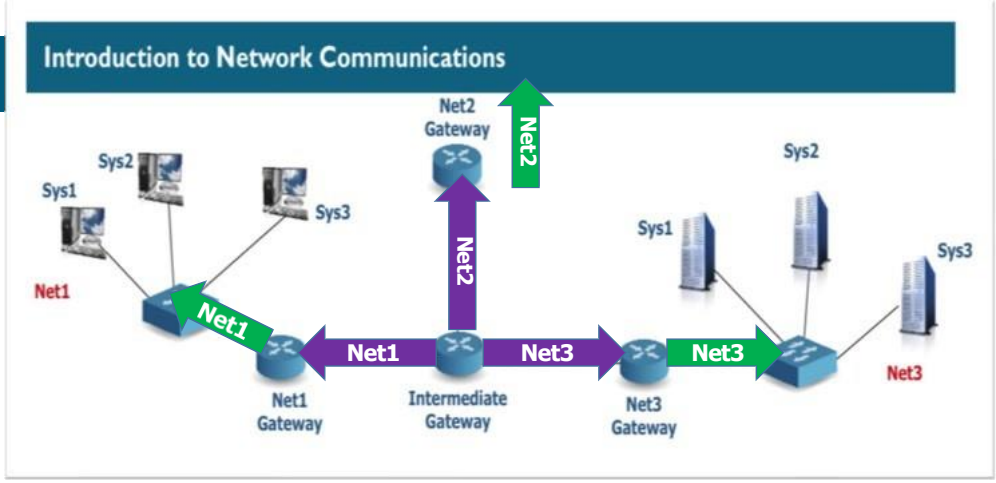
Routing Basics

Routing is a process used to determine where to send a packet next.

Each device looks at the destination IP address

If the device has no idea what to do with the packet, it sends it to the Default Gateway

If a "routing table" entry tells it how to send the packet closer to the destination, it does so



The routing table for the Intermediate Gateway says:

- If the destination is network 1, send it out the left interface
- If the destination is network 2, send it out the top interface
- If the destination is network 3, send it out the right interface

Note, each of the Network 1, 2, and 3 gateways know to send packets to the network they are directly connected to based on their routing tables.

Routing Basics

At this point in the discussion of networking, we understand that when a device on the network does not know what to do with a packet, it sends it to its Default Gateway. But at some point in the process, systems do have to know what to do with a packet. That is when we get into the next aspect of the routing process.

Each device on the network has a "routing table" as part of its configuration. At the most basic, the routing table says "if you are trying to get to network X, send the packet out interface number 1. But if you are trying to get to network Y, send the packet out interface number 2, etc." This was actually explained on the "Common Network Hardware" slide earlier in this book.

In the slide above, the Intermediate gateway has a routing table that says, if the packet is going to network 1, send it out the left interface. If the packet is going to network 2, send it out the top interface. If the traffic is going to network 3, send it out the right interface. Do note that router interfaces are not typically labeled left, top, and right. Those interfaces are numbered, so the intermediate router might have interface 0 pointing left, interface 1 pointing up, and interface 2 pointing right.

Of course, the routers labeled Net1 Gateway, Net2 Gateway, and Net3 Gateway each have a routing table. Those routing tables would indicate that if the packet is destined for the routers own network, route it out the interface pointing toward that network. If the packet is going anywhere else, send it to the Default Gateway, which is the Intermediate router.

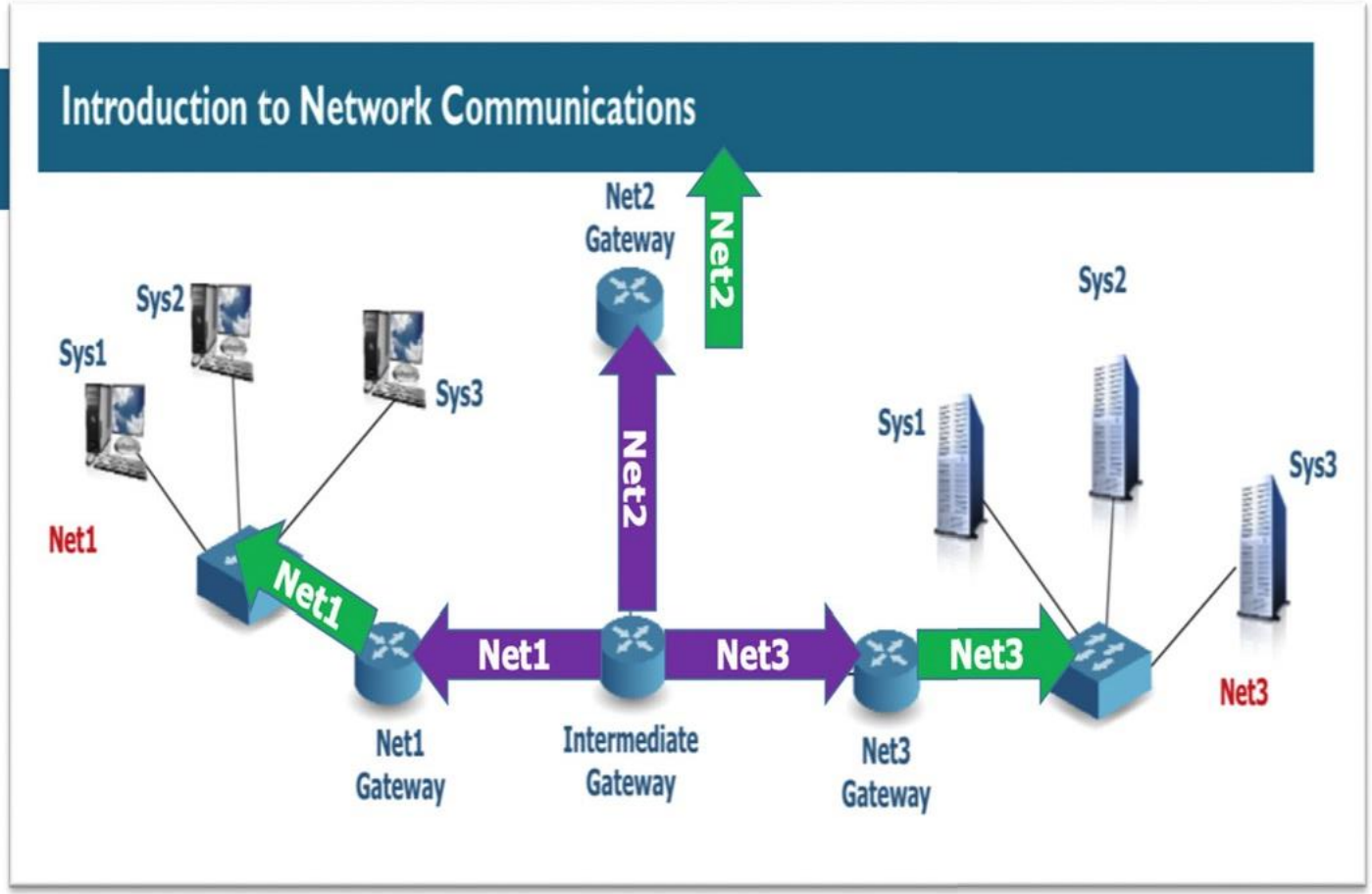
Routing Basics

Routing is a process used to determine where to send a packet next.

Each device looks at the destination IP address

If the device has no idea what to do with the packet, it sends it to the Default Gateway

If a "routing table" entry tells it how to send the packet closer to the destination, it does so



The routing table for the Intermediate Gateway says:

- If the destination is network 1, send it out the left interface
- If the destination is network 2, send it out the top interface
- If the destination is network 3, send it out the right interface

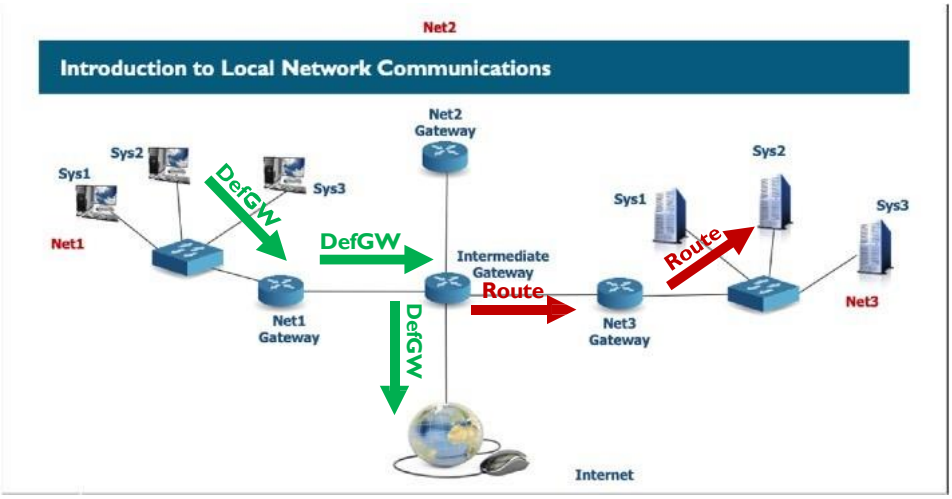
Note, each of the Network 1, 2, and 3 gateways know to send packets to the network they are directly connected to based on their routing tables.

Default Gateways & Routing

Net1 Sys2 to Net3 Sys2:

- Net1 Sys2 sends to Default Gateway (DefGW) of Net1 Gateway
- Net1 Gateway sends to DefGW of Intermediate Gateway
- Intermediate routes to Net3 Gateway
- Net3 Gateway routes to Net2 Sys2

If traffic is going to the internet instead, the Intermediate would send the traffic to its Default Gateway of the ISP



Default Gateways & Routing

To tie together the information on the last few slides regarding Default Gateways and routing, let's combine the topics together. To do so, we bring back the network diagram for the LAN covering a three story building.

If traffic is going from Net1 Sys2 to Net3 Sys2 (our earlier example), then:

- Net1 Sys2 would determine that the destination is not on its own network. Therefore, it would forward the traffic to its Default Gateway of the Net1 Gateway.
- The Net1 Gateway would check its routing table information and see that it does not have a route to network 3. It, therefore, needs to forward the traffic to the Default Gateway of the Intermediate Gateway.
- The Intermediate checks its routing table and sees that any traffic going to Net3 should go out the right-hand interface. It, therefore, sends the traffic to the Net3 Gateway (out the right interface).
- The Net3 Gateway checks its routing table and sees that traffic going to Net3 should go out the Net3 Gateway right-hand interface, so it sends the traffic on to Net3 Sys2.

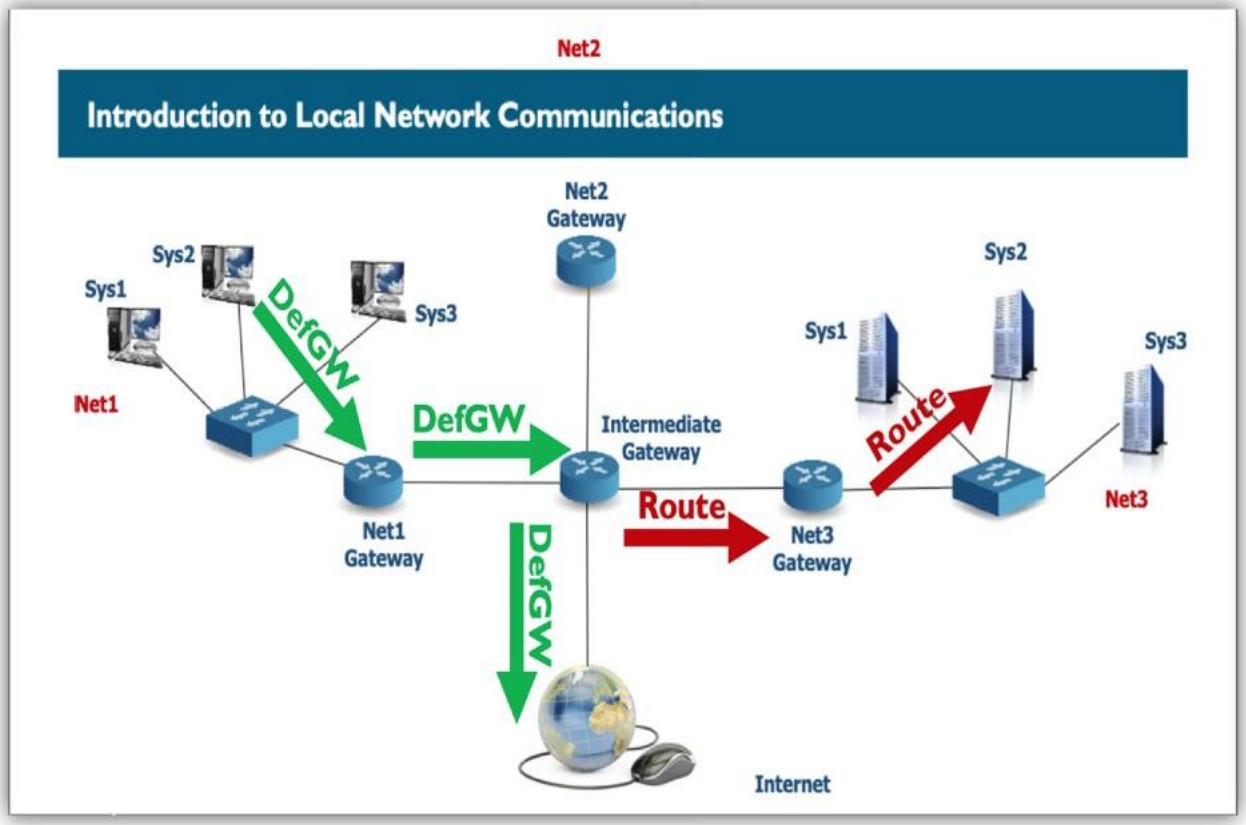
If the traffic were destined to the internet instead, once it reaches the Intermediate Gateway, that device would forward traffic to its own Default Gateway which would be the Internet Service Provider. Remember, a lot of steps are being left out here since we are concentrating solely on Default Gateways and Routing. The ARP process still has to be there, and so on.

Default Gateways & Routing

Net1 Sys2 to Net3 Sys2:

- Net1 Sys2 sends to DGW of Net1 Gateway
- Net1 Gateway sends to DGW of Intermediate Gateway
- Intermediate routes to Net3 Gateway
- Net3 Gateway routes to Net2 Sys2

If traffic is going to the Internet instead, the Intermediate would send the traffic to it's default gateway of the ISP



Routing Internet

Routing on the internet is the same process.

In this example, the packet would move from the home Cable Modem to the Denver router based on Default Gateways.

Then routing kicks in:

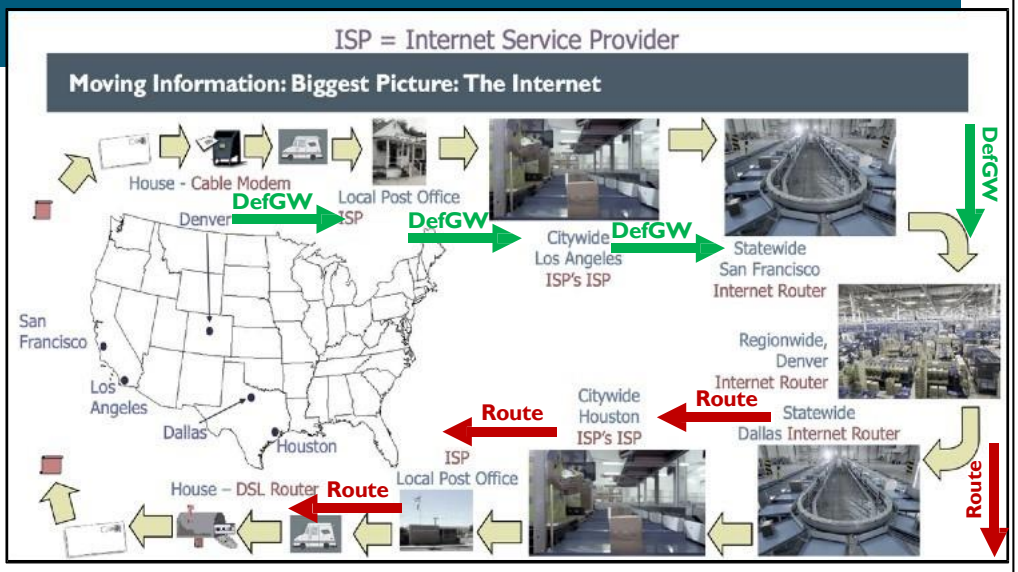
The Denver router would know...

- Texas IP addresses go to the Dallas Router.

- Dallas knows that IP is in Houston.

- That ISP knows which ISP that IP address belongs to.

- The final ISP knows where the customer is.



SANS | SEC301 | Intro to Cyber Security 119

Routing Internet

The only real difference between a LAN router and an internet router is their position. But functionally, they are the same.

To return to the example from earlier one more time, we now see how a routing table might work on the internet. As we saw just a handful of slides ago, the packet in this example would move from the Cable Modem to the ISP, then to the ISP's ISP, then to the San Francisco Router, and on to the Denver Router based on Default Gateways. This process happens because each of those systems says, "I have no idea where the destination IP address is; I need to send it to my Default Gateway to see if it knows."

At the Denver router, the routing tables kick in. The Denver router is not sure where that IP address is, but knows it is in Texas somewhere. Its routing table says that anything destined to a Texas IP address should go to the router in Dallas. The Dallas router does not know where that IP address is precisely but knows it is in Houston somewhere, so the routing table has the router forward to an ISP in Houston. The ISP's ISP in Houston does not know whose IP address the destination is, but recognizes the address as belonging to one of its customer ISPs, so the routing table causes the packet to go to that company. The customer's ISP sees that the destination IP address is one of its customers and their routing table causes the packet to route to that particular house.

If you follow it through, it is the same process, just on a grander scale. Do remember this is a simplified explanation of the routing process. One crucial fact that is left out here is redundancy. For example, if the Dallas router were down and could not receive traffic, the Denver router might have ten or more alternate routes to get the packet to the correct destination.

Routing Internet

Routing on the internet is the same process.

In this example, the packet would move from the home Cable Modem to the Denver router based on Default Gateways.

Then routing kicks in:

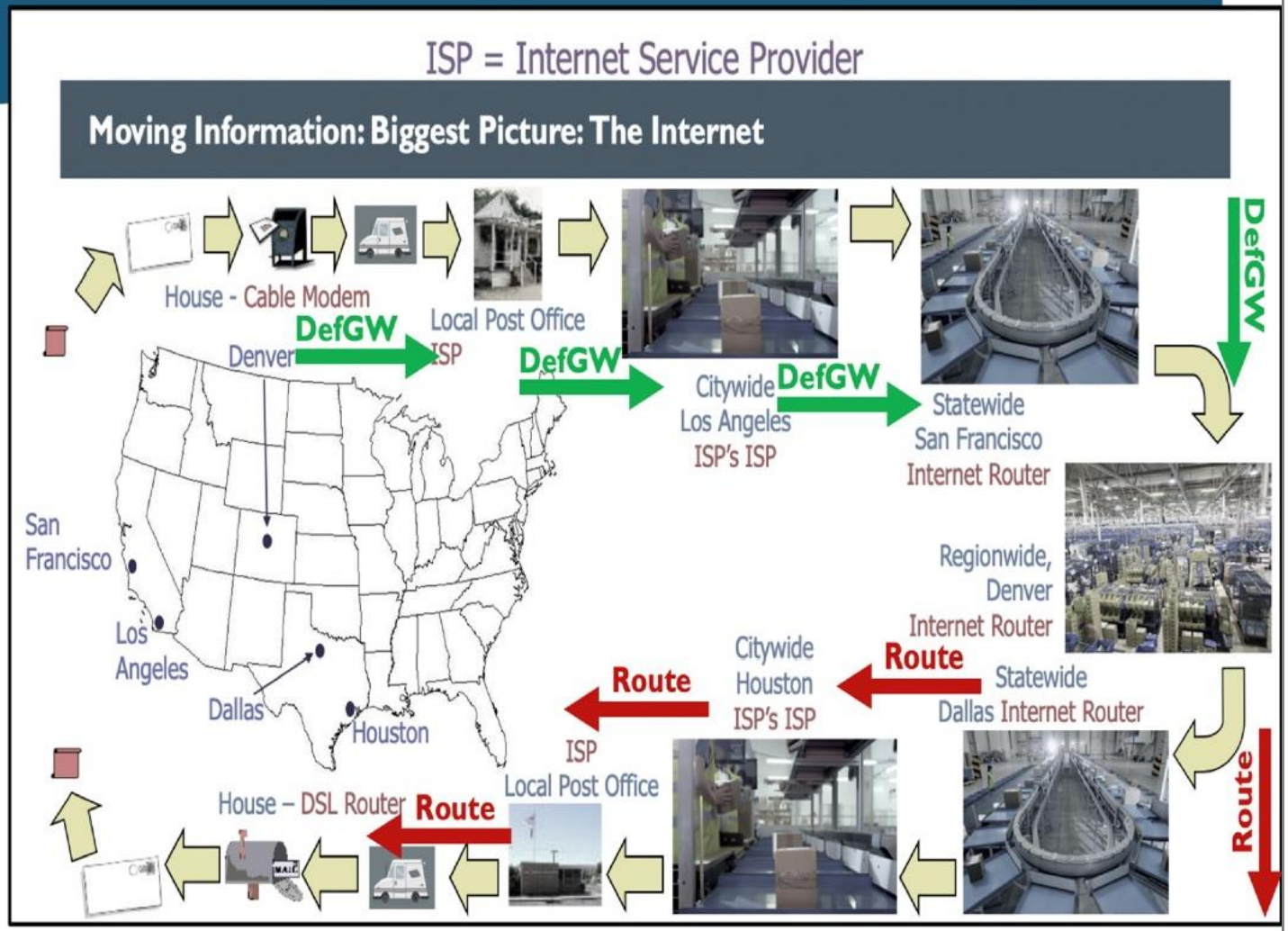
The Denver router would know...

- Texas IP addresses go to the Dallas Router.

- Dallas knows that IP is in Houston.

- That ISP knows which ISP that IP address belongs to.

- The final ISP knows where the customer is.



Internet Control Message Protocol (ICMP)

- The Internet Protocol (IP) is "best effort":
  - Connectionless/Unreliable/Unacknowledged
- Error reporting is required:
  - ICMP is the error reporting mechanism
  - Note: Error reporting, not error correcting
- Also includes Ping:
  - Used to discover if a host is up and listening

SANS | SEC301 | Intro to Cyber Security 122

Internet Control Message Protocol (ICMP)

The Internet Protocol (IP) is a best effort protocol, meaning that it is connectionless, unreliable, and unacknowledged. You send an IP packet and hope it gets to the destination. Because there is no connection, there is also no acknowledgment of receipt, making the protocol unreliable. (Any form of connection, acknowledgment, or reliability has to occur at a higher level of the OSI stack.)

There is a need in IP for an error reporting mechanism. The Internet Control Message Protocol (ICMP) fills this need. It is the error reporting mechanism of the IP protocol. Examples are many, but here are a few:

- If the destination is too many hops (routers) away, the IP header's Time-To-Live field decrements to a 1. The router that receives the packet with a TTL of 1 sends back an ICMP "Time to live exceeded in transit" message.
- If something is wrong with the routing tables and a router cannot figure out where to send a packet, it sends an ICMP "Destination unreachable, Network unreachable" message back to the source.
- If the routing is good and the packet makes it to the final destination network, but the destination computer will not respond to an ARP for some reason, that router sends the source an ICMP "Destination unreachable, Host unreachable" message.

There are many such examples. However, in all those examples (or any others we might use), ICMP reports the errors but does nothing to resolve or correct them. Still, from both a network troubleshooting and network security perspective, having a firm understanding of the ICMP protocol is vitally important.

Another element of the ICMP protocol is called Ping, which is a little different from other types of ICMP messages in that it is not for automatic error reporting. Instead, Ping determines if a host is up (turned on) and listening on the network.

Much like a radar ping where you send something out and see what you get back, with Ping you send an ICMP "Echo Request" message to a destination. If that computer is powered on and connected to the network, you should get an ICMP "Echo Response" back. (This assumes that you do not get one of the unreachable messages we discussed on the last page.)

Ping

- ping
  - Used to test connectivity between two hosts
  - Utilizes ICMP

```
$ ping Google.com
PING google.com (216.58.194.142): 56 data bytes
64 bytes from 216.58.194.142: icmp_seq=0 ttl=51 time=34.163 ms
64 bytes from 216.58.194.142: icmp_seq=1 ttl=51 time=33.336 ms
64 bytes from 216.58.194.142: icmp_seq=2 ttl=51 time=32.836 ms
64 bytes from 216.58.194.142: icmp_seq=3 ttl=51 time=32.316 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 32.316/33.163/34.163/0.681 ms
$
```


Ping

Ping (named for the sonar operation used to locate objects) [1] creates an ICMP echo request packet and sends it to the target machine. When it reaches the target, the data in the packet is echoed back to the originating host. The originating machine collects the reply, along with some information about how long the trip took and reports it to the user.

What makes PING so useful is that if the path between the two machines is blocked, PING reports that the destination is unreachable. You can then investigate the problem to see if it is a routing error, failed equipment, or some other cause.


```
$ ping Google.com
PING google.com (216.58.194.142): 56 data bytes
64 bytes from 216.58.194.142: icmp_seq=0 ttl=51 time=34.163 ms
64 bytes from 216.58.194.142: icmp_seq=1 ttl=51 time=33.336 ms
64 bytes from 216.58.194.142: icmp_seq=2 ttl=51 time=32.836 ms
64 bytes from 216.58.194.142: icmp_seq=3 ttl=51 time=32.316 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 32.316/33.163/34.163/0.681 ms
$
```


[1]The story of Ping: https://www.webcitation.org/5saCKBpgH

Traceroute

- tracert (Windows)/ traceroute (Linux / Unix)
  - Determines the network path between two hosts ("Trace the Route")

```
$ traceroute Google.com
traceroute to google.com (216.58.194.110), 64 hops max, 52 byte packets
 1  192.168.7.1 (192.168.7.1)  5.685 ms  3.550 ms  4.680 ms
 2  192.168.1.254 (192.168.1.254)  5.896 ms  5.827 ms  4.690 ms
 3  75-1-64-1.lightspeed.snantx.sbcglobal.net (75.1.64.1)  45.500 ms  43.945 ms  40.192 ms
 4  70.232.192.61 (70.232.192.61)  33.660 ms  34.751 ms  33.369 ms
 5  75.28.192.122 (75.28.192.122)  34.854 ms  34.716 ms  34.545 ms
 6  12.83.39.9 (12.83.39.9)  25.908 ms
    12.83.39.17 (12.83.39.17)  27.221 ms  26.079 ms
 7  12.122.85.197 (12.122.85.197)  31.555 ms  31.191 ms  30.878 ms
 8  206.121.120.70 (206.121.120.70)  31.245 ms  31.302 ms  30.648 ms
 9  108.170.252.161 (108.170.252.161)  32.695 ms
    108.170.252.129 (108.170.252.129)  30.929 ms *
10  108.170.226.54 (108.170.226.54)  33.241 ms
    108.170.230.116 (108.170.230.116)  31.802 ms
    dfw06s48-in-f110.1e100.net (216.58.194.110)  30.522 ms
$
```

The dual IP addresses at hops 6, 9, & 10 are probably due to load-balancing.

Traceroute

Tracert helps you discover the actual path the network uses between two systems. It does this by manipulating the Time-To-Live (TTL) field of the IP header. This field is decremented by each router the packet passes through. If a router ever receives a packet with a Time-To-Live value of 1, it throws the packet away and sends a "Time-To-Live Exceeded" message back to the originator of the packet. Notice that the Time-To-Live Exceeded comes from the router that throws the packet away.

When you do a traceroute, your computer first generates a packet with a TTL of 1. Your Default Gateway receives it with a 1, throws the packet away, and responds with a TTL Exceeded message from the Default Gateway's IP. Then you send a packet with a TTL of 2. Your Default Gateway decrements the TTL to a 1 and sends it to the next router which receives it with a 1. That router throws the packet away and responds with a TTL Exceeded message from that router's IP address. You then send a packet with a TTL of 3…

Your computer continues incrementing the TTL values until it receives a response from the final destination. Just a couple of notes:

- First, your computer actually sends each TTL value three times (three 1's, three 2's, etc.) and provides some information on round-trip time.
- Second, on Windows, the command is **`tracert <IP>`** but on Unix, Linux, and Mac the command is **`traceroute <IP>`**

For example: traceroute google.com
Or tracert 172.217.12.78

```
$ traceroute Google.com
traceroute to google.com (216.58.194.110), 64 hops max, 52 byte packets
 1  192.168.7.1 (192.168.7.1)  5.685 ms  3.550 ms  4.680 ms
 2  192.168.1.254 (192.168.1.254)  5.896 ms  5.827 ms  4.690 ms
 3  75-1-64-1.lightspeed.snantx.sbcglobal.net (75.1.64.1)  45.500 ms  43.945 ms  40.192 ms
 4  70.232.192.61 (70.232.192.61)  33.660 ms  34.751 ms  33.369 ms
 5  75.28.192.122 (75.28.192.122)  34.854 ms  34.716 ms  34.545 ms
 6  12.83.39.9 (12.83.39.9)  25.908 ms
    12.83.39.17 (12.83.39.17)  27.221 ms  26.079 ms
 7  12.122.85.197 (12.122.85.197)  31.555 ms  31.191 ms  30.878 ms
 8  206.121.120.70 (206.121.120.70)  31.245 ms  31.302 ms  30.648 ms
 9  108.170.252.161 (108.170.252.161)  32.695 ms
    108.170.252.129 (108.170.252.129)  30.929 ms *
10  108.170.226.54 (108.170.226.54)  33.241 ms
    108.170.230.116 (108.170.230.116)  31.802 ms
    dfw06s48-in-f110.1e100.net (216.58.194.110)  30.522 ms
$
```

There's No Place Like 127.0.0.1

- 127.0.0.1 is the "loopback" address
- Sending traffic to that IP is always sending it to yourself:
  - The most common address in use today
  - Services inside the computer use it extensively

There's No Place Like 127.0.0.1

Before we leave our discussion of OSI Layer 3, we need to be aware of one more IP address, specifically, 127.0.0.1. This is a special address reserved as the "loopback" address. This means that anytime your computer sends a packet to that address, it is sending it to itself.

This special NIC is built in to the TCP/IP stacks. If those are installed and running on the computer (in other words, if you can communicate on the network) this interface is always active.

It has become the most commonly used IP address in the world. Many services inside your computer need to communicate with other services inside the computer. They use this IP address as a communications pathway that they can rely on being present.

A little-known fact by many network professionals is that any IP address that starts with 127 acts as a loopback address. The behavior is identical if you send traffic to 127.0.0.1, 127.0.0.2, 127.3.3.3, 127.201.119.5, 127.127.127.127, or any other address in the 127 range. This does not work on the Mac operating system (OS X) or any other system based on BSD Unix. But it certainly works on Windows and most Linux systems.

(The joke in the title of the slide: 127.0.0.1 is equivalent to home for your computer, so “There’s No Place Like 127.0.0.1” is equivalent to "There's no place like home”, which is a line from the movie “The Wizard of OZ.”)

SANS

Port Numbers
TCP
UDP
Comparing TCP and UDP

OSI Layer 4

OSI Layer 4

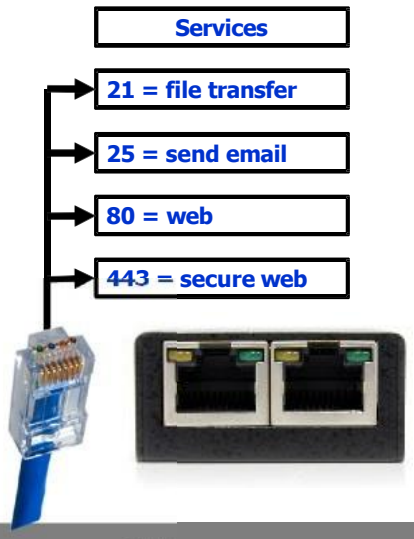
As we move up the OSI protocol stack, we find ourselves at OSI Layer 4, the Transport Layer. There are a number of protocols that you find at this layer, and some of those we will discuss later in the course. For now, we want to focus on the two most common. Specifically, the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Any discussion of TCP and UDP also requires a discussion of port numbers, so that is where we will begin.

Ports: Network Ports

- Network ports:
  - The logical (software) location of a service
  - Well-known ports (0–1,023)
  - Registered ports (1,024–49,151)
  - Ephemeral ports (49,152–65,535)

Why 49,152? It is the first 2-byte binary number that begins with 11
11000000 00000000 = 49,152
So – All ephemeral port #'s start with 11
Well-known and registered ports start other than 11

- Hardware ports
- Many other ports in IT



Ports: Network Ports

On a computer, a network port is a number representing a logical connection point for communications. It is not a physical port that you can see on the back of the computer. Rather, it is mapped via software to a specific function or application. When a packet arrives at a computer (coming in over the twisted pair cable for instance), the packet contains a destination port number. It is that number that notifies the server of which service the user wants to interact with. For example:

- Destination port 21 instructs the server to provide file transfer service using the File Transfer Protocol (FTP).
- Destination port 25 advises the server the user wants to send email using the Simple Mail Transfer Protocol (SMTP).
- Destination port 80 advises the server that the user wants to view the web page via insecure (unencrypted) HyperText Transfer Protocol (HTTP).
- Destination port 443 indicates to the server that the user wants to view the web page using secure (encrypted) HyperText Transfer Protocol Secure (HTTPS).

In a packet, the field of a header that contains the port number is 2 bytes in length. If you perform the math function we learned earlier, you know that a 2-byte field can only equal numbers ranging from 0 to 65,535. Therefore, that is the port numbers range. Ports 0 to 1023 are reserved for use by certain services that run at the application layer of the stack. These are known as "well-known ports" because the most common and well-known network applications use them. A full list of port assignments can be found at the link below. There are many places on the internet

where you can find this list, but because the ports are officially assigned by the Internet Corporation for Assigned Names and Numbers (ICANN), we recommend using the list at its site. (ICANN was formerly known as the IANA, hence www.iana.org below).

https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

Many common commercial and open-source applications use ports with preassigned numbers from 1,024 through 49,151. These are known as registered ports because their use has been "registered" with the ICANN. You should avoid using the registered ports for applications other than the ones to which they are registered. It is not illegal to use a port for your application that is already registered to another application, but this may cause network and communication conflicts and should be avoided.

Network communications go from a source port on the client side to a well-known (or sometimes a registered) port on the server side. With the majority of network protocols, the client selects its port number from the range of ephemeral ports. Ports from 49,152 through 65,535 fall into this range.

So, if you browse the web, your computer is the client and the website you are going to is the server. The communication would be going from an ephemeral port on your computer (say 49,152) to the well-known port 80 on the server. When the server sends information back to you, it would be traveling from the source port of 80 on the server back to your ephemeral port (49,152 in our example).

A common question is, "Why port 49,152? What is so special about that number that they started the ephemeral range there? In many (perhaps most) cases, if you look at TCP/IP in binary, it makes sense. (For this explanation, you will want to remember the binary/decimal/hex and computer math discussion from earlier in the course.)

It just so happens that 49,152 is the smallest 2-byte binary number that begins with 11. In other words, in binary, 49,152 is 11000000 00000000. If we convert those 2 bytes to decimal, we get the numbers 192 for the most significant byte and 0 for the least significant byte. As you may recall, the formula to calculate the value of a 2-byte field is (most significant byte * 256) + least significant byte. So:

(11000000 * 256) + 00000000 = 49,152
or
(192 * 256) + 0 = 49,152

So, every ephemeral port number begins with two 1's. In software, it becomes simple to branch code based on this fact. If the first 2 bits are 11, the port is ephemeral and branch the code to deal with the port number as such. If it is any combination except 11, it is either a well-known or registered port, so branch the code to deal with that situation.

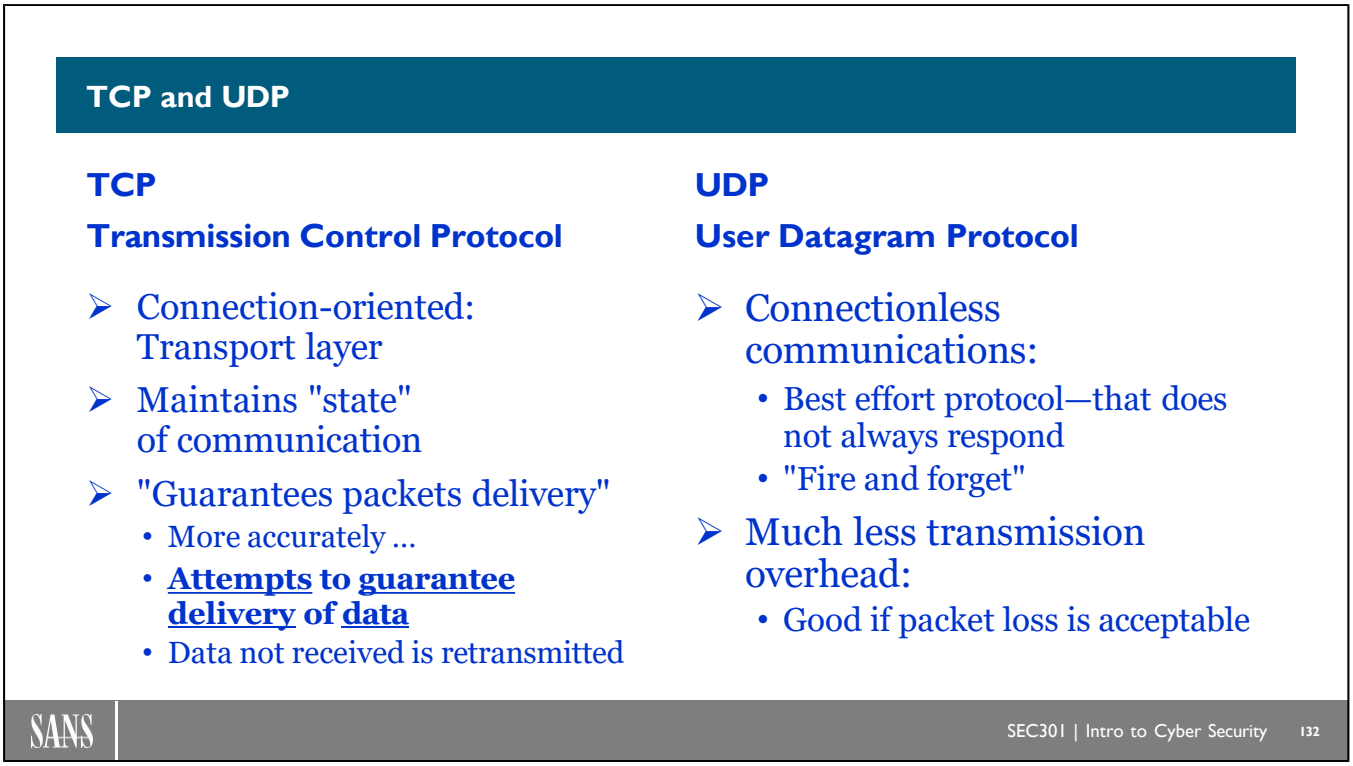
Hardware Ports

In another context, we may refer to a port when we discuss the location on a networking component (for example, switch, hub, and router) to which a cable might be connected. A network switch that has eight ports is described as an "eight-port switch."

Note that the term *port* has multiple definitions in the IT world. A port could also refer to a wall jack into which a network cable could be inserted (that is, a wall port). If I have software that currently runs on the Linux operating system and I want it to run on Windows, I "port" the software to the new operating system. So, for example, we have the terms:

- Network ports
- Hardware ports
- Wall ports (also called "wall jacks")
- Software ports
- Printer or video ports (as in on the back of a PC)
- USB ports

The point here is to be aware that in the IT world, the word "port" is a dangerous term because it actually has at least one-half dozen different definitions. Sometimes, when talking to someone, context may not make it clear which kind of port they are talking about. In that case, it is important to ask.



TCP and UDP

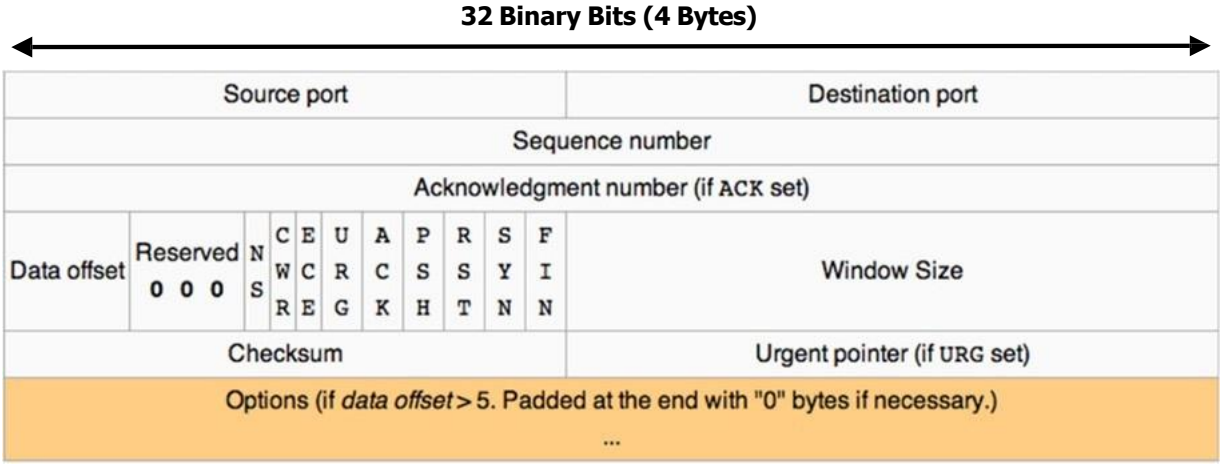
There are two primary Layer 4 protocols. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

TCP is a connection-oriented protocol, meaning that it tracks the state of communication between the two communicating parties. To say it another way, it establishes a connection between the parties and monitors the state of that connection. A common misunderstanding about TCP comes from the wording in many books and references. You often see the words, "Guarantees packet delivery" or something similar. In fact, that is not possible. TCP runs on top of the IP protocol. You should remember from earlier that IP is a best effort protocol that does not guarantee packet delivery. Therefore, TCP can't do so either. What TCP does have is an error control mechanism. When you send 500 bytes of data with TCP, and the recipient only receives the first 400 bytes, the sender can tell this has happened and retransmits the last 100 bytes. That is not guaranteed packet delivery. It is an attempt to ensure data delivery.

UDP, by contrast, is connectionless and does not have state of any kind. It is purely a best effort protocol. Indeed, in many cases when you send a packet of data using UDP, you receive no response at all. Meaning there is much less packet overhead, which is fine if some packet loss is acceptable.

Right now, this explanation is probably confusing to you. Stay with us through the next several slides and allow us to clarify. We will have to give you a little detail for the explanation to make sense.

The TCP Header



Lots of detail here. We focus on the source and destination port numbers, sequence, and acknowledgment numbers, and the flags (URG, ACK, PSH, RST, SYN, and FIN).

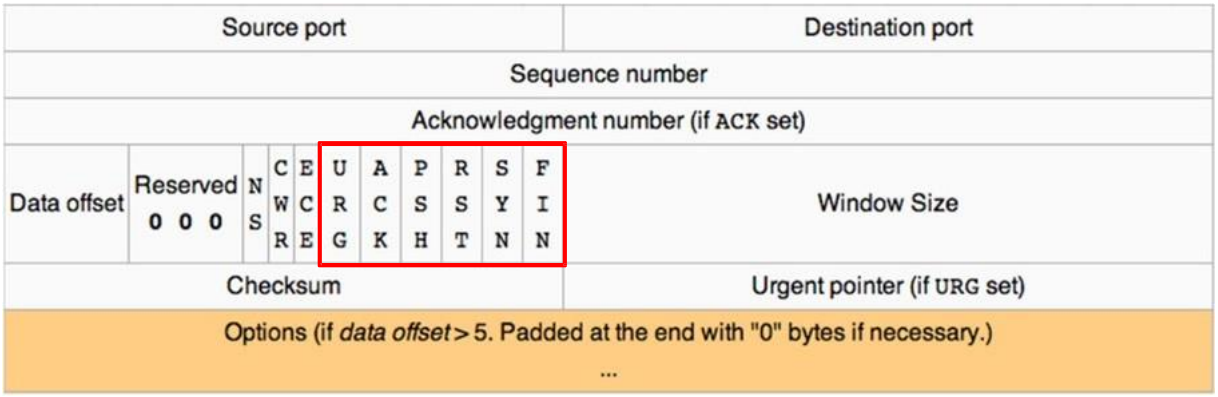
The TCP Header

The TCP header has a number of fields as you can see. Like the IP header, the shortest this header is allowed to be is 20 bytes. Unlike the IP header that almost never has options added, TCP header options are common.

The first two fields in the TCP header are the Source Port and the Destination Port. These contain the port numbers we have alluded to a number of times. The well-known port indicates to the server the service we want to use, and the client uses an ephemeral port. Which of those ports ends up in which of those fields depends on which direction the packet travels. If it goes from the client to the server, the source port is ephemeral, and the destination port is well known. However, if the packet returns from the server back to the client, the source port is well known, and the destination port is the ephemeral the communication originated from. Note that both port number fields are 2 bytes in length, meaning the largest number they can contain is 65,535.

The next two fields in the TCP packet are the Sequence Number and the Acknowledgment Number. Each of these is a 4-byte (32-bit) field, meaning the sequence and acknowledgment numbers can be large (up to 4,294,967,295 to be exact). These numbers make up the core of the TCP error control mechanism. They allow the sender to determine if the receiver didn't get some of the information sent. When that happens, the sender retransmits the information that did not make it to the destination.

TCP Header Flags



- **Urgent (U or URG)**
- **Acknowledgment (A or ACK)**
- **Push (P or PSH)**
- **Reset (R or RST)**
- **Synchronize (S or SYN)**
- **Finish (F or FIN)**

TCP Header Flags

In the TCP Header are a series of flags. If you understand what these flags mean, you can tell what happens in a TCP session by looking at those flags.

The NS flag, CWR flag, and ECE flag are recent additions to the protocol and are rarely used today. We won't muddy the waters here by explaining them. In fact, we are going to concentrate on only a few of these flags.

The **SYN** flag is used during the initial three-step handshake. It essentially tells the distant end that you want to synchronize a connection.

The **ACK** flag is used to inform the distant end that you are acknowledging bytes that were sent to you in one of their prior packets. This is integral to the error control mechanism of TCP.

The **FIN** flag notifies the other end of the communication that you have nothing more to say to them and want to terminate the connection. When received, a four-step session teardown ensues.

The **RST** flag happens when something has gone wrong with a session and cannot be fixed. Either side can send a reset at any time. It causes a TCP session to immediately terminate. You can think of RST as a rude goodbye.

The **PSH** flag indicates a packet contains data. If you have captured suspicious network traffic, it is simple to zero in on the packets with a PSH flag and determine if an attacker accessed your data.

TCP Header: Wireshark

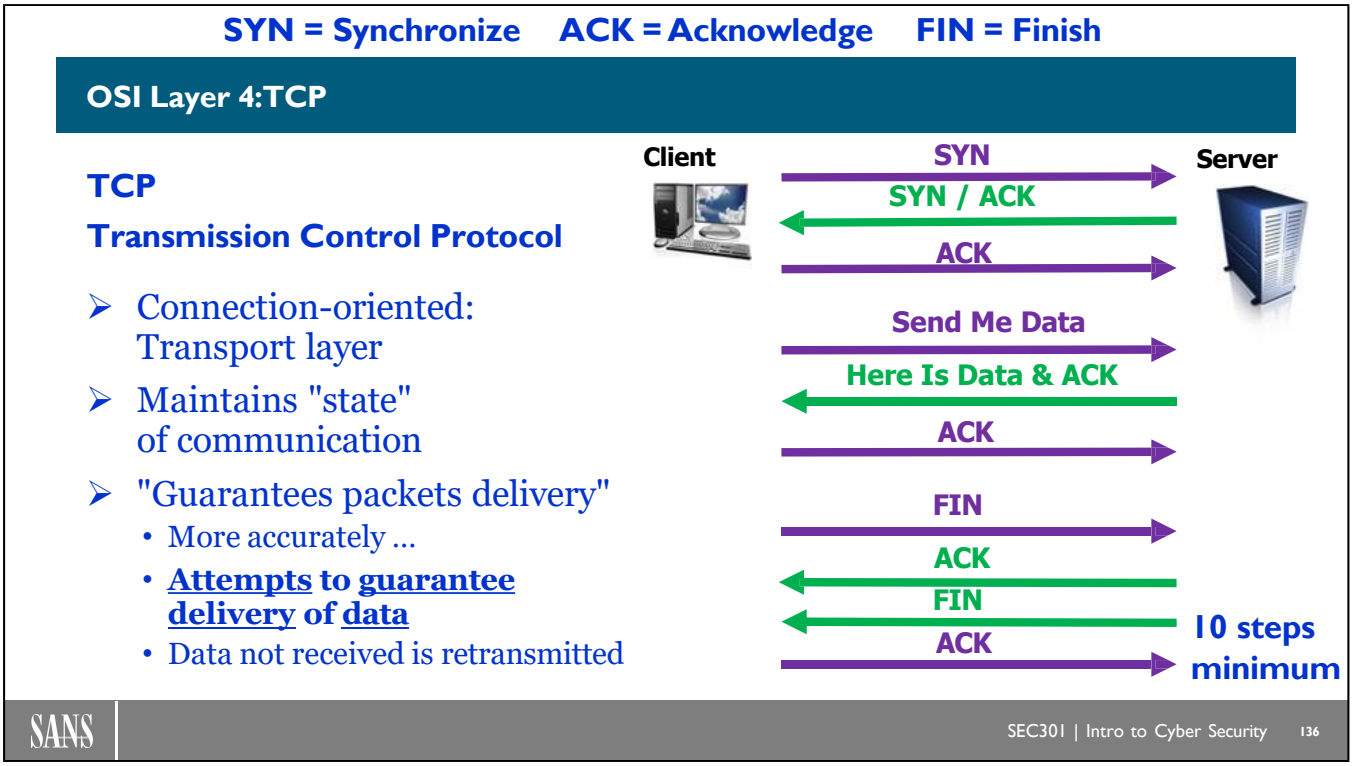
```
▼ Transmission Control Protocol, Src Port: 56669, Dst Port: 443, Seq: 1, Ack: 1, Len: 5
    Source Port: 56669
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 5]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 6    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 4096
    [Calculated window size: 4096]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xcbb4 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (5 bytes)
    [Reassembled PDU in frame: 8]
    TCP segment data (5 bytes)
```

Note: In Wireshark, anything in [square brackets] is not information found in the packet. It is information Wireshark extrapolates about the packet. That is why you don't see "Stream index" for example in the earlier diagram … it is metadata provided by Wireshark.

TCP Header: Wireshark

Here we see the TCP header displayed in the Wireshark tool. In this case, we can tell that this packet is part of a secure web browsing session using the HTTPS protocol because we see the well-known port 443. The fact that the PSH and ACK bits are set indicate to us this packet contains data and is also acknowledging receipt of data from the other system. **Note:** In Wireshark, anything in [square brackets] is not information found in the packet. It is information Wireshark extrapolates about the packet. That is why you don't see "Stream index" in the diagrams … it is metadata provided by Wireshark.


```
▼ Transmission Control Protocol, Src Port: 56669, Dst Port: 443, Seq: 1, Ack: 1, Len: 5
    Source Port: 56669
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 5]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 6    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 4096
    [Calculated window size: 4096]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xcbb4 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (5 bytes)
    [Reassembled PDU in frame: 8]
    TCP segment data (5 bytes)
```



OSI Layer 4: TCP

Taking a closer look at the TCP process, we see that there are several steps (packets) required for even the most straightforward transaction. In fact, the absolute minimum number of packets needed for a successful TCP communication is 10.

If you look closely at the TCP session in the slide, you see three distinct areas. The first three packets are the TCP Three-Step Handshake. The next three packets are the data exchange. The last four packets are the Four-Step Session Termination. Below, we look at each of these three areas individually.

TCP Session Open: The Three-Step Handshake

The TCP Three-Step Handshake occurs anytime there is a new TCP session between a client computer and a server computer. The purpose of the handshake is to establish the connection and let each side know the other computer's initial sequence number. This exchange is critical as it is the sequence number that is used in the error control mechanism of TCP (as you can see on the next slide).

To begin the process, the client sends a TCP packet to a particular port on the server, for example, port 80 for HTTP service. In the TCP header, there is a series of flag bits. One of them, specifically the SYN bit (short for synchronize) is a binary 1. This setting in the packet effectively tells the server, "Here is the client's sequence number; please synchronize on it."

When that packet arrives at the server, if that port is open, the server responds with a TCP packet. In this packet's TCP header, two of the flag bits are binary 1's. Specifically, the SYN and the ACK (short for Acknowledge). The packet effectively tells the client, "Here is the server's sequence number, please synchronize on it, and I acknowledge receipt of the first packet."

Finally, the client responds with a TCP packet with the ACK bit in the TCP header set to a binary 1 (but not the SYN bit). This tells the server, "We are done synchronizing, and I acknowledge receipt of the second step." At this point, we have a fully established TCP session between the client and server.

TCP Session Communications

After the handshake, we see the client ask the server for some data and the server responds with that data. But that is not all that is going on. The client asks the server for some data. When the server returns that data, it also acknowledges the bytes that made up the request. When the client receives the requested data, it acknowledges receipt of those bytes. This is the TCP error correction mechanism in action. If, for example, the client did not acknowledge receipt of enough bytes of data, then the server would retransmit them.

TCP Four-Step Session Termination

At some point, one side or the other is finished talking and wants to close the connection. Note that either side of the communication can initiate this process. In this example, the client initiates the termination.

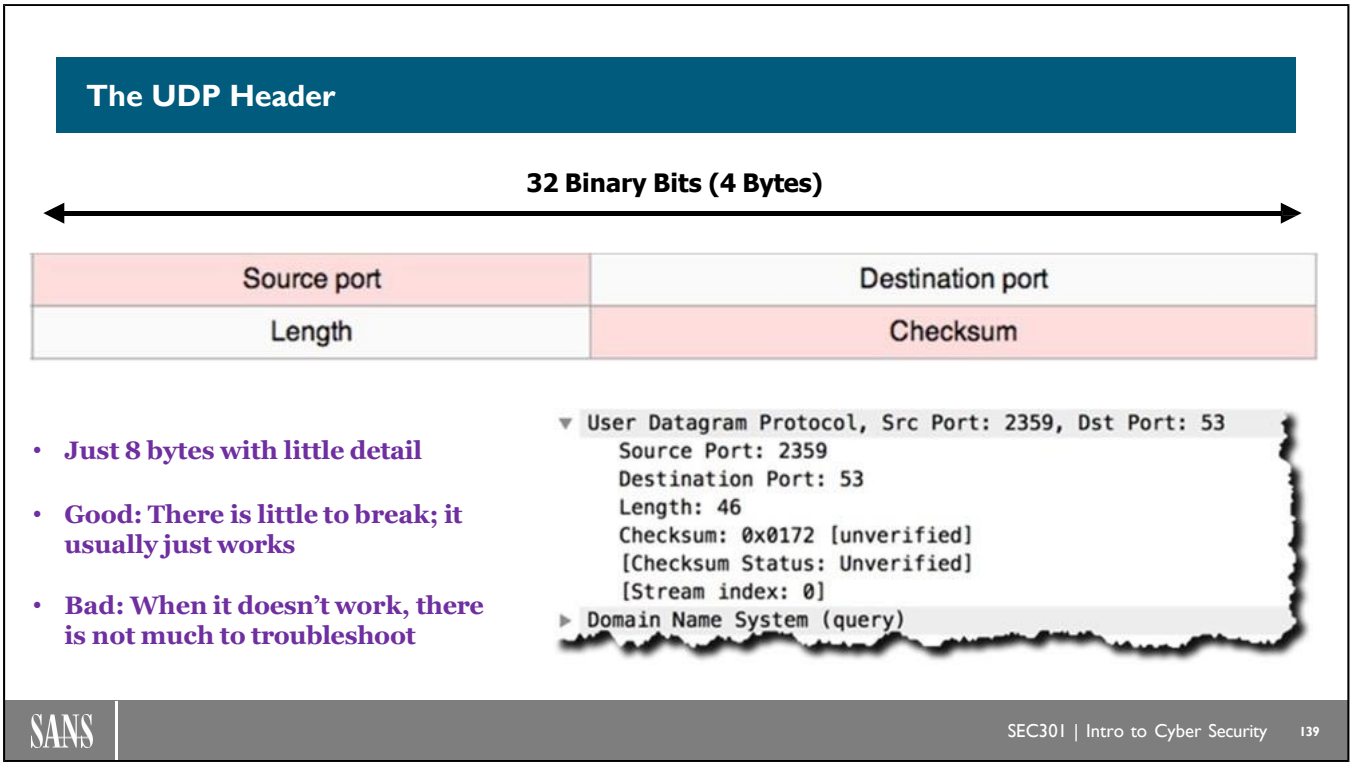
In a normal TCP session termination, one side (in this case the client) sends a TCP packet with the FIN (short for Finish) flag bit in the TCP header set to a binary 1. When the server receives that packet, it responds with an acknowledgment packet (the ACK bit set to binary 1). The server then sends a TCP packet with the FIN bit set to binary 1. The client receives it and responds with its acknowledgment.

These four steps constitute a "normal" or "gentle" teardown of a TCP session. In other words, if everything in the session worked correctly, this is how the session usually terminates. Do note that the two middle steps both come from the server. This means that the server could choose to put both of those steps into a single packet (a packet with both the FIN and ACK bits set to binary 1). This alternative method is not only possible but also it is common today.

BUT…what if the session cannot terminate normally? If something occurs to get a TCP session out of whack so to speak, then either side can at any time send an RST (Reset) packet. This instantly terminates the session. No further data that is part of this session will be transmitted or processed by either side. The session terminates instantly. Although less common, this does happen from time to time on a network.

TCP Summary

TCP is a suitable protocol for ensuring the delivery of information to a destination. However, this assurance comes at a price. When compared to our next protocol, TCP has tremendous overhead. Still, TCP is the way to go when you want to make sure that a message reaches its destination.



The UDP Header

This slide shows the actual fields in a UDP packet. You can clearly see that the UDP packet is much less complicated than the TCP packet.

The UDP packet fields follow:

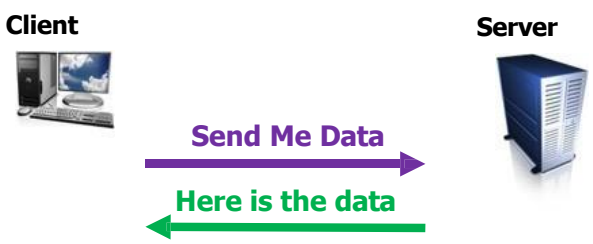
- **Source Port and Destination Port:** Specifies the particular UDP service or application the packet uses. Again, this is a 2-byte field, indicating a maximum port number of 65,535.
- **UDP Length:** This is the total number of bytes in the UDP packet.
- **Checksum:** This provides a basic means for determining whether information has been altered during transmission. It is not infallible and should not be relied upon as a true indicator of the integrity of packet data.

In the Wireshark capture above, we see the UDP header of a DNS lookup (an attempt to determine the IP address for a domain). We can tell this because the Destination Port is 53, which is assigned to the DNS (Domain Name System) protocol.

Reference

http://www.ietf.org/rfc/rfc768.txt

OSI Layer 4: UDP



Much less overhead traffic, but zero error correction or notification

UDP

User Datagram Protocol

- Connectionless communications:
  - Best effort protocol—that does not always respond
  - "Fire and forget"
- Much less transmission overhead:
  - Good if packet loss is acceptable

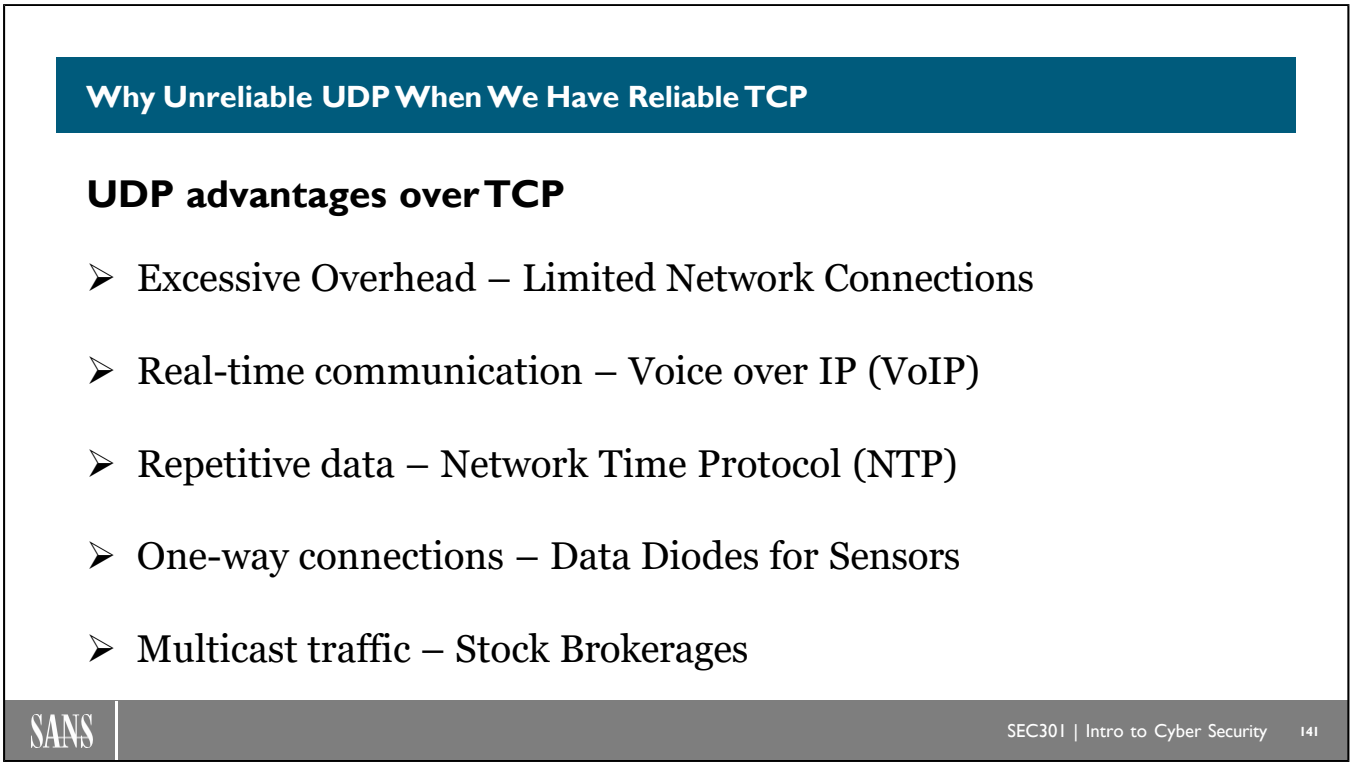
OSI Layer 4: UDP

Like TCP, the *User Datagram Protocol (UDP)* is found at OSI Layer 4 and uses IP as its underlying delivery service, moving pieces of information from one host to another. Unlike TCP, however, UDP does not provide any sort of error correction or notification. UDP sends the packets, but it does not provide any mechanism for ensuring they arrive properly, intact, or in the right order. As a result, UDP is called a *connectionless protocol*; it does not maintain persistent information about the state of the communication between the sending and receiving machines in the same way that TCP does. Whereas TCP will do its part to ensure the delivery of data, UDP leaves this task to the application. For this reason, UDP transmission is often jokingly referred to as "fire and forget" or "send and hope."

The lack of error correction does have its advantages. There are far fewer packets required. There is no three-step handshake to establish a connection or four-step teardown to end a session. There also are no packets to acknowledge receipt of data (unless the application itself sends them).

The minimum number of packets required to exchange data using TCP is 10 as you just saw. By contrast, UDP can exchange that information in just two packets. Of course, there is no guarantee of delivery of that data.

In some cases, small amounts of data loss are acceptable. For example, some applications transmit information every 60 seconds. So if the data is not received this time, it will be in a minute. Also, sometimes you DO NOT want the retransmissions of data that TCP provides (Voice over IP or VoIP is a good example).



Why Unreliable UDP When We Have Reliable TCP

A common question is, "Why would you use unreliable UDP when you have reliable TCP?" Good question! We are glad you asked!

There are certain times when UDP is more desirable. Examples include:

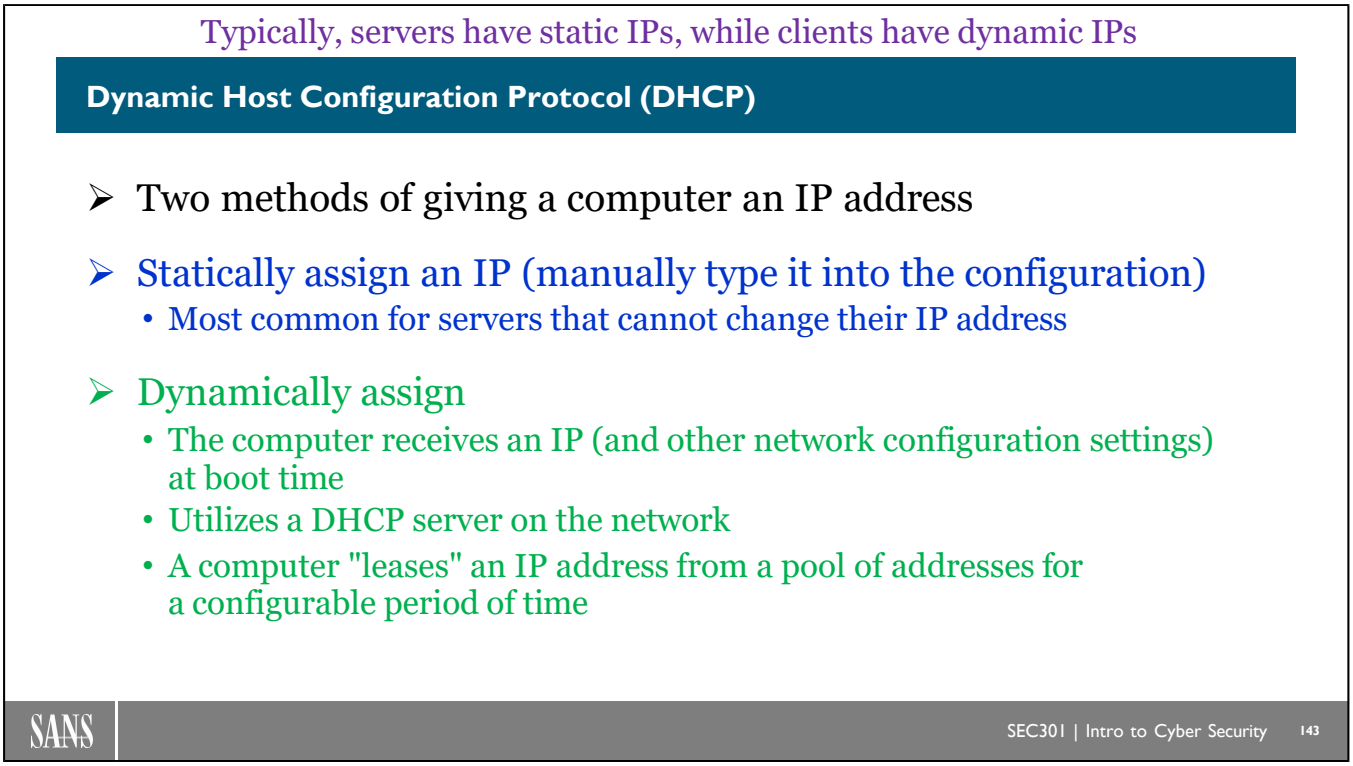
- When you have limited network bandwidth and do not want the excessive overhead that TCP's error checking requires.
- Real-time communications such as Voice over IP (VoIP) or true streaming video. In these instances, you do not want TCP's retransmissions—they would mess up the communication.
- In cases were data is repetitive, such as Network Time Protocol and certain routing protocols. In these cases, the data is sent every minute or two, so if you don't get it this time, you will shortly.
- Sometimes it is absolutely vital that data can only travel in one direction (think nuclear power plant monitoring stations). In those cases, a hardware solution called a "Data Diode" is used. This is a chip constructed in such a way that it is only physically possible for data to move one direction. This means you cannot receive the acknowledgements required by TCP, so you must use UDP with Data Diodes.
- Think about a stock brokerage that is sending data from one source to many recipients simultaneously. A technology called "multicast" is used for this. It simply does not work to have all of those recipients acknowledging receipt of data—you must use UDP for this.

SANS

Dynamic Host Configuration Protocol (DHCP)
Domain Name System (DNS)
Network Address Translation (NAT)

Necessary Network Services

This page intentionally left blank.

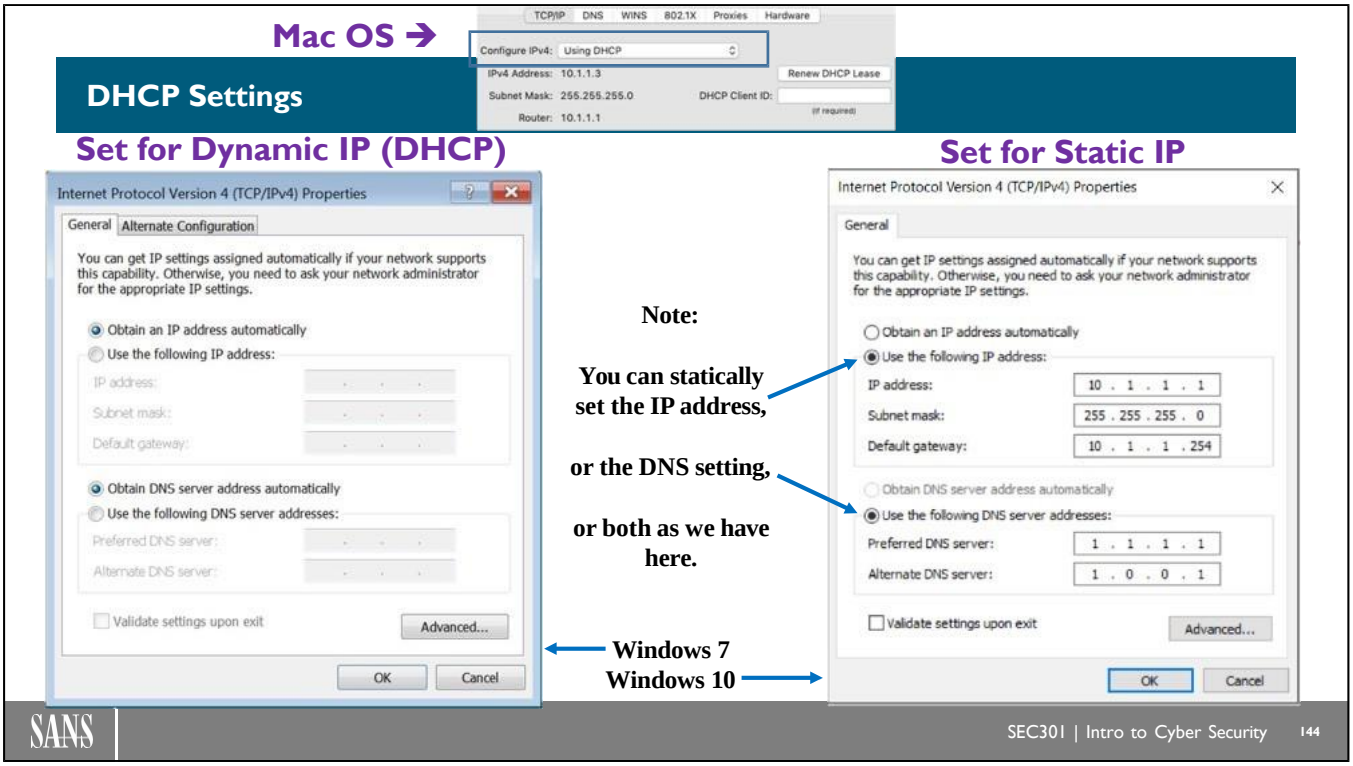


Dynamic Host Configuration Protocol (DHCP)

When you configure a computer to communicate on a network, you have two options for giving that system an IP address and other necessary network configuration settings. The first option is to statically assign those settings. This is typically done by typing them into the computer's configuration screens somewhere. This is very commonly done with servers of various types (file servers, print servers, authentication servers, etc.). Those systems must retain the same IP address from one day to the next.

The second option is to have network settings assigned to the computer automatically at boot time using the DHCP protocol. Organizations commonly employ this method on client computers since those do not have to have the same IP address from day to day, and dynamically assigning addresses means that network administrators do not have to manually type the addresses on each computer. This also simplifies moving a computer from one network to another (think about student laptop computers on a college campus moving from classroom network to classroom network throughout the day).

A DHCP server on the network has a pool of addresses set aside for dynamic assignment. When a PC boots, it sends a broadcast on the network asking if there is a DHCP server available that can assign an address. The DHCP server replies with the address and, typically, other network configuration settings such as the network mask, Default Gateway, and other settings. That PC "leases" those settings for a configurable period of time. In an environment with little change (an office), that lease might be for a day or a week. In a changing environment (a college campus), the lease time is typically 60 to 90 minutes.



DHCP Settings

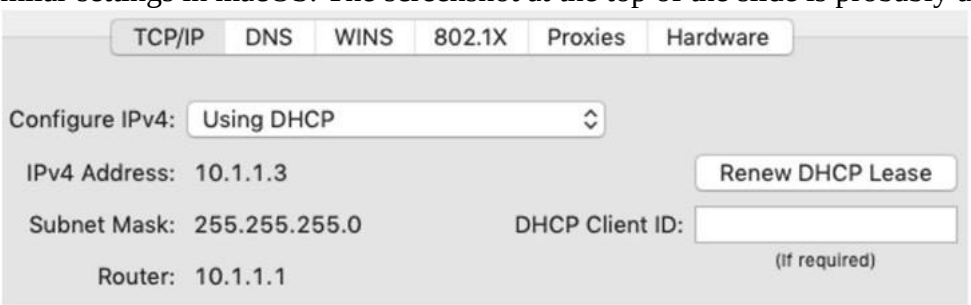
If you recall from earlier lecture, there are three pieces of information that you absolutely must have to communicate on a network: The IP address, the subnet mask, and the Default Gateway. You usually also require the IP address of a DNS server.

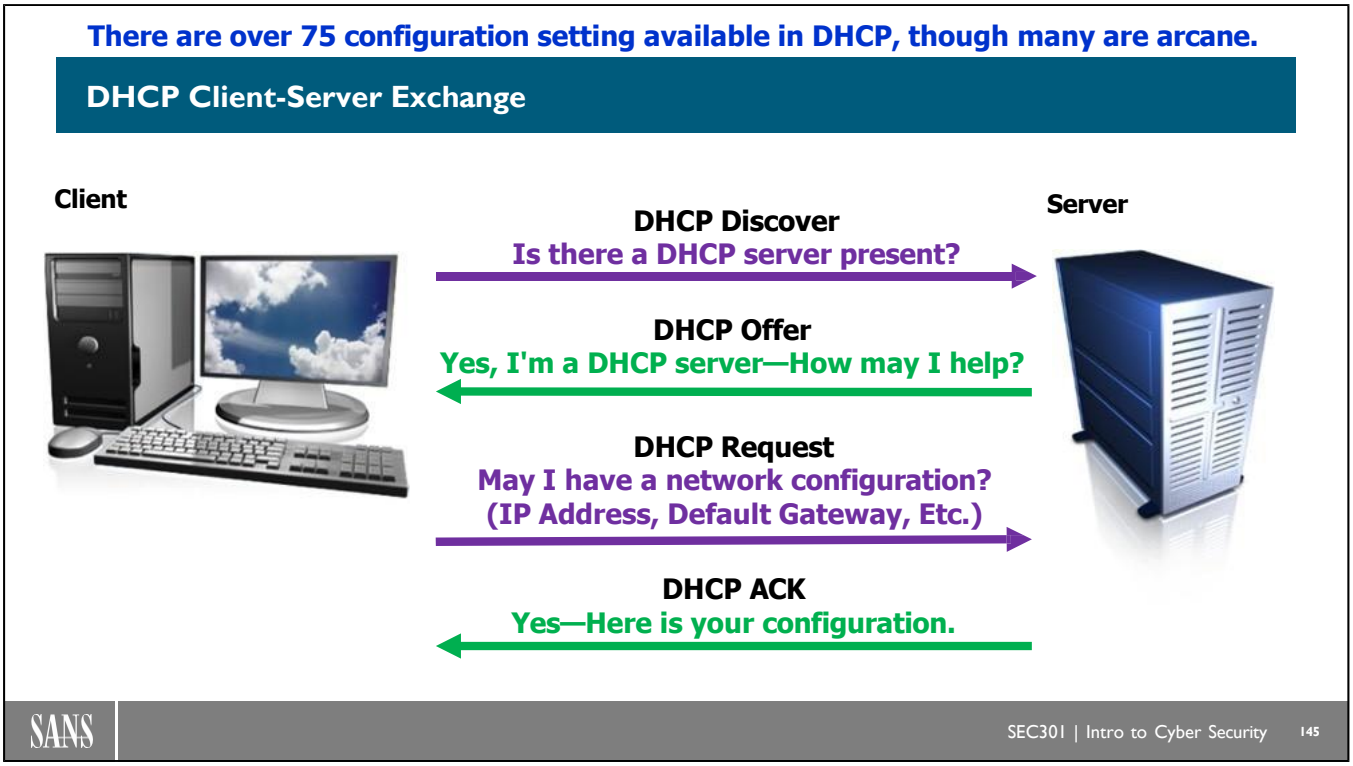
In the Windows operating system, you can select "Obtain IP address automatically" if you want to use DHCP to obtain a network configuration. Conversely, you can select "Use the following IP address" and then manually enter the IP, network mask, and Default Gateway if you want to use static addressing.

You can also choose to have the DHCP server assign the DNS servers that the computer will use for resolving IP addresses. As you can see on the right above, you can also statically enter the DNS server addresses.

Note that you can choose to statically enter the IP information, or the DNS settings, or both. You can mix and match these settings as needed.

There are similar settings in macOS. The screenshot at the top of the slide is probably a bit small…





DHCP Client-Server Exchange

As mentioned, the DHCP protocol is UDP-based. Meaning, DHCP is transported over the network using UDP.

Client DHCP traffic utilizes UDP Port 68 and Server DHCP traffic utilizes UDP Port 67.

The DHCP protocol exchange between client and server occurs in four (4) fundamental steps:

1. **DHCP Discover:** The client broadcasts on the local network segment asking if a DHCP server is present.
2. **DHCP Offer:** The server responds to the client broadcast by asking what type of configuration information the client requires.
3. **DHCP Request:** The client informs the server of what configuration information it requires. At minimum, this will always include at least an IP address, a network mask, and a Default Gateway.
4. **DHCP ACK:** The server sends the client the necessary configuration information.

As stated, the configuration information such as the IP address is leased to the client for a configurable period of time. Thought does need to be given to lease lengths as the correct lease length varies in different situations.

Per Verizon: At the end of Q1, 2021 there were 351.8 million public domains – Up 3.1 million in that Qtr.

Domain Name System (DNS)

- Protocol for resolving a domain name (that is, sans.org) to an IP address
- Utilizes a hierarchical system server
- Top-level domains (TLDs):
  - .com, .org, .edu, .gov, .biz, .info, .net, .int, .mil, .name, .pro, .aero, .coop, and .museum
  - Country codes (.uk, .fr, .au, .be, .ru, .ca, etc.)

The full list of TLDs is at: https://www.icann.org/resources/pages/tlds-2012-02-25-en
The full list of country codes is at: https://www.worldstandards.eu/other/tlds/

Domain Name System (DNS)

The Domain Name System (DNS) has a simple purpose. It is a protocol designed to take a domain name such as sans.org and resolve it to the correct IP address. It is a necessary protocol because you can't put a domain name such as sans.org into the destination IP address of a packet; you must have the IP address for that domain.

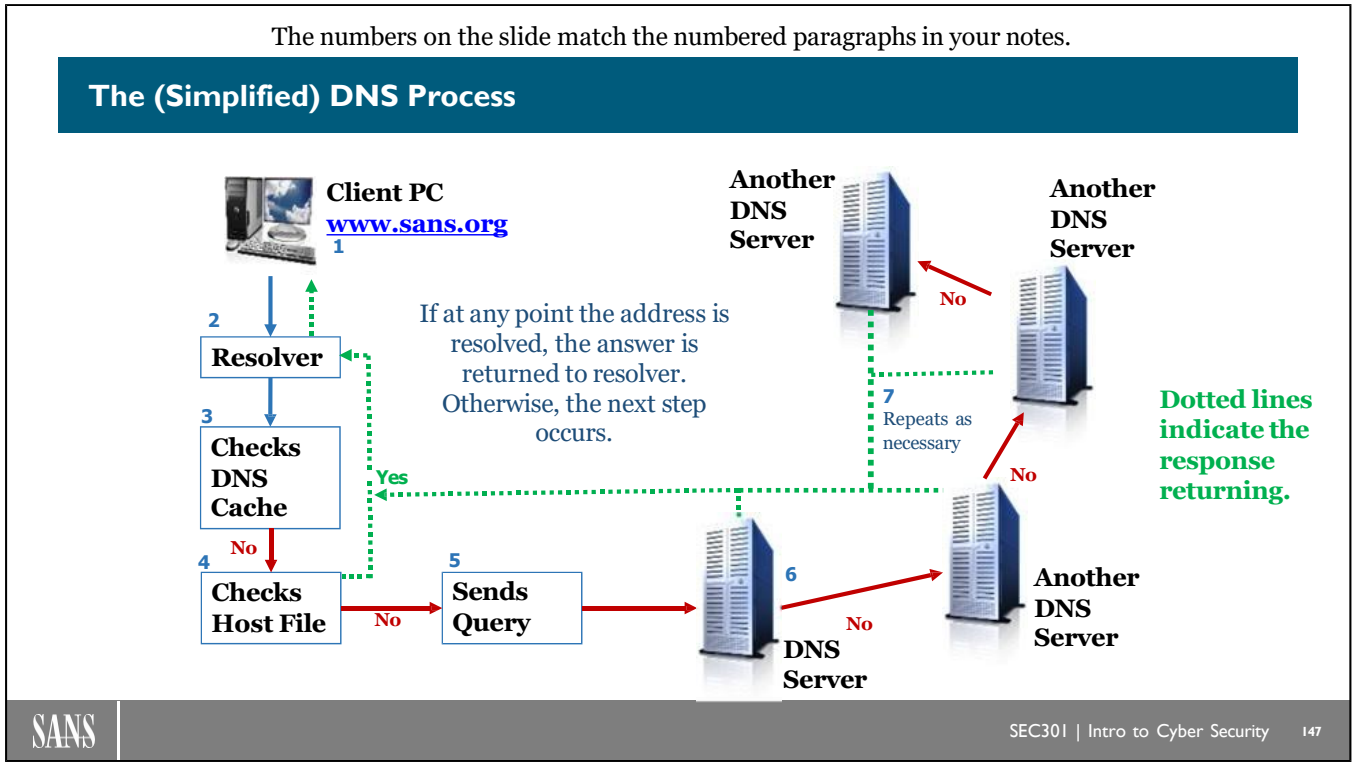
DNS is an amazing thing: It is the largest distributed database ever created. It is maintained by thousands of people around the world, most of whom have never spoken to each other before. Yet, when you need an IP address for a particular domain, it gives you one (and almost always the right one). The purpose of DNS is simple, but getting all that accomplished in a way that is transparent to use by the billions of internet users is *anything but simple*. This is especially true given that, according to Verisign, there are over 342.4 million domains registered around the globe as of the end of Q3 2018, an increase of 2.6 million in that quarter. There are a LOT of domains and associated IP addresses for DNS to keep track of and the number is growing steadily.

Everyone is familiar with the .com top-level domain (TLD), and many know about .mil, .gov, .edu, and others. What many people are not aware of is that there are actually now more than 1,100 such top-level domains listed at the following ICANN link.

References

https://www.verisign.com/en_US/domain-names/dnib/index.xhtml
https://www.icann.org/resources/pages/tlds-2012-02-25-en
https://www.worldstandards.eu/other/tlds/

The numbers on the slide match the numbered paragraphs in your notes.



The (Simplified) DNS Process

There are several steps in the DNS process. Exactly how many steps there are depends on how quickly an IP address is found for the domain name (called a *resolution*). The process looks something like this:

1. The user types a domain name into a browser for instance. The browser cannot create a packet containing a domain name, so the domain must be resolved to an IP address.
2. The browser calls a background application called Resolver. You never actually see Resolver run because it is a background application, but it is part of your PC. Resolver's only job is to resolve a domain name to an IP address.
3. Resolver first checks the DNS cache in memory to see if the domain has been resolved in the last few minutes. If it has, then the IP address is passed back to the browser. If not, then…
4. Resolver checks the local host file to see if the IP address can be resolved there.
5. Resolver sends the query to the IP address of the DNS server in the PC's network configuration settings.
6. If that DNS server does not know the answer, the query can be passed to another server and another server and so on. Eventually, a server in the DNS hierarchy knows the answer and it is passed back to resolver.

In each step, if the IP address for the domain is discovered, the IP address is passed back to the browser and the process ends; otherwise, the next step in the process takes place.

Note: Steps 1 through 4 happen inside the local PC. Only steps 5 and 6 involve external servers.

Links to instructions are in your notes…

Alternate DNS Servers

- Your ISP will assign you DNS servers via DHCP
  - You do not have to use theirs—you can use anybody's
  - Cloudflare: 1.1.1.1 & 1.0.0.1
  - Google DNS: 8.8.8.8 & 8.8.4.4
  - OpenDNS: 208.67.222.222 & 208.67.220.220
  - Verisign DNS: 64.6.64.6 & 64.6.65.6
- "Family Friendly" - Will not resolve adult sites, etc.
  - OpenDNS FamilyShield: 208.67.222.123 & 208.67.220.123
  - CleanBrowsing: 185.228.168.9 & 185.228.169.9
  - AdGuard: 176.103.130.132 & 176.103.130.134 (does not resolve Internet Ad sites)

Alternate DNS Servers

While your Internet Service Provider (ISP) will assign you a DNS server using DHCP, you do not have to use it. You can, in fact, use any DNS server in the world. While the one assigned by your ISP will certainly get the job done, many people prefer to use other DNS servers. In some cases, they feel they are faster. In other cases, they like features such as "family friendly" servers that will not resolve IP addresses for adult content sites and so on.

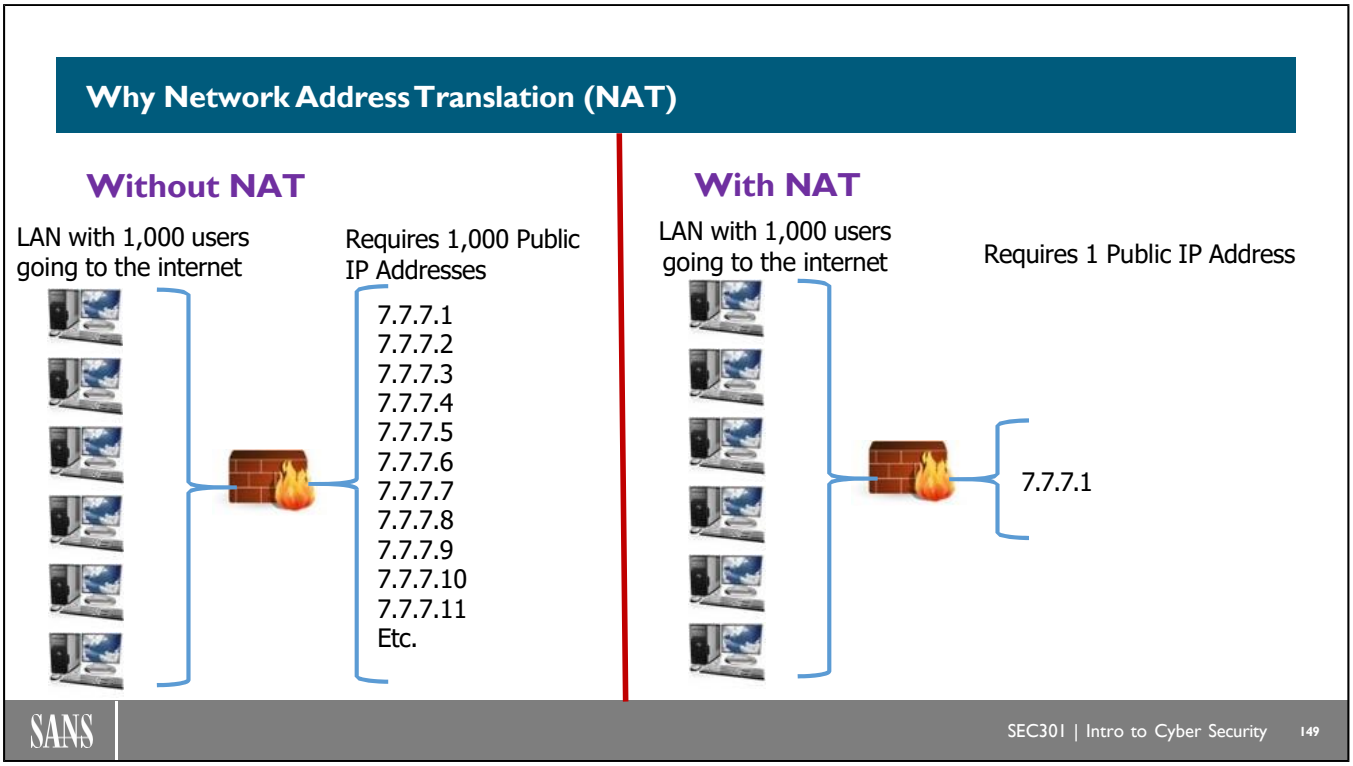
All of these DNS providers are completely free to use and all include instructions for setup (you simply put their IP address in your static DNS settings, or configure it into your DHCP server for automatic DNS address assignment).

- Cloudflare: https://1.1.1.1/
- Google DNS: https://developers.google.com/speed/public-dns/
- OpenDNS: https://www.opendns.com/
- Verisign DNS: https://www.verisign.com/en_US/security-services/public-dns/index.xhtml

Family Friendly

- OpenDNS FamilyShield: https://www.opendns.com/setupguide/#/familyshield
- CleanBrowsing: https://cleanbrowsing.org/
- AdGuard: https://adguard.com/en/adguard-dns/overview.html (This service also does not resolve internet advertisement sites.)

Note that many of these services have Enterprise level DNS for use at work.



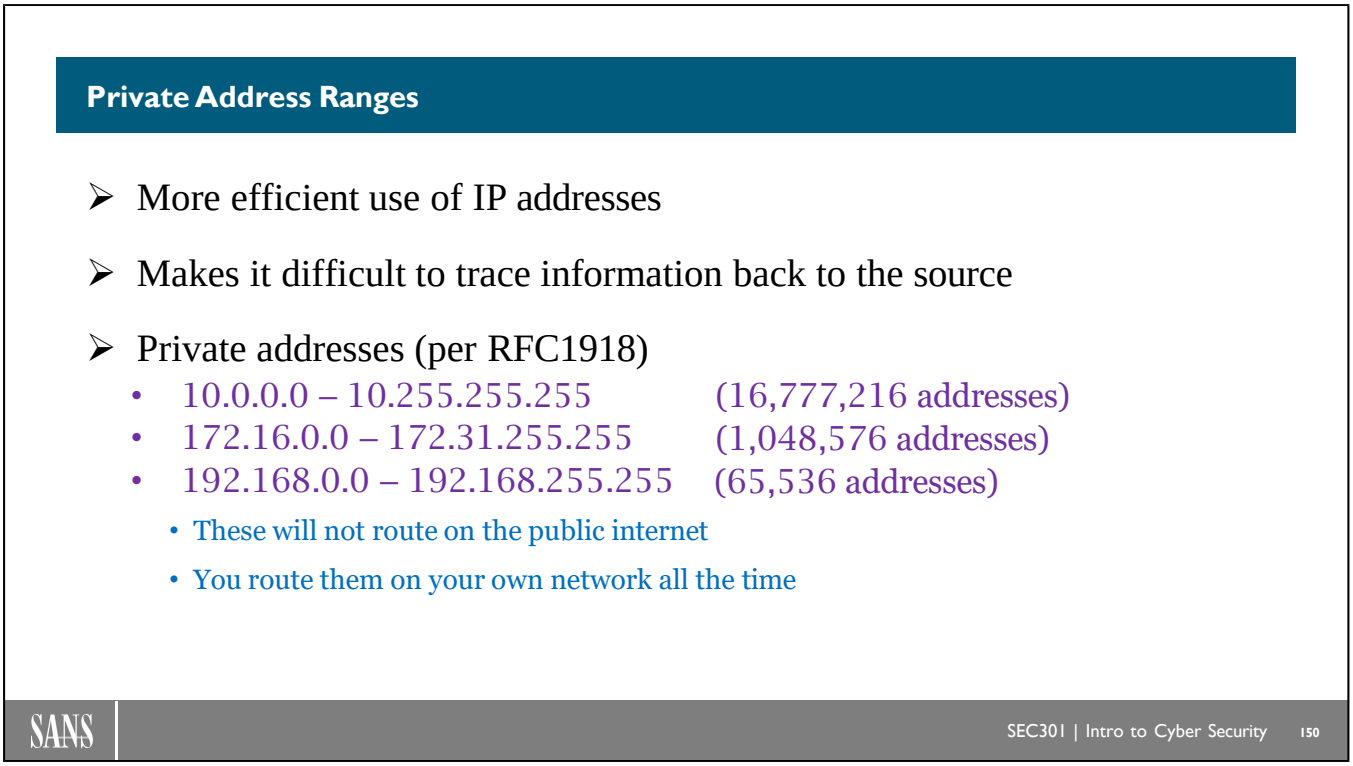
Why Network Address Translation (NAT)

Before we explain how Network Address Translation (NAT) works, it makes sense to tell why we need it. The answer is that we needed to make more efficient use of the available IP address space. You may recall that in the early days of the ARPANET, IP address assignment was hugely inefficient. You may also remember that, with a 32-bit IP address, you have just under 3.4 billion IP addresses, but there are over 4 billion internet users. All of this together means that for as long as people and companies use IPv4, NAT will be an absolute requirement.

As depicted in the slide above, without NAT, you would have to have a public IP address assigned to your organization for each internal user that needs to communicate on the internet. With Network Address Translation, you can have 1,000 internal users (or 10, 20, 30 thousand, or whatever) and still only require a single public IP address.

This is because as traffic leaves your internal network destined to the internet, the packets are edited so that they come from a single source IP address. Of course, this means that all responses go back to that single source IP address and the packets have to be edited back to their original source IP address.

How could this happen? Let's find out on the next few pages…



Private Address Ranges

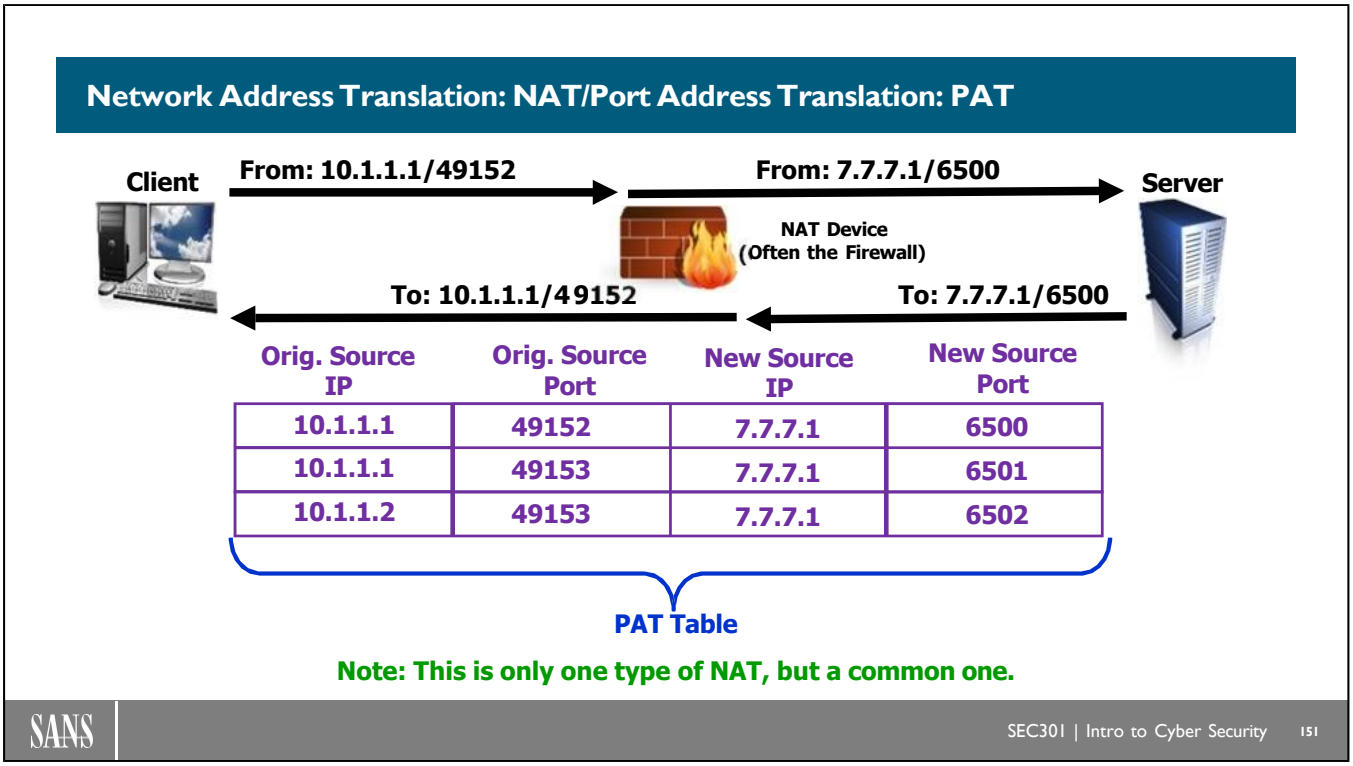
Sometimes, it is not desirable or advisable to use an IP address that is publicly known. First, your network might not be connected to the internet and, therefore, does not need to route information to and from other networks. A second reason is that you might need a larger set of addresses than the amount your ISP assigns to you. A third reason might be that you need to connect and share information with the internet, but you do not necessarily want anyone to know the IP addresses your network uses. All these conditions can be handled using a technique known as *private addressing*. It is described in detail in RFC 1918, "Address Allocation for Private Internets."[1]

RFC 1918 identifies three blocks of addresses for private internets (for example, designated as non-routable). These are shown on this slide and identified by their respective subnet mask slash notation (8, 12, and 16). In pre-CIDR notation, the first is a Class A block, the second is 16 contiguous Class B blocks, and the last is 256 contiguous Class C blocks.

Routers on the internet do not forward packets destined for these addresses. (We route them on our internal networks all the time.) Non-routable addresses, also called private addresses, can be useful for more efficient and secure address use through a technique known as *address translation* that is discussed in the following slide.

Reference

[1] http://tools.ietf.org/html/rfc1918



Network Address Translation: NAT/Port Address Translation: PAT

Network Address Translation can be accomplished in several different ways. One of the most common methods used today is called Port Address Translation (PAT). Note that this type of NAT can be used only for traffic from an internal network to the internet. It does not work for incoming traffic.

To gain an understanding of how PAT works, let's take the example of a TCP session going from a PC on our internal network to a server on the internet:

- The internal client sends the initial SYN pack of the TCP Session. That packet (in this example) has a source IP address of 10.1.1.1 and a source port number of 49152.
- As the packet traverses through the NAT Device, its configuration instructs it to edit the source IP address and source port number to new values.
- The Device chooses a new source port number. It then edits the source port number to the port it selected (6500 in this example).
- It also edits the source IP address to the public assigned IP address (7.7.7.1 in this example).
- The NAT device places the original IP and port number along with the new IP and port number into the four required fields of a PAT table, as you can see in the diagram.
- The packet goes on to the server, which ALWAYS responds back to the source IP and port number of the incoming packet.

- When the response arrives at the NAT device, that device looks at the PAT table and sees the entry created earlier.
- It then edits the destination IP and port number back to the original values (10.1.1.1 and 49152 in this case) and the packet continues to the internal client.

Note that for this method of address translation to work, every entry in the PAT table must be unique. If the same 10.1.1.1 machine generates another simultaneous connection, the original IP address will be the same, but it will have a different port number, so the NAT device picks a different port number for the PAT table entry. Likewise, if the machine at 10.1.1.2 happened to create a connection using the same port number as the 10.1.1.1 machine, the port numbers would differ and therefore the PAT table entry would be different as well.

Lab Time

- LAB 2.2: Networking

- Objectives
  - Find a Windows computer's network configuration
  - View the ARP and DNS caches
  - View your public IP address both with and without NAT enabled

  - Estimated completion time: 25 minutes

This page intentionally left blank.

SANS

Computer Functions and Networking

End Day 2

This page intentionally left blank.

This page intentionally left blank.

SANS

Review Questions

This page intentionally left blank.

SANS

How Computers Work Review Question

This page intentionally left blank.

Review Questions (1)

- A series of 4 bits are referred to as:
  A. A byte
  B. A half-byte
  C. A nibble
  D. A meal

- A series of 8 bits are referred to as (choose two):
  A. An octet
  B. A double nibble
  C. A byte
  D. A big nibble

This page intentionally left blank.

- Decimal is a base _____ numbering system.
  A. Sixteen
  B. Ten
  C. Five
  D. Two

- Binary is a base _____ numbering system.
  A. Sixteen
  B. Ten
  C. Five
  D. Two

This page intentionally left blank.

Review Questions (3)

- Hexadecimal is a base _____ numbering system.
  A. Sixteen
  B. Ten
  C. Five
  D. Two

- What is the place value of the high-order bit of a byte?
  A. Sixteen
  B. Sixty-four
  C. One hundred twenty-eight (128)
  D. One hundred seventy-two (172)

This page intentionally left blank.

- A byte of 00010001 translates to 0x___ ___
  A. 0x11
  B. 0x10
  C. 0xF2
  D. 0x01

- 0x22 translates to which binary string?
  A. 01000010
  B. 10000001
  C. 00100010
  D. 00000011

This page intentionally left blank.

SANS

Networking 101 Review Question

This page intentionally left blank.

- What is meant by the term Sneakernet?
  A. This is a nonsense term
  B. A stealth network for high security
  C. It is an extremely fast network using Quantum computers
  D. Delivering data on foot using USB drives, CDs, and more

- A network with a limited geographical area is called a
  A. LAN
  B. WAN
  C. CAN
  D. MAN
  E. PAN

This page intentionally left blank.

- What is a PAN?
  A. Pan-American Network spanning North and South America
  B. Principle Area Network spanning the primary area of operation for an organization
  C. Personal Area Network made up of very few computing devices—A very small LAN
  D. Panoramic Area Network—A very large LAN

- "Two or more computing devices connected together in some way to facilitate communication and exchange of data" defines what?
  A. A network
  B. An internet
  C. An Ethernet
  D. A cablenet

This page intentionally left blank.

- What is the transport mechanism of the internet?
  A. World wide web
  B. Fiber optic cable
  C. TCP/IP
  D. The internet does not need a transport mechanism

- What is another term for a group of internal subnetworks?
  A. Extranet
  B. WAN
  C. Internet
  D. MAN
  E. LAN

This page intentionally left blank.

- What is the most common physical topology on LANs?
  A. Mesh
  B. Star
  C. Ring
  D. Bus

- What is by far the most common protocol used to create LANs?
  A. Extranet
  B. WAN
  C. Internet
  D. Ethernet

This page intentionally left blank.

- What device inside a computer do you use to connect the computer to a network?
  A. Network Interface Card (NIC)
  B. Switch
  C. Router
  D. Hub

- What is a network device that forwards traffic based on a hardware address?
  A. Hub
  B. Modem
  C. Switch
  D. Router

This page intentionally left blank.

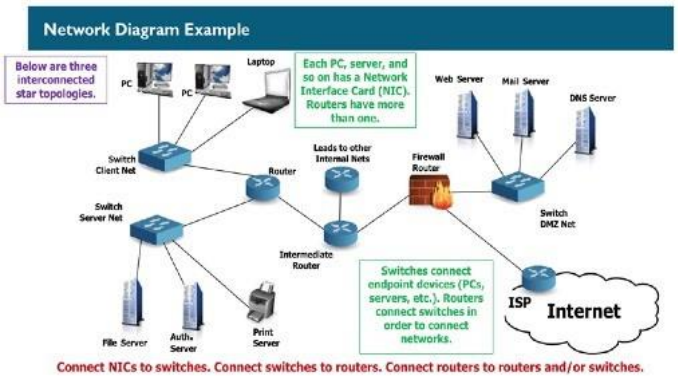
- What is a network device that forwards traffic based on IP address called?
  A. Router
  B. Switch
  C. Bridge
  D. Token Ring

- What do you connect NICs to?
  A. Routers
  B. Switches
  C. Servers
  D. Clients

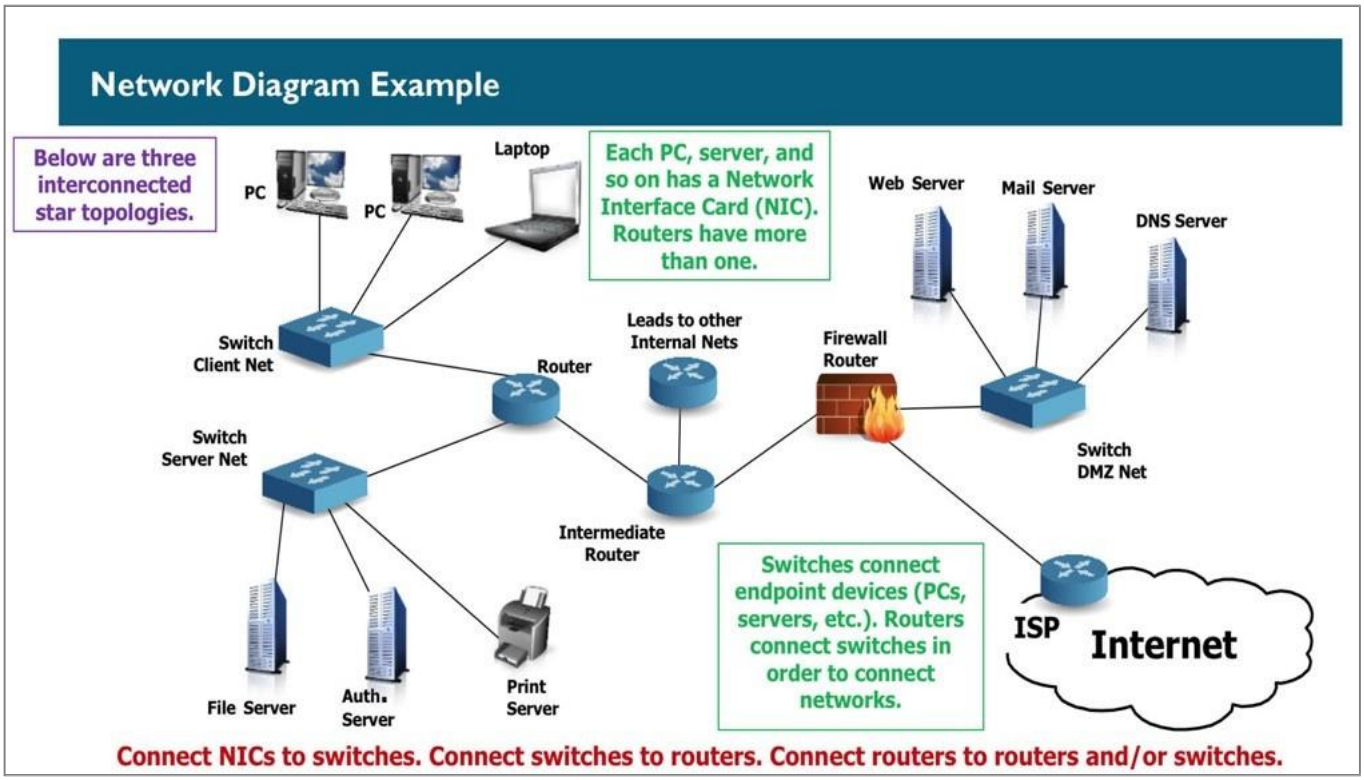
This page intentionally left blank.

- What do you connect switches to?
  A. Routers
  B. Servers
  C. Clients
  D. All of the above

- What do you connect routers to? (choose two)
  A. Routers
  B. Switches
  C. Servers
  D. Clients



This diagram intentionally enlarged! ;~)



- What is the name of the protocol that resolves IP addresses to MAC addresses?
  A. MAC Resolution Protocol (MRP)
  B. IP Resolution Protocol (IPRP)
  C. There is no need for such a protocol
  D. Address Resolution Protocol (ARP)

- How does a computer remember a MAC address once obtained?
  A. It does not remember that information
  B. Via the Routing table
  C. Via the ARP Cache in RAM
  D. It does not have to obtain that information

This page intentionally left blank.

- What do you call the rules governing communication?
  A. Manners
  B. Standard
  C. Protocol
  D. Framework

- When we *do* networking, we use TCP/IP. When we *discuss* networking, what do we use?
  A. STP (Star Topology Protocol)
  B. SQL (Structured Query Language)
  C. OSI (Open System Interconnect)
  D. Uh … TCP/IP, of course

This page intentionally left blank.

- What is the name of OSI Layer 3?
  A. Transport
  B. Session
  C. Network
  D. Link

- What is the name of OSI Layer 4?
  A. Application
  B. Presentation
  C. Internet
  D. Transport
  E. Network

This page intentionally left blank.

- The Network layer of the OSI model maps to which layer of TCP/IP?
  A. Transport
  B. Session
  C. Internet
  D. Link

- What is the name of the highest layer of both OSI and TCP/IP?
  A. Application
  B. Physical
  C. Presentation
  D. Internet
  E. Network

This page intentionally left blank.

SANS

Networking 102 Review Question

This page intentionally left blank.

- How many bytes are there in a MAC address?
  A. 48 bytes
  B. 48 bits
  C. 6 bytes
  D. 8 bytes

- What is the first field of an IP header?
  A. Destination address
  B. Source address
  C. Version number
  D. Time To Live

This page intentionally left blank.

Networking 102

Review Questions (2)

- What is the name of the internet protocol that deals with routing?
  A. Internet Protocol
  B. Ethernet Protocol
  C. TCP Protocol
  D. ARPANET Protocol

- How does IPv4 guarantee delivery of packets?
  A. By ensuring a reliable connection between communicating parties
  B. It doesn't—it is a "best effort" protocol
  C. Through Sequence and Acknowledgment numbers
  D. That is what DNS does

This page intentionally left blank.

- How many bits are there in an IPv4 IP address?
  A. 32 bits
  B. 128 bits
  C. 64 bits
  D. 48 bits

- What is writing an IP address in the notation 206.195.18.204 called?
  A. 32-bit notation
  B. Dotted-decimal notation
  C. Addressing notation
  D. It has no name; it's just how we write IP addresses

This page intentionally left blank.

- How long is a normal IP header (without options)?
  A. 20 bytes
  B. 48 bytes
  C. 20 bits
  D. 48 bits

- Why is "Version" the first field of an IP header?
  A. So network devices know the format of the header
  B. So the receiving device knows if the packet is fragmented
  C. So routers know where the packet is going
  D. So routers know where the packet came from

This page intentionally left blank.

- What does the Protocol field of the IP header tell us?
  A. It indicates the use of Ethernet on the LAN
  B. It indicates the application (OSI Layer 7) protocol
  C. It dictates if we will use ARP or Ethernet on the LAN
  D. What the next header in the packet will be (TCP, UDP, ICMP, etc.)

- What is the common name of the first router a PC sends a packet to?
  A. Router 1
  B. Destination router
  C. Default Gateway
  D. Source router

This page intentionally left blank.

- What is the name of the process that attempts to ensure packets go where they are supposed to?
  A. Nothing does that!
  B. Bridging
  C. Switching
  D. Routing

- When a router does not know where the destination IP address is, what does the router do with the packet?
  A. It discards the packet
  B. It broadcasts the packet out all interfaces in case a neighboring router knows what to do
  C. It forwards the packet to the router's Default Gateway
  D. That condition never occurs because of routing tables

This page intentionally left blank.

- Name a connection-oriented Layer 4 protocol:
  A. User Datagram Protocol (UDP)
  B. Connection Control Protocol (CCP)
  C. Transmission Control Protocol (TCP)
  D. Internet Control Message Protocol (ICMP)

- What does the routing table tell a router?
  A. Destination IP address of the packet
  B. Which interface the router should send the packet out of based on source address
  C. Routers don't use routing tables, switches do
  D. Which interface the router should send the packet out of based on destination address

This page intentionally left blank.

- Which TCP flag is part of the handshake?
  A. URG
  B. PSH
  C. RST
  D. SYN
  E. FIN

- Which TCP flag begins a gentle teardown?
  A. FIN
  B. CWR
  C. RST
  D. DIE

This page intentionally left blank.

- Which TCP flag terminates a session instantly?
  A. URG
  B. PSH
  C. RST
  D. SYN
  E. FIN

- Which IP header "Protocol" field entry indicates a TCP header will follow?
  A. One
  B. Six
  C. Seventeen
  D. Fifty-one

This page intentionally left blank.

Networking 102

Review Questions (10)

- How many steps are there in a TCP handshake?
  A. Five
  B. Four
  C. Three
  D. Two
  E. There is no TCP handshake

- How many steps are there in a normal TCP termination?
  A. Five
  B. Four
  C. Three
  D. Two
  E. There is no TCP termination

This page intentionally left blank.

- How many bytes are there in a UDP Header?
  A. Eight
  B. Twenty
  C. Thirty-two
  D. One hundred twenty-eight

- What is the name of the protocol that resolves domain names to IP addresses?
  A. Dynamic Host Configuration Protocol (DHCP)
  B. Domain Name System (DNS)
  C. Domain Control Protocol (DCP)
  D. Top Level Domain Protocol (TLDP)

This page intentionally left blank.

- What is a common method of hiding internal IP addresses?
  A. Network Address Hiding
  B. Anonymizer
  C. Network Address Disguise
  D. Network Address Translation

- Which of the following are private address spaces?
  A. 10.0.0.0 – 10.255.255.255 (16,777,216 addresses)
  B. 172.16.0.0 – 172.31.255.255 (1,048,576 addresses)
  C. 192.168.0.0 – 192.168.255.255 (65,536 addresses)
  D. There are no private addresses

This page intentionally left blank.

SANS

Answers to Review Questions

This page intentionally left blank.

SANS

How Computers Work: Review Question Answers

This page intentionally left blank.

- A series of 4 bits are referred to as:
  A. A byte
  B. A half-byte
  C. <u>A nibble</u>
  D. A meal

- A series of 8 bits are referred to as (choose two):
  A. <u>An octet</u>
  B. A double nibble
  C. <u>A byte</u>
  D. A big nibble

This page intentionally left blank.

- Decimal is a base _____ numbering system.
  A. Sixteen
  B. Ten
  C. Five
  D. Two

- Binary is a base _____ numbering system.
  A. Sixteen
  B. Ten
  C. Five
  D. Two

This page intentionally left blank.

- Hexadecimal is a base _____ numbering system.
  - **A. Sixteen**
  - B. Ten
  - C. Five
  - D. Two

- What is the place value of the high-order bit of a byte?
  - A. Sixteen
  - B. Sixty-Four
  - **C. One hundred twenty-eight (128)**
  - D. One hundred seventy-two (172)

This page intentionally left blank.

- A byte of 00010001 translates to ox___ __
  - **A. 0x11**
  - B. 0x10
  - C. 0xF2
  - D. 0x01

- 0x22 translates to which binary string?
  - A. 01000010
  - B. 10000001
  - **C. 00100010**
  - D. 00000011

This page intentionally left blank.

SANS

Networking 101
Review Question Answers

This page intentionally left blank.

- What is meant by the term Sneakernet?
  A. This is a nonsense term
  B. A stealth network for high security
  C. It is an extremely fast network using Quantum computers
  D. Delivering data on foot using USB drives, CDs, and more

- A network with a limited geographical area is called a
  A. LAN
  B. WAN
  C. CAN
  D. MAN
  E. PAN

This page intentionally left blank.

- What is a PAN?
  A. Pan-American Network spanning North and South America
  B. Principle Area Network spanning the primary area of operation for an organization
  C. Personal Area Network made up of very few computing devices—A very small LAN
  D. Panoramic Area Network—A very large LAN

- "Two or more computing devices connected together in some way to facilitate communication and exchange of data" defines what?
  A. A network
  B. An internet
  C. An Ethernet
  D. A cablenet

This page intentionally left blank.

- What is the transport mechanism of the internet?
  - A. World wide web
  - B. Fiber optic cable
  - **C. TCP/IP**
  - D. The internet does not need a transport mechanism

- What is another term for a group of internal subnetworks?
  - A. Extranet
  - B. WAN
  - C. Internet
  - D. MAN
  - **E. LAN**

This page intentionally left blank.

- What is the most common physical topology on LANs?
  - A. Mesh
  - **B. Star**
  - C. Ring
  - D. Bus
- What is by far the most common protocol used to create LANs?
  - A. Extranet
  - B. WAN
  - C. Internet
  - **D. Ethernet**

This page intentionally left blank.

- What device inside a computer do you use to connect the computer to a network?
  - **A. Network Interface Card (NIC)**
  - B. Switch
  - C. Router
  - D. Hub

- What is a network device that forwards traffic based on a hardware address?
  - A. Hub
  - B. Modem
  - **C. Switch**
  - D. Router

This page intentionally left blank.

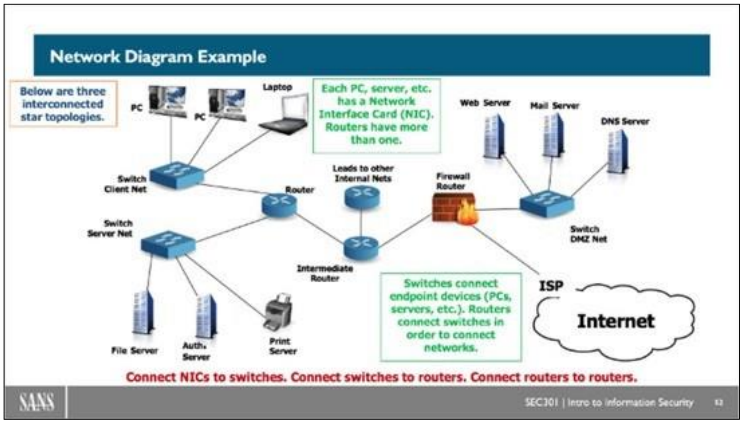
- What is a network device that forwards traffic based on an IP address called?
  - **A. Router**
  - B. Switch
  - C. Bridge
  - D. Token Ring

- What do you connect NICs to?
  - A. Routers
  - **B. Switches**
  - C. Servers
  - D. Clients

This page intentionally left blank.

- What do you connect switches to?
  - A. Routers
  - B. Servers
  - C. Clients
  - **D. All of the above**

- What do you connect routers to? (choose two)
  - **A. Routers**
  - **B. Switches**
  - C. Servers
  - D. Clients



This page intentionally left blank.

- What is the name of the protocol that resolves IP addresses to MAC addresses?
  - A. MAC Resolution Protocol (MRP)
  - B. IP Resolution Protocol (IPRP)
  - C. There is no need for such a protocol
  - **D. Address Resolution Protocol (ARP)**

- How does a computer remember a MAC address once obtained?
  - A. It does not remember that information
  - B. Via the Routing table
  - **C. Via the ARP Cache in RAM**
  - D. It does not have to obtain that information

This page intentionally left blank.

- What do you call the rules governing communication?
  A. Manners
  B. Standard
  C. Protocol
  D. Framework

- When we *do* networking, we use TCP/IP. When we *discuss* networking, what do we use?
  A. STP (Star Topology Protocol)
  B. SQL (Structured Query Language)
  C. OSI (Open System Interconnect)
  D. Uh … TCP/IP of course

This page intentionally left blank.

- What is the name of OSI Layer 3?
  A. Transport
  B. Session
  C. Network
  D. Link

- What is the name of OSI Layer 4?
  A. Application
  B. Presentation
  C. Internet
  D. Transport
  E. Network

This page intentionally left blank.

- The Network layer of the OSI model maps to which layer of TCP/IP?
  A. Transport
  B. Session
  C. Internet
  D. Link

- What is the name of the highest layer of both OSI and TCP/IP?
  A. Application
  B. Physical
  C. Presentation
  D. Internet
  E. Network

This page intentionally left blank.

SANS

Networking 102
Review Question Answers

This page intentionally left blank.

- How many bytes are there in a MAC address?
  - A. 48 bytes
  - B. 48 bits
  - **C. 6 bytes**
  - D. 8 bytes

- What is the first field of an IP header?
  - A. Destination address
  - B. Source address
  - **C. Version number**
  - D. Time To Live

This page intentionally left blank.

- What is the name of the internet protocol that deals with routing?
  - **A. Internet Protocol**
  - B. Ethernet Protocol
  - C. TCP Protocol
  - D. ARPANET Protocol

- How does IPv4 guarantee delivery of packets?
  - A. By ensuring a reliable connection between communicating parties
  - **B. It doesn't—It is a "best effort" protocol**
  - C. Through Sequence and Acknowledgment numbers
  - D. That is what DNS does

This page intentionally left blank.

Networking 102

Review Questions (3)

- How many bits are there in an IPv4 IP address?
  - **A. 32 bits**
  - B. 128 bits
  - C. 64 bits
  - D. 48 bits
- What is writing an IP address in the notation 206.195.18.204 called?
  - A. 32-bit notation
  - **B. Dotted decimal notation**
  - C. Addressing notation
  - D. It has no name; it's just how we write IP addresses

This page intentionally left blank.

- How long is a normal IP header (without options)?
  - **A. 20 bytes**
  - B. 48 bytes
  - C. 20 bits
  - D. 48 bits
- Why is "Version" the first field of an IP header?
  - **A. So network devices know the format of the header**
  - B. So the receiving device knows if the packet is fragmented
  - C. So routers know where the packet is going
  - D. So routers know where the packet came from

This page intentionally left blank.

Networking 102

Review Questions (5)

- What does the Protocol field of the IP header tell us?
  - A. It indicates the use of Ethernet on the LAN
  - B. It indicates the application (OSI Layer 7) protocol
  - C. It dictates if we will use ARP or Ethernet on the LAN
  - **D. What the next header in the packet will be (TCP, UDP, ICMP, etc.)**

- What is the common name of the first router a PC sends a packet to?
  - A. Router 1
  - B. Destination router
  - **C. Default Gateway**
  - D. Source router

This page intentionally left blank.

- What is the name of the process that attempts to ensure packets go where they are supposed to?
  A. Nothing does that!
  B. Bridging
  C. Switching
  D. Routing

- When a router does not know where the destination IP address is, what does the router do with the packet?
  A. It discards the packet
  B. It broadcasts the packet out all interfaces in case a neighboring router knows what to do
  C. It forwards the packet to the router's Default Gateway
  D. That condition never occurs because of routing tables

This page intentionally left blank.

- Name a connection-oriented Layer 4 Protocol:
  A. User Datagram Protocol (UDP)
  B. Connection Control Protocol (CCP)
  C. Transmission Control Protocol (TCP)
  D. Internet Control Message Protocol (ICMP)

- What does the routing table tell a router?
  A. Destination IP address of the packet
  B. Which interface the router should send the packet out of based on source address
  C. Routers don't use routing tables, switches do
  D. Which interface the router should send the packet out of based on destination address

This page intentionally left blank.

- Which TCP flag is part of the handshake?
  - A. URG
  - B. PSH
  - C. RST
  - **D. SYN**
  - E. FIN

- Which TCP flag begins a gentle teardown?
  - **A. FIN**
  - B. CWR
  - C. RST
  - D. DIE

This page intentionally left blank.

- Which TCP flag terminates a session instantly?
  - A. URG
  - B. PSH
  - **C. RST**
  - D. SYN
  - E. FIN

- Which IP header "Protocol" field entry indicates a TCP header will follow?
  - A. one
  - **B. six**
  - C. seventeen
  - D. fifty-one

This page intentionally left blank.

- How many steps are there in a TCP handshake?
  A. Five
  B. Four
  C. Three
  D. Two
  E. There is no TCP handshake

- How many steps are there in a normal TCP termination?
  A. Five
  B. Four
  C. Three
  D. Two
  E. There is no TCP termination

This page intentionally left blank.

- How many bytes are there in a UDP Header?
  - **A. eight**
  - B. twenty
  - C. thirty-two
  - D. One hundred twenty-eight

- What is the name of the protocol that resolves domain names to IP addresses?
  - A. Dynamic Host Configuration Protocol (DHCP)
  - **B. Domain Name System (DNS)**
  - C. Domain Control Protocol (DCP)
  - D. Top Level Domain Protocol (TLDP)

This page intentionally left blank.

- What is a common method of hiding internal IP addresses?
  A. Network Address Hiding
  B. Anonymizer
  C. Network Address Disguise
  D. Network Address Translation

- Which of the following are private address spaces?
  A. 10.0.0.0 – 10.255.255.255 (16,777,216 addresses)
  B. 172.16.0.0 – 172.31.255.255 (1,048,576 addresses)
  C. 192.168.0.0 – 192.168.255.255 (65,536 addresses)
  D. There are no private addresses

This page intentionally left blank.

Many of the slide graphics in this course are provided through a royalty-free license with PresentationPro. http://www.presentationpro.com/