Network Concepts and Protocols

by Ross Bagurdes

Networking concepts can be overwhelming and require a precision with language to be well understood. This course will teach you the fundamentals of data networking in a language accessible to a novice technical user.

Course Overview

Course Overview

Hi everyone. My name is Ross Bagurdes, and welcome to my course, Network Concepts and Protocols. I'm a network engineer with more than 20 years experience building enterprise networks and teaching people about them. Over the last 10 years, technology has moved from our desks to our pockets and has grown at an exponential rate. All this means we need talented IT professionals to keep data networks operating so we can surf the web on our smartphones and other devices. In this course, I will introduce you to the fundamental concepts of data networking operation including IP addressing and subnetting, the OSI model, protocols and their port numbers, cloud computing concepts, and other foundational networking concepts and terms. By the end of this course, you will understand how clients and servers encapsulate messages to communicate with each other, as well as how to identify the major components of an IPv4 and IPv6 address. Since this is an introductory course, you only need an open mind and an interest in learning how data networks operate. This course is part of a five-part series, and you should feel confident moving on to the second course of the series, Introduction to Enterprise Network Infrastructure, upon completing this course. I hope you'll join me on your journey to learn networking with the Network Concepts and Protocols course at Pluralsight.

Networking Concepts

Introduction

Hi everybody. I'm Ross Bagurdes, and welcome to my course, Networking Concepts, for the CompTIA Network+ N10-008 exam. Now, this is the first course of five that will help prepare you for the CompTIA Network+ exam. However, you do not need to be preparing for the exam in order to really appreciate these courses because what we're going to do is introduce networking, talk about all of the critical terminology, and make sure you walk away from this course and the rest of them with a really solid understanding of data networks so that you can use that to both better your career or help understand what's happening inside of a data network better. Our goals for this module will first be to define what is data networking, and I really want to give us a foundation so that we understand what we're talking about here. Next, I want to talk about communication processes. I want to go through an example of using a smartphone to call a friend of yours so that we can see that there's a pretty complex system happening in order for that communication to occur. And then last, this is not for everybody, but if you are studying for the Network+ exam, I'm going to review where you can download the exam objectives and follow along with those throughout these videos. If you're not taking the exam, don't worry about it. You can skip that part.

What Is Data Networking?

So, what is data networking? And if I think about it in the most simple sense, if you've never really thought about what is data networking before, you might think of data networking as these devices in your house. These are cable modems. It could be a DSL modem; it could be a fiber optic modem. But these devices, typically anybody that has an internet connection is likely to have one of these devices in their house that they use to connect to the internet. So you might think of data networking as these devices. You likely have a device that looks similar to this as well. This is a wireless router, and this allows your smartphone, tablets, laptops, or other devices to communicate with the internet as well. However, networking is much bigger than just those home devices. Inside of a data center or a business or a networking closet, you may see some switches like this with lots of bundles of cables plugging into these devices in order to provide network connectivity. Not all of them are nice and neat and pretty like this one. Some of them get a lot messier, like this one here, where it's just a bird's nest of wires. But both of these systems are doing the exact same thing, which is to provide networking to devices, typically, in this case, in an office. So now that we've seen some devices and equipment that look like data networking, let's talk more about what it actually is because that's just the gear to facilitate it. Data networking is actually more about transferring information. So here what I have is, this is an icon of a server, and a server typically is connected to a network, and servers then hold data. And this particular server, I'm imagining, is the Wikipedia server, which contains an article on computer networks. And what I want to do as an end user is I want to go onto my PC over there on the left, type in wikipedia.org, and then do a search for computer networks. So I do that, and what I end up doing is, I connect through the network which I've represented as this cloud. And from this point forward, anytime we see this cloud icon, that means that there is some data network involved there. There's a bunch of switches and routers and wires and fiber optics and wireless that are all working together to transfer this information. And I represent it with a cloud here because it's much simpler when we're just trying to focus on certain aspects of this process. So I go on my computer, I search for that Wikipedia article, it gets to the Wikipedia server, and then the server transfers it to my computer so I can then read it. And this really is what data networking is all about. It's some version of this exact process. So what is data networking? It's a system of hardware, software, and protocols used to move information from one device to another. That's really the crux of it. This is really what data networking is. It's about moving information with a very specific set of software, hardware, and protocols.

Communication Processes

So let's introduce data networking and some of the processes we use here by talking about something that we do relatively often, maybe not as often as we once did. Now we usually are texting our friends versus calling them, but let's just say here that in this example we have Leia over here on the left with her smartphone, and she wants to call her friend Obi over on the right side of the screen to ask him for help. Well, in order to do that, there's all kinds of processes that are happening in order to facilitate this, but if we think about it in some very simple terms, in order for her to contact Obi and ask him for help, she has to go on her smartphone, open up the phone app, find Obi's contact, click on it. That will allow his phone to ring, he will answer and say hello, she'll say hello, and now Leia can ask for some help. And what's going to happen here is Leia is going to start talking, and when she talks, that's going to vibrate the air. And that vibrating air is what carries our voice. Now, if Obi we're just sitting right next to Leia, he could just sit and listen to the air vibrate his own eardrum, and there would be no need for any other equipment. However, since he's on the other end of this phone, what's going to happen is, Leia's voice is going to vibrate the microphone on her phone. There's a little membrane in the microphone. It's going to vibrate that membrane, which is going to create a little electrical signal inside the phone. What the phone will do then is it'll take that electrical signal, and it will convert it into a wireless signal that it can send up to the cell tower. Well, then the cell tower takes that wireless signal, and it's going to convert it again. So now Leia's voice was encoded in this wireless signal. That encoded voice is then going to get translated into a signal that can travel across these wires on the telephone poles. They might be fiber optic; they might be copper. But we can send that signal from one cell tower to another cell tower over that system. The next cell tower that's closest to Obi's phone will then convert that signal that came in over the wire back to a wireless signal, send that down to his phone. That signal will then get translated in his phone to some electrical pulses that vibrate the membrane of a speaker. That membrane vibrates the air and then Obi can hear those vibrations as Leia's voice asking for help. So when we're using our cell phones to communicate with each other, it's this constant process of translating our message from one type of signal to another type of signal to another type of signal without ever actually losing the data that we're trying to send. This right here is a very oversimplified version of what we're doing in data networking. Data networking does the same thing. We take messages, we encapsulate them inside of virtual envelopes, and then we send those envelopes across multiple different types of network, where they get translated again and again and again. And we're going to look at that throughout this entire series of courses, but all the terminology, language, protocols, and systems that are used in order to make that happen.

CompTIA Network+ Exam Information

The CompTIA Network+ exam. Here we're going to take a look at some exam objectives. Now, if you're not studying for the Network+ certification exam, no problem. Skip to the next module. There's nothing that is important for you here. If you are studying for this, what you may want to do is go download the exam objectives at this website. You can go to certification.comptia.org/certifications/network, and that will give you a list of the certification exam objectives. This is all the details of everything covered on that exam. It is broken into five domains, networking fundamentals, network implementations, network operations, network security, and network troubleshooting. Now, the courses that I have designed to prepare you for the certification are generally broken up exactly into the domains that you see here, with the exception of a few topics. I took some of the information out of the networking fundamentals, and I moved it into network implementations, just because it fit a little bit better in there. However, it's not super important that they fit neatly into these categories. It's more how you interpret and understand this information. So I'm trying to make this as easy as possible for you to understand. So if we actually go download these objectives, it comes in this somewhat lengthy document. Here's the front page of it. They have information about the exam. And here's that list of domains that I just went over. And then they go through each domain and they break down exactly what you need to know here. And this document is very thorough, very lengthy, and it covers all of the topics that you're going to need to know about. Now I'm not going to go through each of these exam objectives in this particular document. However, as we go through the course, you're going to learn about each one of these little snippets of information you see here. We're going to learn about all of those. It even has an acronym list at the end. There are tons of acronyms to learn in data networking. There's also hardware and software they talk about as well. So you can download this document for yourself and review it. As you're learning the exam, you can actually use this to check off the stuff that you feel like you know and the stuff that you need more help with.

Summary

So let's wrap up this first module here. What I did here was define what is networking. We looked at communication processes to see how a cell phone call to our friend transfers this encoded signal of our voice across multiple different media to reach our other friend's phone. And then we looked at the CompTIA Network+ exam objectives. We're going to go onto the next section where we look at modeling data networking so that we have a very precise system to understand the order of operations in data networks.

Using the OSI Model to Describe Network Operations

Introduction

In this next module, we're going to be using the OSI model to describe network operations. Our goal here is going to be to introduce this OSI model to really hone in and identify the very specific components of network communication. So our goals for this module are going to be to introduce the OSI model, remind ourselves about that telephone call between Leia and Obi, and then use that to jump into modeling networking with the OSI model.

The OSI Model

So the OSI model stands for Open Systems Interconnect. It's a model that was developed in the 70s in order to describe network operation. Now there's lots of different protocols involved in data networking, and what the OSI model did is it gave us a place to categorize each of these protocols, as well as give the exact order that those protocols need to be processed in. So if you remember this phone call I talked about in the previous module, the order of operations here were such that Leia had to vibrate the air, which vibrated the microphone, which converted it into electrical signals, which then got converted into wireless signals, which got converted into a different type of electrical signal, and so on, until that message reached Obi on the other side. Now, in order for that to happen, there is a very precise set of rules that need to happen and protocols that need to be followed in order for this conversation to successfully happen. Well, when we're working with data networking, it's almost exactly the same; we just have these different components involved versus our smartphones and our voice. We're going to be using a computer or a tablet or a smartphone, like maybe an internet browser on a smartphone, to do our communication. So let's introduce a drawing. Now I communicate a lot in these drawings with icons, so I'll introduce the drawings and all the icons we're using so that we can use that then to talk about the OSI model. So here we have a PC. This is our end-user device. Sometimes this is called a client, sometimes it's called a workstation, PC. That's the device we're going to be using to surf the web. Now another device in our network, especially in our home network, is likely going to be this wireless router device. And this is the icon, the purple icon here is the icon that I'm using for a wireless router. And we can then connect our PC up to that wireless router. That router typically has some switchports on the back, so we can actually plug in a cable to that. Or we might be using it as a wireless device and actually not have any cables at all and just use the wireless communication to communicate between our workstation and the wireless access point. Now in order to get access to the internet, we need some type of device to bridge the network connection between our wireless access point and the internet, and that's where a modem comes in. So here is a modem. I showed you some real-life pictures of this in the previous module. Here this teal icon is just a representation of that cable modem or any other type of modem that's going to connect you to the internet. So we're slowly building this network up. This is a representation of your home network. And then out on the internet, we have a bunch of servers out on the internet, and this icon is representing the servers, the orange icon there with the globe next to it. This is going to be the servers that contain all of the information that we are going to transfer from the server to our workstation so that we can read the news, surf Facebook or other social media, or maybe check our email. For the purposes of this, I'm going to avoid the wireless for this moment.

We're going to come back and talk lots more about wireless in the future. But for now, I'm just going to connect my workstation to our network with a cable. So now we have this network, or this drawing of our basic network. Remember before I said that cloud represents a bunch of devices and protocols that we don't want to represent individually? And that's exactly what I'm doing here as well. That internet now is a cloud. It's thousands of devices that we're going to use to connect to our server to transfer information. So let's say that on my workstation I want to go to pluralsight.com. So I type that into my browser, and that's going to send a message to our Pluralsight server. The Pluralsight server is then going to grab the website that we're looking for, it's going to put that into a message, and then send it over to my workstation so that I can browse the videos on Pluralsight. So in order for that to happen, we need a very specific set of rules and orders of operations for this to happen. So if you start with the most basic things that we can see, we can see the cables that we're using. Now the cables don't have any special electronics in them, or at least most of them don't. They're typically just some wires with a special connector on the end so that we can plug them into things. But there's no special electronics in those devices. Those cables do follow protocols though. So the Ethernet cable that we use to connect our computer to our wireless access point there, or our switch, that cable follows a very precise set of rules on how it's constructed. It just doesn't have anything electronic in it. So all these cables that we have here are used to connect things together. There's even wireless cables. That might be our wireless connection in our home from our smartphone to the wireless access point. Or even out on the internet, there might be some connections that use a point-to-point wireless connection. These are also cables, in that they're used to transfer information, they follow a specific protocol, but it's not a device. It's actually the signal itself that we're talking about here. So in the case of the cables that connect our workstations together, these are called twisted pair cables. These twisted pair cables follow a very specific set of rules, which we'll learn about in a later module. Oftentimes connecting our server to the rest of the network we're going to use twisted pair cabling. Connecting our cable modem to the internet might use a coax cable. And then out on the internet here, we're likely going to be using fiber optics, or very thin strands of glass, in order to transfer our information. So all of these cables, these different cables, whether it be wireless, twisted pair, coax, fiber optics, all of these are technology that we use in order to transfer our data from one device to another. All of this stuff is happening at what we call the physical layer of the OSI model. The physical layer is also layer 1, and the physical layer is really what we're talking about when we're talking about cables, specifications for cables, whether that be wireless, copper, or glass. The next layer of the OSI model here is going to be used to have protocols that allow us to transfer bits of information over the cables. So there is a specific protocol called Ethernet that we use to transfer data from our workstation to the wireless router, from the wireless router to the cable modem, from the cable modem to wherever our internet provider is, from the server to our switches inside of our data center, and then numerous other protocols on the internet used to transfer information. So all these little circles that I have drawn here, those are representing different protocols that could be used to transfer data. And each one of these protocols is a little bit different, but they work together all at the same level. What I have circled here, the connection between my PC and the router, the router and the cable modem, and then the server and the rest of the internet, that's likely using Ethernet. The connection here between our cable modem and the rest of the internet is likely using a protocol called DOCSIS 3. Out on the internet it's mainly Ethernet, but there could be other protocols out there. All of these protocols, Ethernet, DOCSIS, are all part of the data link layer, and they provide all of the structure to be able to use the cables that are connected to it. So they provide the protocols and the rules, specifications for electronics, as well as how to create and move a message across those links. So that's layer 2 of our OSI model, is the data link layer. So the data link layer is facilitating communication within these circles that I have drawn on the screen. It's allowing us to pass data from one device to another. However, if I want to send a message to a device that's far away, that's not on my little local network, what I need to do is I need some other mechanism to do that. So I need another type of protocol in order to do this, so I might need to communicate between my PC and my cable modem, or I might need to communicate from the cable modem out to the internet someplace, or I might need to communicate, in this case, I will need to communicate, from my workstation all the way to Pluralsight servers and back again. So I need some way of sending a message from my workstation to Pluralsight and then getting it back, and this is where we use a protocol called Internet Protocol, and we use something called IP addressing. The IP addressing allows us as users to communicate with nearly any device on the internet. This is where IP addressing occurs, IP routing, and it is all the network layer, which is layer 3 of our OSI model. So far we have the physical layer, our cables, we have the data link layer, which are the protocols that allow our computer and other devices to communicate locally with other devices. Then we have layer 3, the network layer. That allows us to make use of the data link layer and physical layer to send messages long distances across a network. The next layer here we're going to talk about is, in order for us to have a conversation between our client, or workstation, and our server, where pluralsight.com's website is hosted, we need a way to build a session in between these two devices so that they can send information between each other and understand that the conversation is meant for specific things in that network communication. We're going to talk lots about this process. In fact, I have a whole other module that talks specifically about this process. But for now, let's go back to our cell phone example here, where we're making a phone call. Now we talked about kind of like the vibrating air and whatnot with this, but more so there's another way to look at this conversation. There's another protocol we follow, actually, in order to make this conversation work. I can't just pick up my smartphone and start talking and expect my friend Obi on the other end to listen. What I have to do is I have to dial a phone number for Obi, and then I have to wait for Obi's phone to ring. Once his phone is ringing, Obi can answer that phone, he's going to say hello, Leia here says hello, and once that happens then Leia can say, hey, I need some help. And Obi can say, oh, okay, and they can have that conversation. And my point here is that in order for Leia to be able to communicate with Obi, we have to go through that process of dialing the phone, waiting for it to ring, waiting for Obi to say hello, I say hello. And once that happens then I can send any information I want. Transmission Control Protocol does this in data networking. This is also called TCP, and what it does is it sets up a session between our workstation and the server so that we can send data between these two devices. This is the transport layer. So the transport layer in networking is responsible for setting up a session typically using TCP. It has a handshake process, and it builds a connection, and it uses the network layer to find where in the world those devices are.

It uses the data link layer to communicate those messages from device to device to device to device until they reach their destinations, and that all happens on the physical layer, which is our wires and cables that connect our network together. Next, we need a way to transfer information that's in a usable format. One of those ways is to use a web browser and browse to a website like pluralsight.com and then retrieve a web page. The website is nothing more than a document with information in it. The web browser is an application that makes use of a protocol in order to request and transfer information. So in order to get the website to our workstation, we're going to make a request of the server to say, hey, send us the website. The server's going to respond with a document that is the website, and we're going to use a protocol called Hypertext Transfer Protocol. This is also known as HTTP, and it has a sibling called HTTPS, which is the encrypted version of it. Now, HTTP and HTTPS, they transfer documents that are written in a language called HyperText Markup Language, or HTML. So we're using HTTP to transfer HTML documents. So our website is written in some form of HTML, and we use HTTP or HTTPS to transfer those. So all of this is happening outside of our process of building a session between our two workstations or identifying the location of those devices with IP addresses or having protocols that allow us to transfer information from our computer to the router to the cable modem and so on. All this is happening at the application layer of the OSI model. Now the OSI model is seven layers, and you've noticed that I have skipped two of them. The OSI model was written in the 70s when they had different networking needs and other protocols involved here. So there's two layers that I have not talked about now, 5 and 6. So what's going on with that? Well, let's talk about it. First, let's talk about the presentation layer. The presentation layer is layer 6, and we don't really use the presentation layer anymore. And there's a reason for that. Back in the 70s, there were several types of systems. There were a lot of open source systems that universities used, and then IBM had systems as well, and IBM developed very proprietary protocols that were similar but different than the ones used in university settings. So let's say we have this message of "Don't Panic." Well, in order to create Don't Panic in a language that the computer understands and can encode, when we type on the keyboard, each one of these letters capital D, lowercase o, apostrophes, and whatnot, each of those is mapped to a hexadecimal value. It's a 8-bit hexadecimal value, and it's called ASCII, A-S-C-I-I, which stands for American Standard Code for Information Interchange. ASCII is still used today. When we type on our keyboard, it is still translated to these values in ASCII you. So if I translate, Don't Panic to ASCII, I get the D = 44, o is 6F. We're going to learn about hexadecimal numbers in another couple modules here, so don't stress too much if these numbers look goofy because they have letters in them. We're going to talk about exactly why that is and a little bit. Just know that the numbers that I'm showing you here are actually numbers and not some other goofy language. So D is 44, o is 6F, n is 6e, the apostrophe is 27, t is 74, and so on, and when we put that all together we get this string of numbers that represents Don't Panic, including the space in there. In IBM-land, they used a completely different number system in order to encode each key of the keyboard. So each keystroke that meant Don't Panic got encoded differently. And EBCDIC, E-B-C-D-I-C, which stands for Extended Binary Coded Decimal Interchange Code, is IBM's version. It was their proprietary version. And in order for a system that used ASCII to communicate with a system that used EBCDIC, we needed some layer of the OSI model to translate from one system to the other, and that's what we intended to do with the presentation layer. Now that has all become quite antiquated, and we really don't do much with this anymore. You may find in textbooks that they're going to say, oh, JPEG and MPEG and whatnot happen at the presentation layer, and that might be the case. But really, those are outside of the OSI model, and we're ultimately going to be using application layer protocols in order to transfer information, and we really don't worry too much about the presentation layer. The session layer is another layer. I personally don't concern myself with the session layer too much. There are some protocols that do operate at the session layer. However, for networking purposes, we don't stress too much about classifying something as a session layer protocol. Ultimately, we consider it to be part of the application layer.

Summary

Let's wrap up here what we've talked about. We've introduced the OSI model, we talked about the seven layers of it, we started by modeling that telephone call and thinking about how we had to vibrate the air and the entire process that needed to happen in order to make that phone call, as well as another idea of a protocol where we actually had to dial the phone, wait for it to ring, say hello before we could transfer messages between the two devices. And then we talked pretty in depth about all of the components of the OSI model and how they relate to networking. So remember, the OSI model gives us a very precise order of operations that we can work with. It gives us a space to categorize each protocol so that when we are working with networking, we understand which protocols are involved and why.

Encapsulation and the OSI Model

Introduction

In this next module, we're going to take a look at encapsulation and the OSI model. Now, we just talked about the OSI model and how each of the layers have a specific purpose. Now we're going to go be a little more technical about it and introduce how those layers are being used here to send data. So our goals for this module are, first, to review the OSI model, then to introduce the concept of encapsulation, and then I'm going to describe encapsulation layer by layer of the OSI model. So here's our OSI model, and remember, we're going to not worry about layers 5 and 6, we're just going to stick to the application, transport, network, data link, and physical. And as a reminder here, remember the application layer, we're typically dealing with a protocol that allows us to transfer specific types of data in a specific way. Our transport layer is going to set up a session between the two endpoints, between our client and our server. The network layer is going to give us the path that we can take through the internet or through a network to get from our client to our server. The data link layer is going to let us make the individual hops between our client and our switch, the switch and the router, the router and our cable modem, the cable modem and the internet, and so on. The data link layer is going to allow for that. And then the physical layer is going to be where we actually move the bits of information, either as an electromagnetic wave in the case of wireless or as a light signal in the case of fiber optics or as an electrical impulse in the case of a copper wire.

Encapsulation and the OSI Model

So, let's take a look at how this happens. So if we go to pluralsight.com on our workstation, we send that information out onto the internet, and then it transfers the website to our computer, and this all seems to happen magically, seamlessly, and typically very quickly. When we're actually doing this, though, what we're going to do is we're going to take that website, and this is our actual application layer information, in the case of going to a website, we're going to use a protocol called HTTP, or Hypertext Transfer Protocol. Websites are written in a language called HTML, or Hypertext Markup Language, and we use Hypertext Transfer Protocol, HTTP, to transfer it. So this application layer protocol, HTTP, is what we're using the transfer it. When we are working with this, though, these websites tend to be quite large, so we need to be able to break up that website into smaller chunks so we can successfully get it to the client. We're going to find out at the end of this encapsulation process that some protocols, specifically Ethernet, has a maximum amount of data that we can transfer for each frame that we send. For each chunk of data that we send across the network, we have a maximum amount of data that we can send in each one. So, once we have this application layer, it's going to work in conjunction with the transport layer to take that data, break it into smaller pieces, and then add it to a header. So we're going to put a header on this data at the transport layer. The transport layer, since it's setting up a session between our client and our server, we're going to have specific information in there to allow that to happen. In this case, it's a source port, a destination port number, some flags, which is just some general information about what's happening in the transaction, a sequence number, an acknowledgement number, and those keep track of how much data has been sent and received. This is kind of like if you're sending a birthday card to your dad. You would fill out the birthday card, you put any information in the card that you want, much like our website here for pluralsight.com, there's lots of websites out there and this particular one is

Pluralsight, so we can take any information we want and we take that card and we put it inside of an envelope and we seal it up, and then we write on the envelope the destination address, which is our dad's name and address, and then we put a return address on it, which is our name and address. So we're taking this information, we're putting it inside of an envelope. This particular envelope is the transport layer, and we call this envelope a segment. So any data that we have with a header at the transport layer, we call a segment. So a chunk of data with the transport layer header is a segment. This particular segment header, or transport layer header, is a TCP header, and we're going to learn more about TCP in another module in this course. For now, we just need to know that this information allows the client and the server to set up a session and keep track of what data has been sent and received. Now, that's just one component of this transaction. In order to know where we are sending this data, we need to tell it what the source and destination IP addresses are. So we need to know where on the internet or on a network the server and the client are. So we take our transport layer information, which is going to keep that session between our endpoints going, and then we send it down to the network layer, so we take our segment and it becomes the payload of our network layer, and then we add this header, we add the source IP, destination IP, a value called the TTL, or time to live, which tells it how far this packet can travel at a maximum distance, as well as some other information that we add into the network layer header to make this transaction work correctly. I don't go into too much detail on that because it's not important at this level of understanding of data networking. So now we have our segment inside of the payload of our network layer header. This is called a packet. So, any time we have some data that we're transferring with a network layer header, this is a packet. So it's a chunk of data with a network layer header. This particular one is an IP header, or Internet Protocol. Specifically, Internet Protocol version 4. And we're going to learn more about IP as we go throughout this course and throughout the series of courses that will help train you to get a better understanding of data networking. So our network layer header is going to allow us to know what two endpoints on the internet, or any network, we're going to send this information to. Now in order to get our packet to go from one device to the next, from our workstation to the switch, from the switch to the router, from the router to the cable modem, from the cable modem out to the internet, and all of the hops that go along the internet, we're going to need a data link layer header here. So we send our network layer packet down to the data link layer and we put it in a frame. Now a frame is just a chunk of data with a data link layer header. In this particular case, this is an Ethernet header, so this is going to be an Ethernet frame, and we'll often use the Ethernet frame when we're sending data from our workstation to the switch, from the switch to our router, from a router to the cable modem. However, once we get to the cable modem, we're going to use a different protocol there which requires a different frame. So what we'll end up doing is taking the packet out of the frame and putting it into a new frame, and this happens consistently throughout the transaction as removing the data across the internet. The packet will remain the same; the frame header will change as it goes from segment to segment to segment. Now in our frame, especially an Ethernet frame, remember earlier I said that there's a maximum amount of data that we can send in that? So here, our data is still intact here, that chunk of website that we're sending is still encapsulated inside of this frame. There's also a transport layer header, there's a network layer header, and then we have this Ethernet header. When we're working with this, especially Ethernet, the data part of this can have something called a maximum transmission unit for Ethernet, or MTU, and for Ethernet that's typically 1500 bytes. We can make that larger in some cases, but in most cases 1500 bytes is the maximum amount of payload, including the packet header, including the segment header that we can send using an Ethernet frame. So we have to break up the data into these smaller chunks to allow this to happen. And like I've mentioned, we can make that larger in some cases. Lots of times inside of a data center for specific applications, we will increase that MTU size, but for regular day-to-day use on our networks from our PCs, we rarely adjust that size. Once we have our frame constructed with the source MAC address, destination MAC address, and then layer 3 protocol that we're using, we can then take that frame, send it down to the physical layer. When we send it down to the physical layer, what's going to happen here is we're going to convert that into 1s and 0s, and then the 1s and 0s are converted into a signal. That signal could be a light signal that we send across fiber optics, it could be an electrical pulse that we send across a copper wire or it might be an electromagnetic signal that we send with wireless. So in order for us to get the website from Pluralsight over to our workstation, we have to send all that data up and down the OSI model in order to make all the hops that we need to make in order to get it from our server over to our workstation. And that involves taking our application layer information, putting it inside of a segment, taking the segment, putting it inside of a packet, taking the packet, putting it inside of a frame, and then sending it across the wire.

Summary

So to wrap up this discussion about the OSI model and encapsulation, we first reviewed the OSI model. We then introduced this concept of encapsulation, like taking a birthday card and putting it inside of an envelope; however, in this case we're taking that birthday card, putting it in an envelope, pushing that envelope inside of another envelope, putting that envelope inside of another envelope before we put it in the mail. I described the encapsulation concept layer by layer of the OSI model and we introduced the concept of segment at the transport layer, packet at the network layer, and frame at the data link layer. I hope this was useful. This is literally the foundation of everything else we are going to discuss in data networking. So I hope you found this useful. Let's move on to the next module where we start to talk about protocols more

Describe Protocol Uses and Port Numbers

Introduction

In this next module, we're going to look at protocols and port numbers for those protocols, and we're specifically looking at application layer protocols, and in our goals for this module, we're going to look at numerous application layer protocols. This can get a little bit tedious, but I have broken it up with a couple demos, and we're going to try to make it as interesting as possible. So we're going to start with data transfer protocols. Now almost everything in data networking is moving data from one device to another; however, there are some that meet this criteria more than others because we're actually moving a file that we're using as users versus doing some kind of network support or other utility. We're also going to look at authentication protocols, we're going to look at network service protocols too, things that help make our network run. We're going to look at network management protocols, stuff that we use as a network engineer to make sure the network is running smoothly. We're going to look at some audiovisual protocols, specifically the things that make our voiceover IP phone work. And then last, we're going to look at database protocols, specifically SQL Database protocols that we use in order to access database information. So if we take a look again at our OSI model, we're going to be looking at application layer protocols in this particular module; however, there is a transport layer component that we're also looking at, and that's going to be the port number. You may remember from the video on encapsulation and the OSI model that when we build the segment, we put in a source and destination port number, and this port number is going to be directly related to an application layer protocol.

Transferring Files

So let's take a look at these application layer protocols, starting with transferring data. Whenever we go on our workstation and we ask for a website like pluralsight.com, we are asking for an HTML document to be transferred from the server to our workstation. And in order to do this, we're going to be using one of two protocols, which are effectively the same thing, HTTP or HTTPS. This is hypertext transfer protocol or hypertext transfer protocol secure. And these protocols, what they do is they allow us to transfer an HTML document between a server and a client. Now at the application layer, layer 7, we call them by their names here, HTTP and HTTPS. But at the transport layer, they are associated with a specific port number. In the case of HTTP, it's port 80. And in the case of HTTPS, it is port 443. So port 80 and port 443, we typically associate with using web services. Now when we're using HTTPS on port 443, you will often hear engineers and people talk about using SSL, secure socket layer, or TLS, transport layer security. Now TLS and SSL sound very different. However, they are the same thing. You will read documents on the internet that claim that SSL works for function X, and TLS works for function Y. You may hear some fancy stories about how they're different and separate, but the reality is it is literally the same protocol. They just changed the name of it right around 2000. The late '90s, early 2000s, they changed the name from SSL to TLS. This is complicated and beyond the scope of this course. Just remember that SSL and TLS are the same thing, and we use transport layer security to provide the encryption when we're using HTTPS. Now not all of the files that we transfer when we're working with a data network are going to be HTML files via a website. Sometimes we have files on our workstation that need to get transferred to a server or to a network device, which means that we'd have some kind of file here that needs to get sent to the server. Or maybe there's some specific files that are not HTML files on a server that we need to access and download to our workstation. So when we're working with file transfer, we have several options. And these three options here, FTP, SFTP, and TFTP all kind of rely on the same general technology. So these are all called file transfer protocol of some kind. FTP, file transfer protocol. SFTP you can imagine is secure FTP. And TFTP, this is a little bit of an odd one here. This is trivial file transfer protocol. FTP operates on two separate port numbers, port 20 and 21. It is somewhat of a messy protocol, especially if we have to use it through a firewall. Secure FTP is FTP with encryption. It operates on port 22, which we're going to find out is the same exact port number for another utility that we use called SSH or Secure Shell. The reason that it's the same port number is that we're actually creating an SSH connection and then using FTP on top of it. But for now, you have to remember that SFTP uses port 22. And then TFTP reminds me of an old movie that I used to love when I was in high school in the early '90s. It was Bill and Ted's Excellent Adventure. And TFTP uses port 69, dude. So when we are working with these, TFTP is typically going to be used to transfer small files, and I have used TFTP most in my career to transfer an operating system file from my workstation that I use to manage the data network. Use that to transfer some operating system file or upgrade to a device, like a router or a switch, in order to have the file on that device so I can upgrade the device to the latest operating system. So FTP, SFTP, TFTP are all used to transfer files, and they all have generally the same type of operation. There is one more from Microsoft here called SMB, which stands for server message block. Now if you work in a large business and you have a shared drive, a shared network drive mounted on your workstation, typically we're using SMB as the protocol for this. And what it allows us to do is mount a drive on our Windows workstation or other workstations. And with SMB then, we can actually just browse that drive as if it were a local drive on our workstation, yet it is a network drive. And we can copy files from that drive to our workstation or copy files from our workstation up to that drive, and it's going to use server message block to transfer that file instead of these other FTP varieties.

Demonstration: FTP and SMB

So let's do a demonstration on this then, and let's take a look at FTP and SMB in operation. So here I am on my Windows 10 workstation. I need to launch the File Explorer in order to demonstrate SMB. I have an icon here on my quick launch bar. We can also get to it by searching Windows here for File Explorer, and it'll show right up. Or I can click on the Start button here on the Windows icon, and it'll be this icon right here right above the Settings and Power button. We can click on that, and that'll also open File Explorer for us. Now in order to mount a drive here, in order to mount a network drive using SMB, what I need to do is right-click on the Network icon here in File Explorer and click Map network drive. And what this will do is it'll give me a option here to choose the drive letter that I'm going to map to. Typically, network drives are mapped with a letter above A, B, C, D, E. The reason for that is old computers used to have an A drive, a B drive, and a C drive, maybe a D drive for a CD ROM. The A and B drives were floppy drives. So for our purposes, I'm going to choose the P drive, and next we choose the folder that we want to mount. So it gives us an example there, and it says example, we put backslash backslash, the name of the server. My server name is bluebox. I'm a big Doctor Who fan, so I called mine bluebox just like the TARDIS. And then we put in the name of the folder that we are wanting to mount, and that is called SMBdemo in my case. So we click Finish here, and what'll happen now is it's going to say, hey, we need to enter some credentials. So it'll usually try just the credentials that are local to the workstation first. And if those don't work, in this case it failed, and now I can put in my user name, which is bagurdes, and my password and hit OK here, and it should mount my drive. And there we go, SMBdemo on bluebox, and we can see I have a couple images in here, some PNG files, and this is just my purple switch, which is right here. And we also have a router, my router icon, which is in blue here. So, this is how I access these files on my SMB drive. I can copy these files out of here onto my desktop, and I should be able to copy a file into here. Let's see if I have a document that I can put in there. I have this sslkeylog document. So what I can do is I can copy that document by right-clicking, selecting copy, going over to bluebox, and then pasting it in here. And now, I have moved the file from my workstation to my shared drive. So that's SMB in operation. I can then unmount that drive if I want to by clicking Disconnect here. And now when I reopen File Explorer, it is no longer mounted. Second thing, we can check out FTP. So to get FTP to work, I downloaded an application called FileZilla, and you can download this. Just do a Google search for FileZilla. It's a part of the Mozilla project, which makes Firefox. So this is a part of the Firefox suite of applications. So here, this is an FTP client. And what I'm looking at on this screen is up on the top here, I have an option to put in the host name of my FTP server along with a username, password, and port number and then a Quickconnect button. And then over on the left-hand side here, I have a list of all the folders and files on my local workstation. All right, so this is my PC over here on the left side of the screen. On the right side of the screen, this is the remote side. So I haven't connected yet, so let's do that. My FTP server name is bluebox. Let's move the mouse so we can see that. So my host name, my FTP server is bluebox. The username is bagurdes. My password, I'm not telling you. And I don't need to fill in the port number because it's just going to use the default port. And then I hit Quickconnect. And it's going to pop up with a message here because in modern networking, when we're transferring files, we want to avoid using unencrypted connections because when we use an unencrypted connection, it means anybody on the network listening can download that same file and use it for whatever purposes they want, and I may not even be aware that they stole it. So this is just giving us a warning. Since this is a demo in my local environment with data that's not important, it's just these images, I'm not going to worry about that for this particular case. So I hit OK. And then it pops up with a listing of all of the folders I have access to on my FTP server. So here, one of those folders is the SMBdemo. I create a specific folder just for SMB. So if I open that up, we can see that here's that file that I just put into that folder, that sslkeylog file. All right, I also have the FTPdemo, which has my router and switch in it. So what I can do now is I can just do some dragging and dropping. So maybe I want to move the switch into my documents folder. So what I can do is I can just drag the switch-purple over to Documents. And now if I open up File Explorer and look in Documents, we'll see that there is now a purple switch in there. I can do the same thing in the other direction. If I want to go to Documents and take that sslkeylog file that I put into the other folder before, drag it over to my FTPdemo folder, now the sslkeylog file is in that folder as well. So this is how FTP works. I know this is a very basic demonstration. But what we're looking at here is that FTP allows us to move files from one device to another, as does SMB. It's really just the interface that's different. You can also use FTP over the command line. That's a little more sophisticated. However, it basically is doing the same exact thing here of moving files from one device to another using this protocol called FTP or SMB.

Email

The next application layer protocols we're going to take a look at here are email. So email is a way of transferring a file from a server to a client again. We're just using a different format this time, and we're using an email format. There are three methods that we're going to need in order to transfer email. We have POP3, we have IMAP, and we have SMTP. POP3 is Post Office Protocol version 3, IMAP is Internet Message Access Protocol, and SMTP is Simple Mail Transfer Protocol. POP3 And IMAP are typically reserved for retrieving mail from a server to the client. And SMTP is used to send mail from a client to the server. So when we're configuring email clients, usually we need to configure either POP or IMAP, as well as SMTP. Now these protocols are still used today, and they're common. However, the way we configure them has changed pretty drastically over the last 10 years. Mail services like Gmail, Outlook, Yahoo Mail, and others have changed the way we configure our devices for email. A lot of times, we just say that we're using Google, we put in our username and password, and it automatically configures all the protocols for us versus what I may have had to do in 1999. I would have to actually go in and configure all the details of my email system, configuring the correct POP3 server, the correct port number, the correct SMTP server and its correct port number, as well as my credentials. So time has changed on how we configure and use this, but we still need to know that POP3, IMAP, and SMTP are used for email. Now all three of these have two transport layer port numbers they can use. And if you've seen a theme so far here, we've been talking about unencrypted and encrypted transfer of messages. So POP3 here for the unencrypted POP3, it's port 110. Encrypted is port 995. For IMAP, we have unencrypted on port 143 and encrypted on port 993. And then for SMTP, we have port 25 for unencrypted and port 465 for encrypted. Now we typically can't just change the port number in our system to make it encrypted. We actually have to have that protocol installed and enabled on our devices, which, in most modern mail clients, these protocols are enabled and we can use them.

Authentication and DHCP

Let's talk about authentication. Now when we talk about authentication, usually we're talking about authentication of a client to a server. When we're using LDAP, oftentimes we have a workstation, especially in a corporate office, where we have to use a username and password to log into that client. And what some authentication services can do is instead of having those credentials stored locally on your workstation, like you might have with your home computer, the credentials are actually stored on a server inside of the organization's data center. And that way it doesn't matter what workstation you log onto in the workstation, your settings and desktop and mapped drives and whatnot will all show up the same on your client workstation, regardless of which one you're using. There's two protocols that we use for this. It's Lightweight Directory Access Protocol. One is not encrypted; one is with the S and it's encrypted. So what this is is we put in a username and password on the client, we send it to the server, and the server will then send a token back saying yes, this user is authenticated. And then after that, the server may send additional information that has user settings and whatnot for the client. So here for LDAP, we're going to use port 389. For

LDAP secure or LDAPs, we're going to use port 636 at the transport layer. Next, we're going to talk about a series of network protocols that we use for network services. The first one here is called Dynamic Host Configuration Protocol, or DHCP. Now DHCP is responsible for giving your workstation an IP address when it first is plugged into the network. Now IP addresses we're going to learn about in another couple modules. IP addresses are and identifier for your device on the network. It's one of the identifiers for your device on the network, and it operates at the network layer. Now the IP address, it has very specific properties, and it has to meet very specific criteria in order for your client to be able to communicate with the rest of the network or the internet. So instead of having users configure this or configure them manually, we oftentimes will use a DHCP server to do this. As a matter of fact, this is the case in your home network. Your cable modem or your cable modem router or your wireless access point, all those devices have capabilities to offer a DHCP server. That way, when you turn on your device and you connect it to the wireless network or you plug your device into the wired network, it automatically gets an IP address. So here what'll happen is when we plug the client into the network, we're going to send out a discover message. This DHCP server then is going to reply with something called an offer message. And the offer message is going to have the IP address, a subnet mask, the default gateway, DNS server, possibly some other information as well. And then what'll happen is the client will respond back and say, yes, I accept that, and then the server will respond back and say great. I acknowledge that. I've added it to my database. So here, DHCP is going to use port numbers 67 and 68 in order to operate on a network.

Demonstration: DHCP

So let's do a demonstration here of DHCP. We're going to examine the IP configuration using DHCP. So I'm back on my Windows 10 workstation, and we're going to look at DHCP. Now there are several ways to examine this. First, what I'd like to show you is where we tell our workstation to obtain its IP address automatically using DHCP. Now Windows does not necessarily make this an easy place to get to, and there are several options we have to get there. The easiest, I think, is to go to Control Panel. And to get to Control Panel, I do a search for it and click on it. And then inside of this list, we have to find Network and Sharing Center. Once we get to Network and Sharing Center, we're going to click on Change adapter settings, and then we're going to go to Ethernet0 here. On my workstation, we're using this one because this one is connected. You can see this other Ethernet1 here. There's another network interface card on this device, but it is currently disconnected, and you can see that there's an X here, a red X, and it says Network cable unplugged. So I'm going to right-click on Ethernet0. Go to Properties. And we've got one more to do. We go to Internet Protocol version 4, TCP/IPv4, and then we click Properties. And once we're in here now, this will allow us to set the method that we obtain our IP address. So right now, it is set to obtain IP address automatically and obtain DNS server information automatically. We can set this manually, and we will do this in a later demonstration/lab. For now though, we're going to keep both of them set to obtain it automatically. I hit OK, close, close, and close all my windows. And the next thing I can do is open up a command prompt. Now I have a quick icon here to the command prompt. There are other ways to get to this. If I just type into my search again cmd, that will open up command prompt as well. Once I'm in command prompt, a command that we're definitely going to need to use a lot of is ipconfig. All one word, i-p-c-o-n-f-i-g. If we hit Enter here, it will show me my basic IP information. Tells me the IPv4 address that I have assigned, which is 192.168.77.2. Tells me my subnet mask and my default gateway. We are going to learn what each one of these pieces of information mean as we make our way through these series of courses. As a matter of fact, in another module very soon, we're going to talk specifically about IP addresses. So this address was handed out to me by a server. I can ask for an updated address. And chances are it's going to give me the same address. But the way I do that as I say ipconfig, and then I hit a forward slash and I say release. And what this will do is it will release my IP address so that I no longer have one. So if I type ipconfig now, it says I don't have any information here for my IP address. I can ask the DHCP server for another address by saying ipconfig /renew. The DHCP server will likely just give me the same address I had before because it's already stored in that DHCP server's database. We're going to learn a little bit more about DHCP as well later on in this course. But I wanted you to see that we can control getting and releasing an IP address in Windows using the ipconfig /release and ipconfig /renew commands.

DNS

The next network service we're going to take a look at is probably the most important service on the internet. It is Domain Name System, or Domain Name Service, DNS. This is also called a name server sometimes. DNS, what it does for us, is it allows us to take names like pluralsight.com, google.com, facebook.com, you name your favorite website that you go to often, and it allows us to translate the name google.com into an IP address that we can actually use to transfer information at the network layer. So the network layer of the OSI model uses IP addresses as a source and a destination of the packet so that we can move that packet, usually containing a segment of information and maybe some other application layer data, we can use that packet then to move from our client to the device that we're trying to reach without having to know the specific IP address of every device that we're trying to reach on the internet. The way this works is when I go to google.com on my workstation, I will first send a message to the DNS server saying, hey, DNS server, what is the IP address of google.com? And the DNS server will reply to me with the IP address of google.com. Then, after I have that information, I can construct my packet using the source IP address of my client here and the destination IP address of google.com. Here I'm saying hey, 216.58.216.206, which is google.com, send the website that you have available to 203.0.113.55, which is my client workstation. So we send that to the internet over to Google, and then Google can send back the website to my workstation. So DNS, what it's doing for us is it's resolving names that we know as individual people because it's easy to remember something like Google or Facebook or Pluralsight. It's much easier to remember that than it is to remember these series of four digit numbers between 0 and 255. It's going to be nearly impossible to remember, plus those IP addresses oftentimes change depending upon where you are in the world. DNS is super critical, and we have found that it's an easy place for attackers to attack a DNS server and really mess up how we resolve names into IP addresses and can cause some real serious issues. So we have to take some extra security when we're setting up DNS servers to make sure that they don't get compromised by nefarious agents trying to attack our networks. DNS works on port 53. So whenever we see DNS over port 53, we'll know that that DNS is not actually encrypted. There is a newer version of DNS coming out. It's actually not a new version of DNS. It's a new version of transporting DNS information where we actually do it over HTTPS, so it's called DNS over HTTPS, or DoH, and that's going to use the same port number that HTTPS uses, which is 443. So we don't need to know that so much for the Net+ certification exam, but know that in the next several years we should start to see DNS start to operate over HTTPS, making it more secure because it will be encrypted.

Demonstration: DNS

So let's do a demo of this now. We're going to examine DNS now with a utility called nslookup. So we're back again on my Windows 10 workstation. I'm going to open up Command Prompt. I have that icon on my Quick Launch bar here, but you can also do a search for cmd, and that will also get you to the Command Prompt. So in my window here now, what I want to do is take a look at DNS and how DNS is resolving a hostname into an IP address. So if I type in here nslookup, that command is going to send a request to my DNS server and ask to resolve a hostname. So I'm going to type in pluralsight.com and hit Enter, and what'll happen is it'll tell me the address of my DNS server, which is at 192.168.77.1, and then it gives me a list of IP addresses for Pluralsight, and it says 35.165.232.155 and two other addresses here. So there are multiple addresses where we can reach pluralsight.com. The first address is the one that our computer is going to use should I go to the website for Pluralsight. I can do the same thing with other websites here. If I look at google.com, oops, I have to spell it correctly here, here we're seeing google.com. It's giving me 142.250.72.174, as well as this other address that has letters in it. This is an IP version 6 address. It's written in hexadecimal, and we're going to learn more about hexadecimal IPv4 and IPv6 addresses later on in this course. This is a neat utility that we can use here just to see the IP addresses of pretty much anything out there that is available on the public internet. If I look up facebook.com, we'll see that Facebook has both an IPv6 and an IPv4 address, and written in the IPv6 address, we can actually see the word face:b00c, which happens to be available in hexadecimal as long as I replace the k with a c. Anyway, this is nslookup. We're going to learn more about DNS and how DNS actually operates to resolve this in another module in this course as well. Let's keep going.

NTP

The next network service that is incredibly critical on our network is Network Time Protocol. Now Network Time Protocol, or NTP, what it is, is it's a device on the network that has a clock on it. That clock is usually synchronized with some government-run atomic clock, so it has a very precise time. So our server on the network would go retrieve the precise time from some atomic clock some place out on the internet and then synchronize it locally so that when local clients need to know how to set their clocks, they can send a message to our server, our NTP server, and the NTP server will reply back with the time. In this case, it's 3:00 pm. And now the client can set its time. This will include the date on the workstation. Now NTP is important because services like encryption will oftentimes validate whether or not a server that we're connecting to is valid by checking a certificate, and the certificate is often only valid for a specific period of time, so we need some type of clock to validate that the time of the certificate is okay to use. Also, we have log messages that are stored on clients, servers, network devices, firewalls, etcetera, and those logs need a precise time so that if an event occurs, whether it be a technical event or a security event, we know exactly the right time that that's happening so we can coordinate that with other device logs on our network. Now when we use NTP, it uses something called Coordinated Universal Time, or UTC. And the idea here is this is how we accommodate different time zones. Now the way UTC is set up is if we look at a map of Earth here and we zoom into England area, there is an imaginary line called the Prime Meridian, which runs through Greenwich, England. Now Greenwich, England is just a little bit east of London, and there's a lot of big events that happened in Greenwich, England. As a matter of fact, at one point in time it was home to the British Navy. The British Navy invented a lot of utilities so that they could travel around the world. So the Prime Meridian is used as our 0 point when calculating UTC. So when it is midnight in Greenwich, England, the time is 00:00 UTC. If we zoom out here and we take a look at maybe what the time might be in Chicago when it is UTC 00:00, it is -06:00 hours in Chicago. Now this is assuming that we're not using daylight savings or that we're at a time when daylight savings is not in effect. So I'm ignoring daylight savings here to keep this a little bit more efficient. Know that Network Time Protocol servers do have all of the calculations needed for daylight savings time as well. But to keep the simpler for the moment, we're going to assume no daylight savings time. So if it is at midnight, 00:00 UTC, in Greenwich, England, it is 6 hours before that in Chicago, or 6 pm. If we look at that same in Utah at midnight in Greenwich, England, in Utah, it would be 5 pm, 7 hours before that. If we look at another place on Earth here and take a look at the time in New Delhi, New Delhi is +05:30 from UTC, which means that if it is midnight in Greenwich, England, it is 5 hours 30 minutes ahead of that in New Delhi, which is a little bit odd that there's that 30 minutes in there. But it doesn't really matter. It's just a way of measuring what time it is so that we always have the same time no matter where we are on the Earth using UTC. So if it's at midnight in Greenwich, England, it is 5:30 in the morning in New Delhi. So NTP is a critical service on our network when we're talking from the client to the server.

Network Management Protocols

Next, let's talk about network management. So network management means that oftentimes we have devices on our network that we are going to manage from some central location. When I was working as a network engineer, my desk was in one office of a very large hospital organization, and I had to manage network equipment on all of the sites within the organization, which was hundreds, if not thousands, of pieces of gear. So the way that we did that was we used one of two utilities here. We either used Telnet or Secure Shell. The reality is I actually used Secure Shell, or SSH, the most often because it was encrypted versus Telnet being not encrypted. So this is a way to actually use a command-line interface to access devices around the network, whether it be a server, like a Linux server. Could also be a router, or a switch, or a firewall, or a multitude of other devices that we can access with this. So on our Network Administration Workstation, we might SSH to a router, or a switch, or a server here, or even a firewall. The devices that we are SSHing to are going to be the server. The device we're SSHing from becomes our client. The next utility we're going to look at is SNMP, or Simple Network Management Protocol. And what this is, is it's a way for devices to send information back to a centralized server, information like log messages, or information about port up and down, or maybe some other event on our device that we configure to tell our SNMP server about. So in this case, all the devices that are sending messages to the SNMP server become our client. This could be a server. This could be a router, a switch, a firewall. The icons right here, the one that is orange with the globe next to it that I have is just as a server. The circle with the arrows in it, that's a router. The rectangle with the arrows in it, that is a switch icon. And then the circle, the green circle with what looks like a fast forward button on it, that is a firewall. So all these devices can report in to the server what's happening with them, or the server can send out a message to all these devices and say, hey, walk the tree, which means tell me all the information that you have about your devices that has an SNMP number assigned to it. Those numbers are called MIBs, or management information base, and they can send all that information back to the server then, and the server can use that information to populate a table of all the events that are happening on the devices. Another thing that can happen is if an event happens on a device, like some event happens on that switch that needs attention, what can happen here is we can have an SNMP trap. And what that means is that when an event occurs on a device, both good or bad, it can send a message to the SNMP server saying, hey, this event happened. And the idea here is, is with a central server, we can configure that central server to send out alerts to the network administrator saying, hey, the network is on fire; we better fix it. SNMP uses ports 161 and 162. Another network utility we can use that's very similar to SNMP is syslog. Syslog is a mechanism to take the logs on each one of our devices and send them to a centralized syslog server so they can be correlated with other events on our network. So if, for example, I plug a network device into my switch, that's going to generate an event, right? And the event doesn't have to be anything significant. In this case, it's just that the port changed state from down to up. When that happens with syslog enabled and configured correctly, we will send that syslog message from the client over to the syslog server where it can be recorded, so now the syslog server knows of that event as well. Syslog is going to use port number 514. Another utility we can use here is Remote Desktop Protocol, or RDP. Remote Desktop Protocol allows us to access the graphical user interface of devices on our network when we are not local. So we may, on our client, need to configure a Windows Server, so we can use RDP, or Remote Desktop Protocol, to actually make a connection to the server, and then we'll get a screenshot of the graphical user interface of the server that we can then interact with. This operates on port 3389. Let's talk about some audiovisual protocols now. Now the audiovisual protocols that we're using here allow us to either have Voice over IP phone calls or video phone calls between two organizations. And here we're going to use a protocol called H.323. This operates on port 1720, and that allows for video and audio communication between two devices. Another one we can use here for Voice over IP telephones is SIP, or Session Initiation Protocol. And Session Initiation Protocol uses port 5060 and 5061, and it's the protocol we use to transfer voice when we're using our Voice over IP phone to a voice gateway, which gets us out to the rest of the telephone system so we can make calls.

SQL Database Protocols

Last here, let's talk about SQL databases. So databases themselves also use port numbers. SQL is a protocol called Structured Query Language, and it actually means several things. It can mean the server, like a SQL Server. It can mean the language that we use to access data on that server. And it can also be the protocol used to communicate across the network to access that database. So there are three flavors of this that are important. We have MySQL, which is open source; we have SQL*Net, which is by Oracle; and we have SQL Server, which is a Microsoft product. MySQL operates on port 3306, SQL*Net operates on 1521, and SQL Server operates on port 1433.

Summary

That's a lot of protocols to go over. Some of these protocols we're going to do an even deeper dive later in this course so that we can best understand how they work. But to wrap up this one, this was really to identify that there's lots of application layer protocols we use to transfer data, and they have an associated port number at the transport layer. So these application layer protocols we looked at were like data transfer protocols like FTP, SFTP, TFTP, and SMB. We looked at authentication protocols like LDAP. We looked at network service protocols like DHCP, DNS, NTP. We looked at network management protocols like SNMP, SSH. We also looked at some audiovisual protocols like H.323 and SIP, and then also database protocols, specifically for SQL, whether we're using MySQL, SQL*Net, or SQL Server. These protocols are the beginnings of what's going to become a much richer and complex but organized compilation that makes data networking work. So what we're going to do in the next module is actually go down to the transport layer and take a look at how TCP and UDP work because all of these application layer protocols are either going to use TCP or they're going to use UDP, and there's going to be that specific port number associated with it, which I discussed in this module here. So let's move on.

TCP and UDP

Introduction

In this next module, we're going to talk about TCP and UDP. Our goals for this module are to talk about transport layer protocols. We've spent a lot of time in the last several modules talking about application layer protocols and their transport layer port number. Now we're going to talk about two transport layer protocols, TCP and UDP, Transmission Control Protocol and User Datagram Protocol. And we're going to wrap up this module by talking about some protocol hierarchy so we can actually see how all of these protocols interact with each other using the guidelines of the OSI model.

TCP and UDP

Here's our OSI model again, application, presentation, session, transport, network, data link, and physical. We are going to focus this time on transport layer. The transport layer is responsible for building a session and maintaining a session between two endpoints, typically a client and a server on a data network. Transmission Control Protocol is one of the most popular Layer 4 protocols that we use. Transmission Control Protocol is also called TCP. You'll typically hear it called TCP. So the way I like to think about TCP is that it is a protocol that initiates communication between two devices. And this is a lot like the way we initiate a phone call between ourselves and our friend. So here is Leia and Obi again, and Leia wants to make a phone call to her friend, Obi, but in order to do that, she just can't pick up her smartphone and start talking into it and expect Obi to be there. She has to go through a very precise set of steps to reach Obi. So she has to pick up her cell phone and wake it up. She's going to then open up the app for the phone and dial a phone number or find a contact for him, and that's going to cause Obi's phone to ring. And once Obi sees his phone ringing, he can pick up his phone, hit the button to answer it, and then each one of them are going to say a message. And he's going to say, hello, and Leia is going to respond with hello. Now we have established a session for these two individuals to have a conversation with each other. Before that, they could not have a conversation. They were not in the same room. What they needed to do though is Leia needed to go through those very precise set of steps, wait for Obi to say hello, she says hello, and now they can transfer any information they want. And built into this conversation are cues and messages that allow each other to understand that the message that was sent was the one received. Leia may say, hey, Obi, I need some help, and Obi may respond and say, oh, I see. Obi may say, wow, that sounds really difficult, and Leia responds with uh-huh. So we send these cues throughout our conversation to validate that the message that was sent was actually received. However, we can also have messages that indicate that we did not receive the message. So Leia may say hey, I need some help, and Obi may say, hey, you're breaking up. I can't hear you. Or Obi may say something and later responds with, I don't understand. And these are signals that we need to retransmit the information being sent. Then when we're done with the conversation, unless it's some kind of argument where we just hang up the phone, typically we're going to say goodbye to each other. So Leia may say, okay, great, thanks. I'll see you soon for your help. She says goodbye, Obi says goodbye, and then the conversation is over. We don't need those goodbyes to end the conversation. It just makes it more polite. In TCP, we have something called the three-way handshake, and this is a very precise setup process that the client and the server need to go through before we can send information. So if I'm trying to receive a website from the web server, before I can do that, I have to go through this TCP process where I first send a message to the web server called a SYN message. The SYN message is the first message to the web server indicating that the client wants to start a conversation. Then what'll happen is the web server will reply with a message called a SYN-ACK. The SYN-ACK is just alerting the client that yes, I'm available for this conversation. Let's get started. And before they get started actually transferring the website though, the client is going to respond with an ACK, or an acknowledgment. So the three-way handshake goes SYN, SYN-ACK, ACK. Once that process is done, now the PC can say, hey, web server, send me that website, and then the web server can reply with a message containing the website. Here's the website, and now it's loaded on the client's computer. When the conversation is all done and the website has finished transferring, the web server or the client will send a message called a FIN, and then the client will respond with a FIN-ACK. Once the FIN-ACK has been sent, the client will respond with its own FIN message, and then the web server is going to reply back to that with a FIN-ACK. So the four-way disconnect is FIN, FIN-ACK, FIN, FIN-ACK. And then the two ends know now that the conversation is over, and now we can no longer send messages between these two devices. Much in the same way that Leia and Obi said goodbye to each other and hung up the phone, now they cannot communicate anymore unless one of them picks up the phone again and dials the phone number, waits for it to ring, answers the phone, then they can send the message again. Same thing is true in TCP. In order for this client and this server to communicate in the future, they will need to send another three-way handshake. Now that's not the only way to end a conversation in TCP. There's another way here that we call a TCP reset. And the TCP reset can be sent from either the web server, and when the TCP reset happens, boom, the conversation is over. It's like Obi or Leia just hanging up the phone and stopping the conversation, right? In order to start that conversation up again, Leia or Obi would have to redial the phone to start that conversation. TCP reset doesn't have to come from the web server though; it can also come from the client. The client can just say nope, we're done. And another possibility here is that there's some device in the middle, like a firewall, that's paying attention to this conversation, watching for nefarious activity, and that can actually send a reset as well. However, that is a more sophisticated application of this reset. Regardless, we have several ways to disconnect our conversation. We can either do the four-way disconnect or we can do a reset. Next, let's talk about User Datagram Protocol. UDP works similarly to TCP in that we're going to set up a session between the two devices. However, this time, we're just going to start talking. We're just going to tell the server, hey, send me some data. And then the server, if it receives it and it's able to, it will send the data to the client. And there's no mechanism to make sure that the data sent was actually received. There's no three-way handshake. There's no four-way disconnect. There's no resets. This just works. So here with UDP, it's no three-way handshake, no reliable communication, no sequence numbers, acknowledgement numbers, and it's used as an efficient way to transfer data. However, we don't use this as much as you think we would because there's no way to validate that the data sent was actually received.

Port Numbers

Let's talk about transport layer addressing. We've already talked about port numbers. We've talked about the application layer protocols and their associated transport layer port numbers, so let's take a little bit deeper dive into this. There are two sets of port numbers here. There are server port numbers, which are called well-known and registered port numbers. There are also client port numbers, and these are called ephemeral port numbers. Ephemeral is another word for temporary. So when we are working with this, the well-known port numbers here are typically from 0 to 1023, and then the registered port numbers are from 1024 through 49,151. The ephemeral port number ranges are from 49,152 all the way up to the last port number, 65,535. Now how does this work? Well-known port numbers are typically port numbers for applications that we have already discussed, HTTP on port 80, HTTPs on port 443, FTP on ports 20 and 21, SSH 22, Telnet on 23. Registered port numbers are going to be for custom applications, and they can both be official and unofficial. There are many games that would use a registered port number. There are other applications that might use a registered port number for like file sharing, a lot of custom software from manufacturers like Dell or Microsoft or IBM, and they come up with a special network application that will use a registered port number. The ephemeral port numbers are going to be used by the client or the PC side of this in order to identify the session that they're creating. So if I'm going to send a message to my router here, my router is going to act as the server for Telnet. So if I'm going to Telnet to my router, what I would do is the source port would be an ephemeral port number. That's my Layer 4 address for my PC. It's going to be source port of let's say 49,152, and the destination port for that would be 23 because I want to reach the Telnet service on my router. All of these protocols that we've been talking about have application layer protocol dependencies. So what this means is that for these protocols that I've listed here, these are all protocols we've talked about before, HTTP, HTTPS, FTP, SFTP, SMB, POP3, IMAP, and so on, all of these have a port number that they're assigned to, and we've discussed this, and then in addition to that, they will use a specific transport layer protocol. So they will either use TCP, UDP, or they have a possibility of using one or the other. They won't use both at the same time, but they would use one or the other. So in this case, most of the popular application layer protocols here we're seeing, including HTTP and HTTPS, our email protocols, our File Transfer Protocols, these are using TCP. LDAP and TFTP can use UDP. As a matter of fact, TFTP only uses UDP to work. All of these protocols will use IP at the network layer. So we have application layer of the protocols, we have transport layer port numbers and the associated protocol they're using, and then the network layer protocol for everything is going to be IP. Here are some additional protocol dependencies: Telnet, SSH, and Remote Desktop all use TCP, DNS, SIP, and H.323, SNMP all can use either TCP or UDP. DHCP and NTP, our time protocol there, will use UDP. And all of these protocols will use IP at the network layer. So this is a really good introduction now to understanding the protocol hierarchy. We have the application layer protocol, which calls a transport layer port number and transport layer protocol, either TCP or UDP, and then below that at the network layer we have Internet Protocol as the network layer protocol we're going to use.

Summary

Let's wrap up what we've talked about here. Talked about transport layer protocols, both TCP and UDP. We saw TCP was connection-oriented and that we need the three-way handshake in order to build that conversation and a four-way disconnect or a reset to end it. And during the conversation, we have mechanisms to make sure that all the data sent was actually received. UDP, User Datagram Protocol, was a lightweight version of that. No three-way handshake needed. No disconnect needed. We just start sending data. And then last, we looked at that protocol hierarchy and saw the application layer protocols. We talked about the port numbers, the transport layer protocol being used, and then the network layer protocol being used.

Introduction to Binary and Hexadecimal

Introduction

I think I've been teasing you over the last few modules about binary and hexadecimal values, saying that we're going to cover this in a later module, and, well, we're here now. So in order to really understand what we're going to do next with IP addressing, we need to understand how binary works and how hexadecimal works and how to convert between binary, hexadecimal, and decimal. So let's get started with this module. Our goals here are going to be first, introduce the need for binary, review some primary school mathematics so we understand how to count so we can learn how to count in binary then. Then we're going to convert binary to decimal, we're going to convert decimal to binary, and then we're going to introduce hexadecimal and see how we can convert between the three different number formats.

Binary 101

Binary 101. So, to understand binary and give some context here, let's start by talking briefly about base 10. Now we learned this at a very young age; we learned how to count. We count 1 to 9, and then after 9, we add another placeholder and count to 10. And that's likely because we have 10 fingers on our hands, and it's likely a very ancient method of counting. However, when we talk about binary, really we're talking about only two placeholders we can have. The position can either be on or it can be off. It can be a 1 or a 0. So when we count in binary, it's going to count a little bit differently with the same rules. It's just a matter of understanding how to apply those rules correctly. So let's do an example here first and count in decimal, or powers of 10, or base 10. All these mean the same exact thing. So when we count in decimal, we count starting with 0 over in the ones placeholder, and we count to 1, 2, 3, 4, 5, 6, 7, 8, 9, and then once we get to 9, we have used up all 10 values 0 through 9 when we're counting here. Once we reach that highest level, 9, we need to add a placeholder, and that's the tens placeholder. So then we start counting over again, and now we start with 1 0, or 10, and then we count to 11, 12, 13, leaving our tens placeholder the same in every case here and incrementing our ones placeholder until we get all the way up to 99. We'll count 11, 12, 13, 14, 15, 16, 17, 18, 19, and then we increment our tens placeholder to 20. And that goes all the way up into the 90s until we reach 99, and then we've maxed out the placeholders for both the tens and the ones, and the number that comes after 99 is 100. And then we count all the way up to 999, and the number after that is 1000, so we add another placeholder. And we keep adding these placeholders until we reach the number that we need, right? We can add more and more and more placeholders here in the decimal system until we reach the number that is important. So when we're thinking about decimal, we're thinking about counting in powers of 10, 0 through 9. Computers though, they really need to think in binary. All of their circuitry is designed to think either as on or off. Most of the components in computers work with bits, and bits are nothing more than a placeholder for a 1 or a 0. So when we're counting in binary, it's going to work just like we counted in decimal; however, we only have two values per placeholder. It's either going to be 0 or the next one is 1. So here, when we count 0 in the 1s placeholder in binary, the next value that we can put there is 1. And once we hit 1, we're out of values to put in the placeholders now, so we have to add a placeholder, and that's when we add the 2s placeholder. And now when we count higher than 1, we're going to count to 1 0. So 1 0 is the third value in binary. It goes 0, 1, 1 0, and then the next placeholder, we're just going to keep the 2s placeholder as a 1 and increment the 1s placeholder by one value, which becomes 1 1. Now that's not 11. This is 1 1. It actually equates to 3. And we're going to look at how to get that conversion to work in just a little bit, but for now, we're just gonna be counting here in binary. Now you should be able to see now once we reach 1 1, all the values that we can increment in the 2s placeholder have been used up, 0 and 1, all the placeholders in the 1s placeholder have been used up, again, 0 and 1, so we need to add another placeholder here, and it's going to be the 4s placeholder. So after 1 1 in binary comes 1 0 0. Next, we're going to increment the 1s placeholder by one value to bring into 1 0 1, then 1 1 0, then 1 1 1. And you might see a pattern forming here with binary. Now my brain seems to be tuned into patterns pretty well, and one of the things we can see here is that if we look in the 1s placeholder column and we ignore the rest, we see that the 1s placeholder, as we're counting, goes 0 1 0 1 0 1 0 1 as we go down that column. If we look at the 2s placeholder, it goes 0 0 1 1 0 0 1 1 as we count down the column. If we look in the 4s placeholder, it goes 0 0 0 0 1 1 1 1. So as we're counting through these, there's going to be a pattern forming here that we should pay attention to. If the pattern doesn't make sense to you right now, that's okay. Don't worry about it. We're just going to continue to follow the rules here. In our last value here, we've used up 1 1 1 in all of our placeholders. It means we need to add another placeholder, and in this case, it's going to be the 8s placeholder. So in the 8s placeholder then, all I've done is add 0s in the 8s placeholder then. Next, we're going to scroll up here and start counting again because now we've used up 1 1 1 in the 4s, 2s, and 1s. Now we're going to put 1 in the 8s placeholder and start counting all over again, 1 0 0 1, 1 0 1 0, 1 0 1 1, 1 1 0 0, 1 1 0 1, 1 1 1 0, and 1 1 1 1. Now that there is a 1 in each of the placeholders, we can't have any more values here, we've maxed it out, we're going to add another placeholder. It's going to be the 16s placeholder. Then once we get to the maximum value there, we add another one, the 32s placeholder, then the 64s, then the 128s. If we keep counting, we'll go 256, then 512, then 1024, then 2048, and so on. So every placeholder is going to be double the previous placeholder.

Converting Binary to Decimal

So that brings us now to converting from binary to decimal. We saw how we count in binary. Now let's see if we can do some conversion. So converting from binary to decimal is actually pretty easy. These 1s and 0s are literally just on/off switches, and what we do is we multiply the value of the placeholder times the placeholder itself. So in this case with 11000000, what we do is we start at the 128s and we say okay, there's a 1 in the 128s, so 128 times 1, and then we say there's a 1 in the 64s, so 1 times 64, then 32 is 0 times 32 all the way down to 1. So we have 128, plus 64, plus 0, plus 0, plus 0, plus 0, plus 0, plus 0, and we end up with 192. So in binary, 11000000 equals 192. If we try a different number here, let's try 00010110, we're going to apply this same format here again, 0 times 128 is 0, 0 times 64 is 0, 0 times 32 is 0, 1 times 16 is 16, 0 times 8 is 0, 1 times 4 is 4, 1 times 2 is 2, and 1 times 0 is 0, we add those all up and we get 22. So 10110 is 22. Let's try a few more here. So this one, we can look at this, like I said before, as a series of ons and offs. So if the value of the placeholder is 0, it's off, we don't count it. If it's a 1, it's on, we do count it. So here we have 0, plus 64, plus 0, plus 0, plus 0, plus 4, plus 2, plus 0, we end up with 70. Let's try a different one here, 11001000, on, on, off, off, on, off, off, off. We get 128, plus 64, plus 0, plus 8, plus 0, plus 0, plus 0, that equals 200. Not going to help you this time. Here's your binary number. I've removed the placeholders, so you'll have to add those on there. Sketch this out on a piece of scrap paper, and we'll do a countdown here. At the countdown end, I'll show you the answer. If you need to pause, pause. So this ends up being 225, and that means we take 128, plus 64, plus 32, plus 1, which is 225.

Converting Decimal to Binary

So now that we've seen how to convert from binary to decimal, it's a pretty easy process, converting from decimal into binary is a little bit more complex, it has to ask a few more questions in order to get this to work out. So if we have the value of 210 here in decimal and we want to convert this to binary, we have to start by asking some questions. The question goes like this, can I subtract 128 from the number 210 and end up with a positive number or 0? And the answer here is yes we can, and if the answer is yes, then we subtract 128 from our value of 210, that leaves 82. Now 128, that placeholder is going to be an on because we put a yes there. We'll come back to that. So let me go to the 64's placeholder and we use the number that we calculated in 128 and can we subtract 64 from 82 and end up with a positive number or 0? And the answer is yes, we can. So 82 minus 64 leaves 18. Can we subtract 32 from 18 and end up with a positive number? And the answer is no, we can't. If we subtract 32 from 18, we're going to end up with a negative number so we can't do that transaction. How about 16? Can we subtract 16 from 18? The answer is yes, we can. We are left with 2, 18 minus 16 is 2, that's a positive number or 0. How about eight? Can we subtract eight from two. Nope, it's going to end up with a negative number. How about four? Can we subtract four from two? Nope, that would end up with a -2. Can we subtract 2 from 2 and end up with a positive number or 0? The answer is yes, we end up with 0. Can we subtract 1 from 0 and end up with a positive number or 0? No we can't that, would be -1 so we put no there. So in order to convert this, we're going through this iterative process and saying, can I subtract this value? How about this one? And every time we say get a yes, we do the calculation and we end up with a new value. So here, wherever there is a yes, we put a 1, wherever there is a no, we put a 0 and this is our binary number. So 210 in binary is 11010010. Should we see that again? Let's do a different number. So here we have 47. So can we subtract 128 from 47? No. How about 64? Can we subtract 64 from 47? Nope. Thirty-two, can we subtract 32 from 47? Yes, yes we can and that leaves us with 15. So how about this? Can we subtract 16 from 15. Nope, that would leave us with -1. How about eight? The answer is yes. It's going to leave us with seven. How about four? The answer is yes, it's going to leave us with three. How about two? The answer is yes, we can leave, we can subtract two from three, that leaves one. Can subtract 1 from 1 and end up with 0? Yes we can. So for 47 now, we end up with 00101111. Whenever we're describing a binary number, if we're always describing an 8-bit binary number, it's completely okay to lead with the zeros in the beginning of the number. However, generally speaking, it's not necessary. We can ignore zeros that are in the higher placeholders. So in this case, if I wanted to say 47 in binary, I could just say 101111. And in the same way I could add a bunch of zeros in front of the 47 and say oh at 000047 in decimal and that's still going to equal 47. So we can either, in binary, we can either say those leading zeros in our binary number or we can ignore them.

Hexadecimal

So let's take a look at hexadecimal next. Hexadecimal requires us to be able to convert from binary to decimal, and then once we convert from binary to decimal, this will start to be easier to see how easy it is to convert to hexadecimal because each hexadecimal value is represented by four binary bits. So if we start counting here, the counting is going to be real easy in the beginning. So on the left-hand column there, I'm going to count up in 4 bits of binary from 0 to 8. So 0000 in binary is 0 in decimal, 0 in hexadecimal. 0001 in binary is 1, 0010 is 2, 0011 is 3, 0100 is 4, 0101 is 5, 0110 is 6, 0111 is 7, 1000 is 8, 1001 is 9, and then 1010 in binary is 10 in decimal, right? It's the 8s place holder, plus the 2s placeholder. In binary, that's 10. In hexadecimal, instead of coming up with a new number symbol, which we don't really have, instead, we're going to use a capital letter A. So if you remember back to the time where we were looking at an IP address, specifically an IP version 6 address, it had some letters in it, well, that's because those are written in hexadecimal. So a hexadecimal A is 1010 in binary, and it's 10 in decimal. We're going to find out that 1011 in binary is 11 in decimal and B in hexadecimal, 1100 is 12 and C, 1101 is 13 in decimal, D in hexadecimal, 1110 is 14 in decimal, E in hexadecimal, and then 1111, all the bits are turned on, in decimal it's 15, in hexadecimal it's an F. That is the highest value we can have in hexadecimal. So when we get to a decimal 16, our binary is going to have to add a placeholder, so it's going to be 10000, and then hexadecimal is going to count up one value, it's going to add the 16s placeholder, and it's going to be 10. So this is not ten. This is 10 in hexadecimal, which is equivalent to a decimal 16. So 10000 in binary is 16 in decimal, which makes sense because we only have 1 bit in the 16s placeholder. In hexadecimal it's 10, and that's equivalent to 16 in decimal, and what that means in hexadecimal is we have the 16s placeholder and then the 1s placeholder.

Summary

So to wrap this up, what we did was we introduced the need for binary, learning that computers operate with these digital switches that can either be a 1 or a 0, so we need that binary, we reviewed some primary school math and just looked at how we count in decimal, adding those placeholders, and then we started a count in binary using the same method. Next, we converted some binary to decimal, then we did some decimal to binary, and then finally, we looked at hexadecimal and saw that each hexadecimal value is represented perfectly by four binary bits. I hope you found this very useful. Hopefully it wasn't too complicated of a math review. This is all the information we need to understand IP addresses, as well as MAC addresses later on when we talk about Ethernet. Let's move on next and talk about IP addressing.

Introducing IPv4 Addressing

Introduction

Now that we know how to count in binary, translate binary to decimal, decimal to binary, and have a better understanding of hexadecimal, we can now move into IP addressing. So let's start this module where I introduce you to the IP address and we can see its components. First, we're going to answer the question what is an IP version 4 address? Next, we'll look at the difference between classful versus classless addressing. Next we'll look at different address types so that we can better understand how we use IP addresses. Then we're going to do a demonstration using IP version 4 addresses so we can see how they actually work and what rules we need to follow to make communication possible at the network layer.

What Is an IP Address?

So in the OSI model now, we are down at the network layer. We've covered application layer protocols like FTP and HTTP, we've talked about transport layer protocols like TCP and UDP, as well as how we address things at the transport layer, which was with port numbers. And remember the port numbers at the transport layer were directly correlated with the application layer protocol. Now we're at the network layer, layer 3, and this is where we can talk about IP addressing. Internet Protocol operates at layer 3. So what is an IP address? The IP address is a mechanism for us to be able to communicate across long distances on the internet. The network layer address is much like your postal address, the address of your home or your business. It gives you a very unique, specific address on the face of the earth where anybody can send a message to you via that address. Same thing in Internet Protocol. The IP address is a unique identifier for your device on the public internet. There are routing tables on the public internet that allow anybody to communicate with your IP address, assuming they're able to find out what it is. Generally we find out the IP address of devices by using DNS, which we've talked about briefly and we'll talk about a little bit more in a few modules. But for now, the network layer is where we use IP addressing to get messages from one device to another device on the internet. So here is an IP address, 203.0.113.10, and this address has two components to it. It has a network portion of the address, and it has a host portion of the address. The network portion of this address is very similar to the ZIP Code of our street address. So if we have the street 123 Main Street in Cityville, Illinois 60787. This is a made-up address, made-up ZIP Code, made-up city. The ZIP Code here represents an area of a city, and inside that area of the city we have specific street addresses. So in this case the network portion of my IP address represents a group of network devices. The host portion of that address represents a specific device on that specific network. So just like 123 Main Street 60787 can identify a specific location in a specific area, 203.0.113 is our network, .10 is our host. It's a specific device on that network. So let's take a look at this a little bit deeper. Our IP address construction consists of four octets. Now an octet is just a set of 8 bits, and there are 4 sets of 8 bits in this address. And what we do is we take each set of 8 bits, we convert it to decimal, and we separate them with periods, and this becomes our IP address. Now, this is not the best way to construct this. As a matter of fact, it makes for working with I P addresses a little bit messy, especially when we get into subnetting. But it's okay. As long as we understand that we can convert these 4 decimal values into 8 bits of binary, that's all we need to understand, because our IP address is really 32 bits long, and those 32 bits are one continuous string. The decimal points we only put there when we write it in decimal in order to make it easier to read and say. So how do we identify the network portion and host portion of that address? Remember we have that ZIP Code, the area of devices that are represented, the network portion, and then we have a specific host address? Well, how do we identify where the network portion and host portion are? Well, we do that in one of two ways, and and one of them is antiquated and we don't use anymore, and the other one is the only way we do it. So way one here is classful addressing, and we used that up until 1995 or thereabouts. The second method here is classless addressing, and that's what we use today, and we've been using that from 1995 until now. There was a transition period where it was a little messy, but now we exclusively use classless addressing. This can get messy, depending upon the network engineer you talk to because sometimes they talk about a classful subnet mask. Don't worry about that right now. What we're going to learn is how classless addressing works. We're going to do a review of what classful addressing is.

Classless Addressing

So classless addressing. This is what we use in modern networks. The way that we figure out which portion of the address is network and which portion is host is with something called the subnet mask. Now the subnet mask, what it's going to do here is, I have colorized the network portion as blue and the host portion as green. So when we write our subnet mask, what we're going to do is we're going to put a binary 1 in our subnet mask where we want the address to be network portion, and we're going to put a binary 0 in the part of the address where we want the address to be host portion. Now the subnet mask has to follow a set of rules that says the subnet mask is a series of 1s followed by a series of 0s. Wherever there's 1s, that's network portion; wherever there's 0s, that's host portion. So you can see in my slide here where I've put all 1s in the blue portion and all 0s in the green portion, and now we can identify which part of the address is network and which part is host. When I convert that to decimal, I get 255.255.255.0, and that becomes our subnet mask. So 203.0.113.10 with a mask of 255.255.255.0. What that means is the first 24 bits are network portion; the last 8 bits are host portion. Understanding this division between the network and the host portion is really, really important, because it is literally how we identify where devices are, and there are rules about what devices can communicate with each other, depending upon the network portion of the address. Now our subnet mask does not have to fall at 24 bits. Let's say we have this address of 10.0.0.10. Well, if I convert that to binary, you can see my binary written out there, and let's say I want the first 8 bits to be in the network portion and the last 24 to be in the host portion. Well, what I would do here is put eight 1s in my subnet mask, and then I'd follow it with 24 0s, and that would make a mask of 255.0.0.0. Now you might be asking yourself, why am I making the first 8 bits network portion in the last 24 bits host? This at this point is purely arbitrary. I am making this up so you can see that our subnet mask can change, and when we change it it changes which portion of the address is network and which portion is host. That's really all I want you to get out of this. Later on we're going to learn more about how to make subnets appropriately sized for the network we're working with. But more often than not, your job is not going to be to design the subnet mask. Rather, your job is going to be to use the subnet mask to figure out if an IP address is configured correctly and also better understand how the rules work. So let's look at another example here where the subnet mask is different. So the subnet mask does not have to fall at an 8-bit boundary. There is no rule for that at all. So here we have an address, 10.0.0.10 again, but this time I have put the first 20 bits in the network portion and the last 12 bits in the host portion. So now I have a subnet mask here of 255.255.240.0, and what this means is that the division between network portion and host portion falls right in the middle of an octet. This is 100% valid. There's nothing wrong with this. It makes figuring out what's happening with an IP address, especially when it's written in decimal, this makes it a little bit more complicated to deal with. The next module we're going to look at is called subnetting, and in subnetting I'll show you a little bit more about this and I'll also give you another course that you can look at to get a real deep dive into IP addressing and subnetting. The more you know about IP addressing, the better off you are in a career that involves data networking or network security because we can really clearly understand when devices can and cannot communicate.

Classful Addressing

We have an understanding now of how we divide our network portion and host portion using a subnet mask. Let's take a look at what we did initially. When IP addresses first came out in 1981, the design was that they would create 5 different classes of IP addresses, Class A, B, C, D, and E and they set up a rule to determine what defined Class A address, what defined a Class B, Class C, D, and E and here is the results. Class A goes from 0.0.0.0 through 127.255.255.255. B is 128.0.0.0 to 191.255.255.255. Class C is 192.0.0.0 through 223.255.255.255. E Is 224.0.0.0 through 239.255.255.255, and E is 240.0.0.0 through 255.255.255.255, which is the very last IP address we can have. These first three classes are called unicast addresses and these are the addresses we use on the internet and on our local networks in order to communicate. Class D addresses are for multicast. Multicast is when we have one server that sends out a stream of information and then there are lots of devices on the network listening to that. It's kind of like you might think a radio station like FM radio, it's kind of like FM radio where you have one transmitter sending out a signal, and then you can have this little box, this radio, and you can tune in to that station and listen to it. There is no way to send a message back to that radio system back to the radio station, that's kind of what multicast does here. The last address range here, Class E, that is experimental and we don't use it. So in classful addressing, the class of the address determines the division between the network and the host portion. So Class A addresses, the first 8 bits are network, the last 24 bits are host. Now remember this is all historical. We don't use classful addressing anymore so this is purely historical and it's part of the Net+ exam. You're also going to find some people talking about classful addressing and saying, oh, it's a Class A subnet mask. Well, okay, what that means is that there is 8 bits in the network portion, 24 bits in the host portion. Class B address Is going to have the first 16 bits in the network portion, the last 16 bits in the host portion. And then a class C address is going to have the first 24 bits in the network portion, the last 8 bits in the host portion. And then a class D address has all bits in the network portion effectively and this is our multicast addresses. So remember that classful addressing is useful to understand the history of this, but the reality is is we don't use that anymore. It was an antiquated way of handing out addresses and it forced the whole system to rethink how we hand out addresses in the late 80s and early 90's because no one really predicted the growth and the speed of growth of the internet, and had they not switched from classful to classless addressing, we definitely would have run out of IP addresses and the internet would not be what it is today.

IP Address Types

Let's talk about different IP address types here. There are a couple different types and a network address is an identifier for a group of devices, right, this is kind of like our zip code. This is also called our network prefix. So network address, network prefix, they're basically the same thing. The second IP address type here is a broadcast address and this is an identifier for all devices on a network. So you might say, well Ross, if the network address identifies a group of devices and the broadcast address is an identifier for all devices on the network, what's the big difference here? The big difference is the network address is kind of like our zip code. The broadcast addresses is what a junk mail company might use to send you some advertisements to your home address. A lot of times it'll just say resident and it might just put a zip code on the address so that the postal carrier will put that same advertisement in every single mailbox in the entire neighborhood. So the broadcast address is kind of like that junk mail you get, it allows a sender to send a message to every single device in the neighborhood all at once. Now we don't use the broadcast address very often, however, we do need to understand that it is a type of IP address because we can't actually use the broadcast address to assign to a device. A host address is the third type of address here and it identifies a unique device on the network. This is most likely what you are currently familiar with because you may have had, at one point, to tell somebody, a technical support representative or a family member who is trying to help you, what your IP address of your device is so it identifies a unique device on the network and it's a combination of a network portion of the address and a host portion of the address where the host portion isn't identifying the network address nor the broadcast. Let's look at these three addresses in more detail. The network address. The network address follows a precise rule. It follows a precise rule and you kind of have to drill this into your head to make it work. Network addresses have all zeroes in the host portion, period. So we look at the subnet mask, the subnet mask tells us the dividing line between host and network. We look at our IP address. If we have all zeroes in the host portion just like this address, that identifies a network address. The broadcast address is just the opposite, it is all binary ones in the host portion. So here we have our 203.0.113 address, our network portion of our address, and then if we put all ones in the host portion, then we get .255. This is our broadcast address, all binary ones in the host portion of the address. Remember, we identify the host portion of our address by looking at the subnet mask. Wherever there are zeros in the subnet mask, that is our host portion. If we put all ones in our IP address in the host portion, that becomes the broadcast address. The host address, which is the address that we can actually assign to a device that can't be the network address, it cannot be the broadcast address, but it can be anything in between. So here, if we look at the host portion of our address and we see that it is not all zeros and it is not all ones, that is a host address and we can assign that to a device on the network, it's anything, except all binary zeros or all binary ones in the host portion of an address. In this case, we have 00001010, that is not all zeros, that is not all ones. The secret here is making sure that we use our subnet mask simply to identify which portion of the IP addresses the host portion, and then when we are looking to see if it's network, broadcast, or host address, we are looking at the IP address and not the subnet mask. The subnet mask is just telling us where in the IP Address to look for all zeros, all ones, or anything, except all zeros and ones. So the host address is what we assign to devices. Let's do some practice here. What kind of address is this? If you need to pause to convert this to binary, give that a shot. And the answer here is if we convert that to binary, we look at the subnet mask to identify where the host portion is, we see that it is neither all zeros nor all ones in the host portion. This is a host address. Let's try another one. Pause, write it out in binary, and then I'll come back in a second and we'll give you the answer. So here is the binary. If we look in the host portion of the address, we can see that there are neither all zeros nor all ones. This is a host address. Alright, let's try another one. Is this is a broadcast address, network address, or a host address? Pause the video, do your math, and come back. So in this case, we look at the host portion, we can see that 255 converts to 11111111, which is all ones in the host portion making this a broadcast address. Let's try another one. What kind of address is this, 10.10.0.0, 255.255.0.0? Pause the video, do your math, and come back. So in this case, we put it all in binary, we look at the host portion of the address. You can see that it is all zeroes in the host portion making this a network address. Now, you might be saying, Ross, these are super easy, why are we converting them to binary? And you you may not need to convert them to binary to figure this out, but when we have a mask that doesn't fall on an 8-bit boundary, it makes it more complex. So how about this one, 10.128.224.64 with a 255.255.255.224 mask, convert that to binary and find out if it's a network address, host address, or broadcast address. So when we convert that to binary, we see that the last 5 bits of the address are host portion, and if we look at the host portion of the IP address, we will see that it is all zeros making this a network address. This is where things start to get a little bit messier if we look at them in decimal versus looking at them in binary. If we look at it in binary, the division line is pretty clear cut, you can see right where the ones end and the zeros begin in our subnet masks to find that dividing line. That's a little harder to do when we're looking at it in the decimal numbers. So how about this 1, 10.128.225.0 with a 255.255.254.0 mask. This is one that trips a lot of people up. So network address, host address, or broadcast, pause the video, and come back. So this trips up a lot of people because the division between network and host portion is just one bit off of the 8-bit boundary for our decimal conversion. So when we're looking at this address, you can see that the host portion is the last 9 bits, and if we look at the host portion of our IP address, it's 100000000, that is not all zeros and that is not all ones making this a host address that looks a lot like a network address, but this is definitely a host address. So we have to be cautious when we're working with IP addresses to not jump to conclusions about what the IP address is here. It's always important to convert this to binary, look at where the ones in our subnet mask tell us the network portion ends so that we can get a very clear understanding of what type of address this is.

CIDR and Private IP Addresses

Now there's another way to notate the subnet mask and it's called CIDR notation. Now CIDR stands for Classless Inter-Domain Routing, CIDR, and we pronounce it CIDR like apple cider. CIDR notation here works like this. We have an IP address, 203.0.113.10 with a subnet mask of 255.255.255.0. If we convert that to binary, we see the first 24 bits of the address are network portion. So what we can do is we can say, instead of saying 255.255.255.0, which is a mouthful, we can simplify this and say that any time we have 24 bits in the network portion, we can use Classless Inter-Domain Routing, or CIDR notation, and make this a /24. So a subnet mask of 255.255.255.0 is equivalent to /24, and this is just simply the length of the network prefix. The first 24 bits of the address are network portion. So the way we'd write that would be 203.0.113.10/24, and if we wrote it in binary, it would look just like what I have written there with my IP address written out in binary and then the subnet mask having 24 1s with eight 0s at the end. Next let's talk about private IP addresses versus public IP addresses. Prior to 1994, there were no private IP addresses. We just had IP addresses. But because we were running out of IP addresses, we changed to classless addressing first, which gave us the subnet mask, and then we also introduced private IP address ranges. And the idea here was was that any organization could use these private IP addresses. They would not be routed publicly on the internet, and it would allow organizations to set up very large IT infrastructures and very large networks using IP addresses without conflicting with other organizations. And this is really what saved the internet and allowed it to expand and grow as much as it did. We're later going to learn about a technology called network address translation, which allows us to take these private IP addresses, or large blocks of private IP addresses, and allows us to route out on the public internet by translating it into a public IP address for a moment while it routes across the public internet, and then it gets translated back into a private address when it reaches our internal network. The three ranges of address here, what they did was they picked ranges from each one of the classes. So we have a class A here with a 10.0.0.0 through 10.255.255.255. We have a class B address, 172.16.0.0 through 172.31.255.255, and a class C at 192.168.0.0 through 192.168.255.255. And this is all documented in something called a Request for Comments, or RFC. Now, all of the protocols on the internet that we use, or nearly all of them, are written and publicly available via these RFCs. As a matter of fact, you can do a Google search for RFC 1918. So if I just do a search for RFC 1918, we can click on this first one that's at tools.ietf.org. IETF is the Internet Engineering Task Force, and they are the maintainers and approvers of RFCs. Anybody can write an RFC and anybody can submit it. Whether it gets published or not is up to the IETF. Some of these RFCs have been out there for a long time and have never been approved. And there's lots of text in these RFCs. Let's make the text a little bigger here for us. And you can scroll through and read this. I am not going to read this to you. This is late night sleepiness. But here we go, here is our private IP space, and it's listed right here in the RFC that's been approved by the IETF. We can see that happened in February of 1996. There are other protocols in RFCs, including IP. If we look for RFC for IP, we'll see that it is RFC 791, and that came out in September of 1981. And if we scroll through this, you can see all kinds of information about the construction of IP addresses and how the protocol is supposed to work. If you look at that private IP address space with CIDR notation, we have 10.0.0.0/8, 172.16.0.0/2 and 192.168.0.0/16, in the next module we're going to learn how to subnet these into smaller networks so that you don't have to use this massive chunk of a network address in one single network. There's also another network here called the APIPA network. APIPA stands for automatic private IP addressing. It's something that Microsoft and other organizations have implemented. In my opinion, this should never be used. When you see a device with this address on it, it's an immediate red flag for me to know that somebody doesn't know what they're doing or something is broken. Boot up a computer and it gets this IP address, it means that something is wrong and it should be corrected. If you're using this address in your home network, you probably shouldn't do that. It's usually a good indicator to people trying to troubleshoot that something isn't configured quite correctly. So we should definitely avoid using that space. There's another special IP address called the loopback address and it's 127.0.0.1, and we can send messages to this address. When we send messages to this address from our workstation, the messages never leave our computer, and it's generally a way to test to see if IP is working correctly on our workstation.

Demonstration: Modify and Test IP Configuration

So with all of that, let's do a demonstration and modify and test IP configuration on a device. Let me give you some information about what we're going to do first though. What we're going to look at is connecting two devices together. Now two devices when we connect them together need to follow some specific rules in order for messages to be sent between them and the rule is is that the network portion of the address must remain the same for both the devices that we are communicating with. So in this case, we have 192.168.10.10 and 192.168.10.100, both with 24-bit masks. So what this means is the first three octets are the network portion and those network portions align for both devices. So when we have 192.168.10.10 is on the same network as 192.168.10.100, meaning these 2 devices will be able to communicate with each other. If I were to change the IP address of the first device on the left there to 192.168.11.10, now the third octet here no longer matches and these 2 are on different IP networks, and in order for them to communicate, we'd have to put a router in between them. Since we're not using a router, these two devices should no longer be able to communicate. The last thing we can do is change the subnet mask of 192.168.11.0 to be 255.255.254.0. Make that mask change on both devices. And now if we convert our IP addresses to binary, we will find out that both these devices are on the same IP network and now they will be able to talk to each other. So let's go do this on some real workstations. So I'm on my Windows 10 workstation here and I actually have 2 Windows 10 workstations. This is the screen for the first one with the gray background with Pluralsight. The second workstation is this one here with the tent lit up in the night sky here. So this is the one that I'm going to assign the .100 address and let's do that right now. So we're going to go to Control Panel and select Control Panel here, we go to Network and Sharing Center, click on Ethernet0 here, go to Properties, click on IP version 4 here, click Properties, and then we click on Use the following IP address and I am going to use 192.168.10.100 and that's going to have a 255.255.255.0 mask. We'll hit OK here, Close, Close. I'm going to ignore that Windows firewall message because we're not even connected to the internet right now. So I just want to be able to send a message between these two devices. So right now this device has already been configured. Let's go back to the original workstation here, and here, we can configure the IP address as well. We go to Control Panel, Network and Sharing Center, click on Ethernet0, select Properties, IP version 4, and use the following IP address 192.168.10.10, in this case, with our 24-bit mask of 255.255.255.0. Hit OK here, OK and Close. We can now open up our command prompt. If I type in cmd, that'll open command prompt for me, and now I can run the ipconfig command and verify that that address is correct here, 192.168.10.10. And now there is a utility I can use called ping and ping stands for packet internet groper, which sounds bizarre, but it's actually what it's called. And what ping does is it allows us to put in an IP address of 192.168.10.100, which is the other workstation, and it'll let me verify that these two devices are connected together. Now I have a switch connecting these two devices together specifically so they can reach each other. And here we get a reply now from 192.168.10.100. So what this means is that we have a successful connection between my workstation and the 192.168.10.100 workstation. Now let's go and change our IP address to be 192.168.11.10 to see what happens. So we go back to Control Panel and we go to Network and Sharing Center, click on our Ethernet connection, click on Properties, IP version 4, and then we're going to change this from a .10 to .11, and we're going to hit OK, OK, and Close. And now when I try to ping that same address, I can actually just hit the up arrow and that'll automatically pull up the last command that I typed, and hit Enter, and now we get a message that says woah, transmit failed. General failure. What this means is that it's saying, hey you're trying to ping an IP address that isn't on your local network. The network portion of the address of 192.168.10.100 is not in the range of your IP address, which is 192.168.11.10. Since they're not on the same network, my workstation would typically send this to the default gateway. The default gateway is actually a router, and if our network addresses are different, our workstation will send the traffic to the router and the router can figure out what to do with it, but since I don't have a default gateway configured, when I try to ping 192.168.10.100, I get this general failure because it doesn't know where to send it, it has no place to go with it, it just says I can't do this. I'm abandoning the process. Let's do one more experiment here where we change now our address to have a different subnet mask. So I'm going to change my mask to a 23-bit mask here of 255.255.254.0 and we're going to have to go on the other workstation now and do the same thing. So I can verify this by typing ipconfig and see that right now it is 192.168.10.100 with our 24-bit mask. Let's go to the Control Panel, Network and Sharing Center, Ethernet, Properties, TCP/IP, or excuse me, IP version 4. And we're going to change this to be a 23-bit mask as well, hit OK, hit Close, and Close. Let's verify that the address is correct here. We have 192.168.10.100 with our 23-bit mask. Let's go back to the original workstation now and send our ping message. So now I'm back on the first workstation here, we can verify our IP address by typing ipconfig, see that it's 192.168. 11.10. 192.168.11.10 and 192.168.10.100 are on the same network portion of the address if it's a 23-bit mask. So now I should be able to ping 192.168.10.100 successfully, and I can. So hopefully this illustrates the rule for you now that two devices need to have the same network portion in order for them to send network layer information between them. If the network portion of the address is different, we need a router. We need a default gateway to send that traffic to. We're going to learn more about routers and routing in a future course. For now, just understand that this is a hard and steadfast rule that you can use almost immediately to start helping troubleshoot networks.

Summary

So to wrap up here, we looked at what is an IP version 4 address, saw that it is a 32-bit number broken up into 4 octets written in decimal format. We looked at the difference between classful and classless addressing, noticing that we use classless addressing exclusively today, although sometimes you may hear people use the word classful subnet mask. We don't worry about that too much. All subnet masks are classless by definition. We looked at address types and saw the difference between a network address, host address, and broadcast address. And then we did a demonstration to illustrate the rules of IP addressing, knowing that two devices must be on the same IP network in order for them to communicate with each other. If they are not the same, we need a third device called a router to route the traffic in between it. I hope you found this module useful. Let's move on and talk about subnetting of IP addresses.

Subnetting Networks

Introduction

Now that we have a good understanding of binary, converting binary to decimal and decimal to binary, we've looked at IP addresses and understood how they're constructed, we looked at the subnet mask and learned how the subnet mask tells us which portion is network, which portion is host, now we can move on to this idea of subnetting networks. And when we talk about subnetting networks, we're talking about subnetting IP networks from a large network into smaller components. Our goals for this module will be first to review the address types. And I know I really spent a lot of time in the previous module talking about the different types of address, the network address. host address, broadcast address, but honestly, that is so important to understanding how subnetting works. You need to have that mantra in your head about what each one is, so we'll review that before we get into subnetting. Next, we're going to break networks into smaller networks, which is the exercise of subnetting itself. We'll talk very briefly about Variable Length Subnet Masks, or VLSM. And then last, I'll introduce you to a course you can take that will help propel your knowledge about IP addresses and subnetting to a new level. This course has all kinds of exercises you can do as well, and it's actually a course, in my opinion, that if you're working in IT, everybody should know this level of IP addressing. The math can get a little complicated, but with some practice, it ends up being not so hard.

Subnetting Networks

So the components of an IP network. In order to have an IP network, we have three components to it. We have that network IP address, which is all binary 0s in the host portion. We have our broadcast IP address, which is all binary 1s in the host portion. And these two addresses, remember, define the beginning and the end of a range of addresses in an IP network. Everything in between all 0s and all 1s is our host IP addresses, and these are the addresses we can actually assign to devices. So if we take a look at our private IP space here, 10.0.0.0/8, we can rewrite this in the dotted decimal format of 10.0.0.0 with a 255.0.0.0 mask. Remember that /8 is just our CIDR notation. Now this network, or this subnet even, is a very large subnet. There are 16,777,214 host addresses on this network. Now this, we could accommodate a very, very large network; however, we typically segment our network into smaller components, and I have never worked for an organization or saw an organization that had a /8 subnet that they were using for the devices on their network. They've always broken it up into smaller pieces. It makes it more manageable. And later on, we're going to learn more about Ethernet and how Ethernet works, and Ethernet really doesn't allow us to have 16 million devices on a single subnet unless we do some sophisticated things. So what we want to do is break this 10.0.0.0/8 subnet into smaller components. So the range that we have to work with here is we can break this up in any way we want. We just have to stay within the constraints of 10.0.0.0, our network address, through 10.255.255.255, our broadcast address. So here that is converted to binary. And if I draw a line between our network and host portion, it'll be at the 8-bit boundary so that our first 8 bits are network, the last 24 bits are host portion. So the best way I can describe how we're going to subnet this is let's find a host address on this network. And I didn't pick this address arbitrarily. I picked it for a reason, and we'll see it in just a second. So here I have this host address of 10.0.10.0, and right now I have a /8 mask applied to it. So the first 8 bits are network, the last 24 are host still. We can tell that that is definitely a host address because if we just look at the host portion of the address, we can see it's not all 0s and not all 1s. Now what if I did this? What if I moved my subnet mask from /8? What if I moved it down over two /24 and I apply the

/24 just to this single address of 10.0.10.0? Well, when I do that, now if I look at the host portion of my address, the last 8 bits, there's all 0s in there now. So now suddenly if I apply a /24 mask, now 10.0.10.0 becomes a network address. Now you might be asking, where did I get /24 from? I literally just made it up. I pulled this mask out of a hat for this exercise to say, hey, we can take this /8 network and we can carve out multiple /24 networks. And all I have to do is pick an address here that has all 0s in the last 8 bits, like 10.0.10.0, apply my /24 mask, and now I end up with a much smaller network. As a matter of fact, this is a lot more manageable now because now I have from 10.0.10.0 through 10.0.10.255. This allows me for 254 different hosts. Zero to 255 is 256 unique addresses. If I remove two of them, one for the network address, which I can't apply to a device, one for the broadcast address, which I can't apply to the device, I end up with 254 addresses I can apply to host devices on my network. So now I have a carved out portion of 10.0.0.0/8. So if I graphically represent this, this orange bar would represent the entirety of the 10.0.0.0/8 network, and my 10.0.10.0/24 would be this small slice of it here that I've represented with that purple bar. So here I have just a small piece of the 10.0.0.0/8 network. I don't have to use 10.0.10.0/24. I can actually use almost any combination I want here as long as the last 8 bits of the address are 0 so that I can use that as the host portion of my addresses. So if I make a chart here, this is just a portion of what I can submit 10.0.0.0/8 into if I were to use a /24 mask. I can have 10.0.0.0/24, and that would be 10.0.0.0 as my network address through 10.0.0.255 as my broadcast address, and the next available network after that, or next available IP address, is 10.0.1.0 and then 10.0.2.0, 3.0, 4.0, and so on. We can continue to count these subnets up from 10.0.0.0/24 all the way through 10.255.255.0/24. So really all we're doing here with subnetting is we're breaking up this large network into a bunch of smaller ones and just counting them out. Now I do not have to always use a /24 mask. What I can do here is I can use different sized masks for different sized network applications. So here with 10.0.0.0/8, I can have 10.0.10.0/24, 10.0.16.0/22, 10.1.0.0/16 or 10.2.0.0/30. These are all valid subnets of 10.0.0.0/8. There's some trickery to this though. We can't just randomly and arbitrarily pick IP addresses and masks and apply it to networks. We do have to make sure that there's no overlap between our subnets. This is called Variable Length Subnet Masking, and this is a more complicated way to do subnetting, but it is one that we see in the real world a lot. So what we can do if we really want to learn how to do this subnetting itself, I highly recommend taking this Network Layer Addressing and Subnetting course. It is part of the CCNA learning path. This is of course that I authored. It has lots of exercises in it. It has basically a deep dive into IP addressing and subnetting. You'll get all the methodologies, the tips, the tricks, and a bunch of exercises you can practice for yourself to really understand how IP addresses and networks work. This is one course, in my opinion, as an IT professional that you should really take because it is going to give you skills that are very valuable in IT, especially in data networking.

Summary

To wrap up what we've done here, we looked at the address type review and we reviewed that the network address has all 0s in the host portion, broadcast address has all 1s, and the host address is everything in the middle. I've repeated that a lot, so that must be important, right? Next, we looked at breaking networks into smaller networks, looking at that 10.0.0.0/8 network and breaking it into smaller ones. We looked very briefly at Variable Length Subnet Masks, or VLSM, to see that we do not have to apply the same mask to every single network. And then I introduced you to the course, Network Layer Addressing and Subnetting, which I highly recommend you go take a look at. You have all the skills right now to perform well in that course. It really requires you to understand binary and decimal and this basic structure of the IP address. If you have that, you'll end up leaving that course with a extremely solid understanding of IP addressing and subnetting. Let's move on to the next module where we continue to talk about IP addressing, but this time, we're going to talk about IPv6 addresses, which are going to look wildly overwhelming in the beginning, but we're going to find out that they're actually a little bit easier to work with than IP version 4.

Introduction to IPv6 Addressing

Introduction

So far, we've covered a lot about IP addressing, but that was all specific to IP version 4. Now we need to talk about IP version 6. Now this can be a little overwhelming at first, but we're going to find out that the rules between IP version 4 and IP version 6 are generally the same, we just have to deal with a new format of address that's much larger than our IP version 4 counterpart. Our goals for this module will first be to review some terms to make IPv6 understanding a little easier, then we're going to talk about the difference between the size of the IP version 4 address versus the IP version 6 address. I'll then introduce IP version 6 addresses, we'll talk about how they're written and how we can simplify how they're written. We're going to look at how they operate and what rules they need to follow in order to communicate. We're going to look at different IP version 6 address types. There are more address types in version 6 than there are in version 4 and it has to do with some of the way IPv6 operates, which we're going to take a look at that, as well as how IPv6 gets addresses assigned to the interface. We actually have more options in IP version 6 than we did in IP version 4. And then we'll wrap this up by talking about IPv6 tunneling, which we can use when we have some IPv6 addresses on a network that we want to talk to another IPv6 network, but there is not an IPv6 connection between them.

IPv6 Address Components

How about some terms here? Let's start with bit. We've talked about a bit before. It is a single placeholder that can either be a 0 or a 1. A bit is a binary placeholder here. It can either be 0 or 1. If we have 4 bits, each 1 of those 4 bits can be 0 or 1. And when we have 4 bits, we have a term for that, and it's called a nibble. So, 4 bits is a nibble. A nibble can be written as a single hexadecimal value. Each hexadecimal value is written in binary with 4 bits. There are 16 different possibilities, and in hexadecimal, we count from 0 to 15 before we add another placeholder. Remember, we use letters for any number value above 9. So, here A, B, C, D, E, and F are all hexadecimal values, and I've written it as 0xA. And we can identify hexadecimal values often because there is a 0x in front of it that indicates that the number that follows that is hexadecimal. So if you encounter documentation and you see a bunch of letters and numbers written together, if there's a 0x in front of it, there may or may not be one, but if there is a 0x in front of it, that's definitely hexadecimal. Next, if we have 2 nibbles, that's 8 bits. Well, there are 8 bits in a byte. So, this is some IT nerd-ary humor here; 2 nibbles is a byte, and a byte is 8 bits. So in our 8 bits here, we can assign that 8 bits any value we wish between all 0's and all 1's. If we were to write this in hexadecimal, hexadecimal now would be 2 hexadecimal values is equivalent to 1 byte of data. So, here the number A8, the hexadecimal value A8, that is 1 byte worth of information, those are 2 hex values. Each hex value is 4 bits. You put them together, you have a byte. Now in IPv6, we typically work with values in 16 bit sections. So, this is 2 bytes, and we've coined the term hextet to represent 16 bits of data. So we're going to use our hextet here a lot in IP version 6. Hextet can be written as 4 hexadecimal values, which is equivalent to 16 bits. Each hexadecimal value, again, 4 bits, 4 times 4 is 16. So now that we have some terms out of the way, let's talk about the IPv4 and v6 address size. So, an IP version 6 address is 32 bits long. We write that as 4 octets, or 4 bytes worth of data. Each octet is 8 bits. We convert that to decimal to work with our IP addresses. This is unfortunate because IP addresses are much easier managed in hexadecimal, but we write them in decimal just out of tradition. If we convert that to binary, we can see that there are 4 sets of 8 bits. That is 32 bits of address space. An IP version 6 address is 128 bits long. It has four times as many bits as an IP version 4 address. This is an enormous amount of bits for an address. We write that with 32 hexadecimal values, or 32 nibbles, and we separate it out into 8 hextets, and each hextet is separated with a colon. So here, my IP version 6 address is 2001:0DB8:0002:008D:0000:0000:00A5:52F5. That is a mouthful. It is a lot to read off, and we're going to learn how to make this address simpler, actually. If we were to write out the binary for this, this number is quite large. If we had to convert these to binary all the time, this would get messy because it's 128 bits. It's a lot of binary values in order to write down to work with this address, which is why we work with it in hexadecimal. And there's some other tricks here that are going to make it a lot easier to work with compared to our IP version 4 addresses with subnetting. So, in IP version 6, typically we use a /64 mask on everything. There are occasions when we don't, and sometimes when we look at routers and routing tables and things, we may see something different than a /64 mask. But when we're working with IPv6 addresses on our workstations, typically we're going to see a 64-bit network portion of the address, and then a 64-bit host portion, which we call the interface identifier portion in IP version 6. So we have a network portion and an interface identifier portion. So, our IPv6 address is messy to work with, to write down, to communicate to somebody, to enter into a device. And there are actually some tricks we can use here to help make this address easier to read, write, configure, and work with. So this looks a little daunting, and I realize that. Let's make it simpler. So, first thing we can do is we can eliminate leading 0's. So, when I say leading 0's, what do I mean by that? Well, in leading 0's, if I were to give you a dollar, or I were to give you $01 or $0001, all those are a dollar. It doesn't matter how many 0's I put before that 1, that does not change the value of the amount of money I'm giving you at all, versus if I were to give you a dollar and then add a 0 to the end of that, that will be $10, or add three 0's to the end of that, that's $1000, those are not all equivalent to a dollar. So if I put a 0 in front of a number, it doesn't change the value. If I put a 0 after a number, it does change the value. So, when we are working with our IPv6 addresses here, if I want to simplify this, I can take away any leading 0 in each hextet. So here I have 2001, no leading 0's there, but 0 DB8, there's a 0 there that I can get rid of and just write as DB8; 0002, I can write that as 2; 008D, I can write that as 8D; 0000 I can write as a 0, and I can do that twice here; 00A5, I can write that as simply A5; and then 52F5 are my last 4 hex values. So now I've simplified that address down. My 64-bit boundary still exists here. I've taken some of the values out, but each hextet here, every space between the colons, is representative of 16 bits of information. So, I can quickly and easily find the dividing line between my network and interface identifier portions by just counting the hextets. So I count 4 hextets, and that is 64 bits. Now, there's another way I can simplify this, and I can eliminate series of 0's with a double colon, which is just 2 colons side by side. So here what I can do is I can take this series of 16 0's, and I can collapse that down into a double colon. So here this long address that I had in the beginning suddenly becomes 2001:DB8:2:8D::A5:52F5. And you may say, wow, that's still a mouthful and that's still a lot. It is. However, what we're going to end up finding out is that those first two hextets are typically never going to change. So in our entire network environment, all the networks are going to start with 2001:DB8, or whatever 2 hextets are assigned to your organization for IPv6. So, when we're working with IPv6, oftentimes the first 2 hextets are going to be identical everywhere in your environment, so we're really just working with the last 2 hextets in the network portion, and then whatever interface identifier portion we have. The double colon typically is going to fall at 64 bits. It doesn't have to, but it typically does when we're working with IPv6 addresses. There's some potential errors here with the double colon. We can only use that double colon once in an IPv6 address, and the reason is is that we have to know how many bits, exactly how many bits are being replaced with that double colon. And we can only do that if we only have one of them, so we can know where it goes. So here, this is a valid IPv6 address, but if I do this one and I put 2 double colons in, I change the address a little bit here, so I have 2001:DB8:8D::A5::52F5, with double colons around the A5, I can't do this. My workstation and the network will have no idea what this means because I don't know if it means this, 2001:DB8:8D:0:A5:0:0:52F5, or if it means this, 2001:DB8:8D:0:0:A5:0. I don't know which one this this replaces. So we can't use an address with two double colons in it. Another one we have to watch out here for is if we forget to put the double colon in, in this case, we don't know where that double colon goes, so we have no idea where to divide the network and the host portion. We don't know what the host portion actually looks like. There's actually not enough bits here to make an IPv6 address.

IPv6 Address Operation

Let's talk about IPv6 address operation now. So, this is going to work very similarly to how we saw IP version 4 work. In fact, I did this exact demonstration in a previous module where I connected 2 PCs together, and I had them both on the same IP subnet of 192.168.10.0, and we were able to send a ping message between them when I changed the network address of 1 of these devices. In this case, I'm changing the PC on the left to be 192.168.55.10. Now the network portions don't match anymore, and in order to get these two devices to communicate directly, I'd have to install a router, so these guys are not going to be able to talk with this configuration. Same thing is true with

IP version 6. When the network portions of the address are the same here, in this case, 2001:DB8:4:A, this means that these 2 devices can communicate with each other without any additional hardware. However, once I change one of these to be on a different network, in this case, I'm changing the PC on the right to be 2001:DB8:4:B, well, now these 2 devices are no longer on the same network, so now they can't communicate unless I add a router to the mix. When we are communicating between devices on the internet, the internet is full of routers that can route IP version 6. So when we are communicating with devices on the internet, we are using something called IPv6 unicast addresses. Now, IP version 4 has unicast addresses as well, and those were the addresses that fell in the old school class A, B, and C. Those are all unicast addresses. Class D addresses, remember, were multicast. So here in IPv6, most of our addresses that we're going to work with are going to be these IPv6 unicast addresses, and these addresses are used for global communication, and by global I mean communication on the internet with devices that are not local to your internal network. So, these addresses are for global communication. There are another type of address that we can have in IPv6. As a matter of fact, we must have this address to get IPv6 to work, meaning that every single device in our network is going to have 2 IP version 6 addresses. One of them is going to be the IPv6 link local address, and this address is used for local communication. It's always going to start with FE80, and it's going to have some identifier after that. And this link local address allows IP version 6 to communicate with other devices on the local subnet. This address is used because IP version 6, unlike IP version 4, sends out periodic messages to find out who's on the network using IP version 6 to send out information about how to obtain addresses to send out information about what network, what unicast network they're on, among other things. So it uses this link local network in order to communicate IPv6 information, and then we use our global unicast IP addresses to communicate with devices when we want to transfer data between our devices or out onto the IPv6 internet. So in IP version 6, we have these unicast addresses, we have the global unicast address, and the link local address. These are both unicast addresses. The link local address, though, is going to have that FE80. It's going to have a mask of /10 by default, and that address is going to be automatically assigned to every single device that's using IP version 6. There's also a loopback address. If you remember, there's an IP version 4 loopback address of 127.0.0.1. Here in IP version 6, it's ::1, which just means 127 binary 0's, followed by a single binary 1 with a /128 mask. So the loopback address is ::1. I don't expect you'll use that very often, but you may need to know what that is. The ones that we're going to be using most often when we're working with IP version 6, at least in modern current networks, is going to be these unicast addresses, both the global unicast address, which will let our device communicate with the IPv6 internet, and the link local address, which will allow our device to communicate IPv6 information within each subnet. There's a couple other address types that are important here. One is the multicast address. Multicast address is one to many communication. IP version 6 has capacity to open this up for multicast to be used on the public internet. In IPv4 we do not have a public way to use multicast addresses. All multicast is on a local network not connected to the internet. However, at some point in the future, we will have IPv6 multicast available to us. Another address that we have here is anycast addresses. Anycast is where we use 1 IPv6 address for many devices, and this allows for load balancing of devices on a network where we may have multiple servers serving up content to a user, maybe for a website, and we use anycast in order to route traffic to the lowest utilized server in our network. Likely, you will not encounter too many multicast or anycast addresses, at least not with the net plus certification. Once you get into more advanced engineering in IT, you will more likely see multicast and anycast being used. So, how many IPv6 addresses are there? The inventors of the internet had no idea that the internet was going to explode in the way it did. IP version 4 was written in 1981. And in 1981, I don't believe engineers then could conceive of the possibility that everybody, or nearly everybody on the planet would have a computer in their pocket that needed an address. So, how many IPv6 addresses are there? Well, engineers did not want to run out this time. So if we take a look just at the interface identifier portion of our address to tell us how many host addresses are available. Remember, there's still 64 bits in our network address; we're just looking at these 64 bits in our host portion or interface identifier portion. With 64 bits, there are 2 to the 64th possibilities here. Two to the 64th, that is, I actually had to look up all these numbers to get it right, and I'm going to read this number off, and in some areas of the world they use different terms for this. That's okay. I'm using the U.S. English version. So we have 18 quintillion 446 quadrillion 744 trillion 73 billion 709 million 600 thousand addresses. This is an obscene number of addresses. As a matter of fact, I can't wrap my head around that. I actually did some research to figure out what this means. I don't think a human brain is good with a number this large. So, what does this mean? What this means is that we could give to each star in the Milky Way galaxy 184 million IP addresses. Now, if we did it for every insect on earth, we would get 2 IPv6 addresses for every bug. So there's about 9 quintillion bugs on the earth, and we could give each of them 2 IPv6 addresses just for this 1 single network. Just for one single network, each bug could receive two. There's about as many grains of sand apparently as there are insects, so we could have each grain of sand could get two IPv6 addresses as well. This is just a really absurd way to think about it. I was just at the beach recently and there was just an obscene amount of sand there, so to think that every single grain of sand could have 2 of these addresses, and we would only use up 1 single IPv6 network, it's really impressive how many addresses we have available here. So this is 18 quintillion host addresses for each IPv6 network. I don't expect that we'll ever use those up. However, it gives the engineer tremendous flexibility when designing networks.

Demonstration: Examine IPv6 on a PC

So let's do a demonstration now. What I'd like to do is examine IPv6 information on a workstation, as well as examine what dual stack is on a PC. Dual stack simply means that we are running both IP version 4 and IP version 6 on a workstation. If we can use IPv6, we will. If IPv6 is somehow not available for the device we're communicating with, we will revert back to IPv4, in that case. We'll take a look at both of these on an actual workstation. So I'm back on my Windows 10 workstation. What I want to do is open up a command prompt and we're going to issue the ipconfig command and that will show me my current IP configuration. I have not changed the configuration since the previous lab that we did where we were trying to ping between two devices by changing the subnet mask, that was earlier in this course. Now right now if I issue ipconfig, I should be seeing my IPv6 configuration information. However, when I'm doing labs, I find that by enabling IPv6, sometimes if I don't have it set up exactly right, it breaks what I'm trying to do, so I usually leave that off. So let's look at how to turn on IPv6 and configure an address. So I'm going to open up Control Panel and we're going to browse to my Network and Sharing Center just like we're going to change the IPv4 address here, so we click on our Ethernet adapter, then Properties, Internet Protocol Version 4, and if I scroll down a little further, we have Internet Protocol Version 6 and there is no checkbox there right now, so I'm going to put a checkbox there and select Properties, and inside of here is where I can configure my IPv6 address. Now, I don't currently have an IPv6 router in my network, and because of that, I won't automatically configure an IPv6 address so what I need to do here if I want this to work is manually enter one. So I'm going to enter 2001:db8:10::10 and then it has a 64-bit mask. Now you might ask, where did it come up with that address? 2001:db:8, this is a documentation address for IPv6. IPv6, unlike IPv4, does not have private addresses. All the addresses are public, and because of that, we really don't have any type of way of translating addresses from private to public. In the protocol for IPv6, they set aside this address space 2001:db8 to use for documentation purposes so that people aren't publishing public IP addresses. We can use it to transfer information locally, we just can't access the public internet with it so this would be perfect for this exercise. So I'm going to hit OK here and Close and Close and close. So if I reissue the ipconfig command. now we will see my IPv4 and my IPv6 information. So right here, we can see I have my IPv6 address that I configured, the 2001:db8:10::10, and then right below it, you can see my Link-local IPv6 address, fe80:: and then this long string of hex characters. That fe80 address is the link local address and it allows for local communication usually using just the IPv6 protocol in order to send protocol specific information between devices. And then my IPv6 address that I've configured, that would typically allow me to communicate with the public internet. Since I'm not connected to the public internet, I can't actually use that address to reach anything publicly, however, I do have another device on the network here, 2001:db8 and then we put in 10::100 as the other device, so I'm just changing the interface identifier portion of this to be 100, which is what I've configured my other device to be, and if I hit Enter here, it automatically figures out that this is an IPv6 address and it is going to use IPv6, instead of IPv4, to send the ping message, and then I get a reply from the other workstation saying yes, I am here. I can also ping on IPv4, that address is still 192.168.10.100 and I get a response from that as well. So what we're working with here, whenever I configure both whoops, ipconfig, whenever I configure both IPv4 and IPv6, we call this dual stack, meaning that I can run both protocols simultaneously. Now when I'm communicating with the public internet and surfing the web, what Windows is going to do is it's going to attempt to use IPv6 first and if it can use IPv6, it will. Sometimes this can cause some messiness, sometimes it works beautifully. However, the point here is that if we have IPv6 enabled and there is an IPv6 router in our network, Windows is going to prefer IPv6 over IPv4. The way that we get around that if we want to strictly use IP version 4 for learning or for labs or for some other reason is to turn off our IPv6 protocol like I showed you in the beginning here. So this is how we look at our IPv6 information in Windows, it's also an examination of how dual stack works when we have both IPv4 and IPv6 configured.

IPv6 SLAAC

So, let's talk about different mechanisms for IPv6 address acquisition. Now we've already looked at one of those, and that's to manually configure it. There's 2 other options we have here for IPv6. One of them is called SLAAC, which is actually stateless address auto-configuration. This is a feature that does not exist at all in IP version 4, but in IP version 6 it does, and the way it works is like this. Let's say I have a router, a default gateway or router in my network, and it has an IP address of 2001:DB8:4:A::1. What that router is going to do is periodically it's going to send out something called a router advertisement. And in that router advertisement, it's going to advertise what network this particular segment of the network is. So it's going to advertise the network address, so the IPv6 network address, as well as some other information that our endpoints or our workstations can use to automatically configure themselves without asking. So what will happen here is we'll send this router advertisement, or RA, out into the network. That happens periodically, so we don't have to wait for a new device to come on. That router is just sending those messages out regularly. What will happen then is the workstation will then automatically configure its address. So it now knows that it has network 2001:DB8:4:A with a 64-bit mask, and now the way that the interface identifier portion is configured depends upon the operating system we're using. So if it's Windows, it's going to choose a random 64-bit interface identifier, so it's going to put our 2001:DB8:4:A, the first 64 bits, and then it's going to randomly assign the next 64 bits of information to become the interface identifier. And it's actually going to create two of these addresses to apply to our interface. Just like you saw earlier, we had our manually configured IPv6 address in Windows and then we also had that link local address, which means there's always at least 2 addresses on our workstations in order for them to use IPv6. We can configure more than two addresses besides our link local address on those interfaces. So when Windows chooses this random identifier, it actually chooses two of them and applies them to the interface. Now, if we're on Linux or Unix or Mac systems, including a lot of smartphones, an Android, an iOS, it doesn't really matter, they're all one of these flavors of operating system typically, we're going to use something called a modified EUI-64 address. And what we're going to do is we're going to take the MAC address of our network interface card. Now, the MAC address, we haven't talked a ton about that, but if you remember from the module where we talked about encapsulation, we talked about how at layer 2 we put all of our data in the payload and then we address it with a source and destination MAC address. MAC addresses are layer 2 addresses, and our hardware addresses for the specific network interface card that we're using, that could be wireless, it could be wired, and we're going to talk more about MAC addresses in the future, but for now, remember, that MAC addresses are layer 2. IPv4 and IPv6 addresses are layer 3, and then up at the transport layer, remember we address things with port numbers. So here with the MAC address, what we're going to do is we're going to grab that MAC address off of the network interface card, and we're going to split it into 2, and then add FF:FE in the middle. And this will make the address 64 bits long. Traditionally, MAC addresses are 48 bits long, so by adding FF:FE in the middle, that adds 16 additional bits to our address, making it 64 bits. So now we have a 64-bit interface identifier portion, but with modified EUI-64, we do one more thing. We take the first 8 bits and we convert them to binary. Then, what we do is we take the 7th bit in that list and we flip it, so if it's a 0, we make it a 1, if it's a 1, we make it a 0, and then we convert it back to hexadecimal, and now this becomes the interface identifier portion for Unix/Linux/Mac workstations. So we just add that into our network portion of 2001:DB8:4:A, and then the rest is basically our MAC address. The challenge with this is is that if we're using the MAC address in our host portion of our address, or our interface identifier portion of our address, what this means is that anybody on the public internet that is receiving traffic from us will know what our MAC address is. What we're going to find out is that that MAC address can identify the brand of the hardware we're using, as well as the device type if we have the right kind of table. So we can actually figure out what computer somebody's using to connect to the IPv6 internet with this type of address. So, although there is somewhat of a concern here from a security perspective, we don't worry too much about that in our day-to-day operations with IPv6. There are other ways to solve that problem, which we're going to look at next here. Whenever we're using IPv6 SLAAC, once our router sends out that router advertisement, we then configure our address and then we send a message back out to the rest of the network saying, hey, everybody, I am on the network now, this is my IPv6 address. Since some of those addresses are configured randomly, we're going to double check to make sure that no one else has that address. If somebody else has that address, there's mechanisms in IPv6 to sort out that duplicate address and have our workstation re-choose a new address.

IPv6 DHCP

So now one of the ways we can solve the problem of using SLAAC and having it automatically configure an address using our MAC address is to use IPv6 DHCP. So DHCP here is going to work very similarly to the way IP version 4 does. We're going to look more at DHCP services in another module of this course, but for now just know that, instead of the address automatically configuring itself, we add a DHCP Server into our network. When our device comes online, it's going to send out an advertisement saying, hey, I need an address and our DHCP Server will reply with one. In order to do that though, we do have to set our router up to turn off SLAAC, so that is a feature of IPv6 and we can actually turn it off in the router advertisement telling our client not to use SLAAC to configure itself and instead use the DHCP Server. This allows for a lot more control, it allows for shorter addresses, especially in our internal networks. There is a lot of value to using an IPv6 DHCP Server. Last thing we're going to look at here is tunneling IPv6. One of the problems that we have with IPv6 is not everywhere supports it, not all ISPs have full support of it, it is not used globally in the same way that IPv4 is. So let's say that we have some IPv6 internet here locally and we want to connect to a device far away that has an IPv6 address. So what would happen here is that in order to make this work, we're going to have to find a way to get our IPv6 internet traffic across the IPv4 internet and those 2 aren't backwards compatible with each other. So what we do instead is we build something called a tunnel and a tunnel is nothing more than a mechanism where we can take an IPv6 message, an IPv6 packet, and we can put it inside of an IPv4 packet, move it across the IPv4 internet to another device where it will be pulled out of the IPv4 packet and then put back into an IPv6 message and it can be forwarded. This tunneling allows our IPv6 devices to traverse the IPv4 internet and access a device that has an IPv6 address.

Summary

To wrap up what we've talked about in this module with IPv6 as we looked at the terms for IPv6 talking about a bit, a nibble, a byte, and a hextet. We looked at the IPv4 address size, saw it was 32 bits and then compared it to an IPv6 address, which was 128 bits, 4 times as many bits in that address. We looked at IPv6 address operation and saw that it works very similarly to the way that IPv4 addresses work. We talked about IPv6 address types including the unicast addresses, both global and link-local. We also looked at multicast and anycast addresses. We talked about IPv6 address acquisition. We did a demonstration where we manually configured an IPv6 address on a Windows workstation and then we took a look at both SLAAC, the stateless address auto configuration, as well as IPv6 DHCP. And we wrapped it up here by looking at IPv6 tunneling. Let's move onto the next module where we start to get into some network services like DHCP, DNS, and NTP. We're going to look a little bit deeper at those protocols next.

Network Service Protocols

Introduction

This next module, we're going to discuss network services. Now, network services are functions on the network that end users typically don't know about. However, these services are critical to making networks operate correctly. So our goals this module will be first to talk about network address translation, or NAT. Then we're going to talk about dynamic host configuration protocol, or DHCP. We have talked about DHCP already in the protocols and ports module; however, this time we're going to take a bit deeper dive into it. Next, we're going to talk about domain name system, or DNS. And again, we have discussed this briefly, but in this module, we're going to take DNS to a deeper level and understand the terms involved in DNS and see how DNS actually works.

Network Address Translation (NAT)

Let's start with network address translation, or NAT. Now, NAT is a network layer function, and we needed it in order to make the internet grow in the way that we use it today. So let's learn about how this actually works. So, at the same time that network address translation was written as a protocol, we also wrote the private IP address specifications where we set aside these three or four ranges of IP addresses to use on our internal networks, and these IP addresses cannot be routed on the public internet. As a matter of fact, if we send a message into the public internet with a destination address of any of these addresses that I've listed here, the internet will just throw it away. All the routers on the internet are programmed to discard messages with a destination address of one of these ranges. So how does this work? So here I have a small network in our internal devices. I've used the private IP range of 10.0.0.0 that I've subnetted into a /24. And then on the outside here I have an address of 203.0.113.6. This is a public IP address. So let's imagine that we want to go on our workstation and browse to pluralsight.com. So if I were to generate a message, I would put a source IP address of 10.0.0.10 and a destination address of pluralsight.com at 52.24.195.195. I can then send that message out on the internet to Pluralsight's website, and it gets there just fine because my destination IP address is a public IP address, which is entirely routable on the public internet. However, when Pluralsight responds to my message, it's going to reverse the source and destination IP addresses, and this time the source IP address is going to be pluralsight.com and the destination is going to be my workstation at 10.0.0.10. So when that message gets sent into the internet, the internet router is going to look at the destination and say, hey, 10.0.0.10, that's a private IP address, I can't route that. That can be literally anywhere on the internet, and it throws the message away now, and we can't get the message back to the destination. So instead what we do is we still create the same message where we have 10.0.0.10 as our source address, destination address is Pluralsight's website at 52.24.195.195. We send that message to the router, and when we get to the router, we have the router configured to do our network address translation. And what happens is the router will remove the source IP address from the packet of 10.0.0.10 and store it in a table. And when we store it in the table, we put some other parameters in it to make sure we know exactly which packet this is for, and then we replace the source IP address with the IP address of the router at 203.0.113.6, which is a public routable IP address. We then forward that message on to the internet, Pluralsight's website receives it, gathers up the website, packages it up, and then it's going to send it back to us. So now it has a destination IP address of 203.0.113.6, we send that into the internet, the internet knows how to get to our router because it's in the routing tables of the internet. Once our router receives that message, it looks up in the little table that it's set up and says, oh, this message, this is destined for 10.0.0.10. It then replaces the outside public IP address with our inside IP address of 10.0.0.10 and forwards the message on. So this is how network address translation works. So, network address translation is swapping out addresses in our packets in order to make those messages routable on the public internet. There's lots of uses for NAT beyond this, but this is the primary use of it. And in our home networks, we're actually using something called port address translation. As we are in this setup here, I didn't get into all the details of that. If you want to see more of the details of network address translation, there's another course you can watch. It's called Network Address Translation Operation and Configuration. It's a course for Cisco networks; however, I do go into extreme detail of how port address translation works and how that table is set up to work. So if you're curious about that, check out that course in addition to this one. However, just know for this particular scope of content, network address translation is replacing IP addresses in our packets to make them routable on the public internet.

Dynamic Host Configuration Protocol (DHCP)

Next, let's talk about dynamic host configuration protocol, or DHCP. Now, DHCP, this is an application layer protocol. The way DHCP works here is typically we need a DHCP server, a device on our network that can hand out addresses to the clients as they come online. There are lots of ways to do this. In large enterprise organizations, they will have a dedicated DHCP server that hands out the addresses for the entire organization, regardless if that organization has 100 devices or 50,000 devices. In your home network, which might look a little bit more like this one, typically that DHCP server is built right into the router. So in your home network, the DHCP server is built right into your device. Let's talk about how this works. So when we configure our device, and we have taken a look at this in a previous module in a demonstration, we can configure our IP properties here and say obtain an IP address automatically, and that's going to use DHCP to get that address. And our DHCP server then is set up with a scope. Now, the scope is basically the parameters of DHCP. So here we're saying the scope is going to be for network 10.0.0.0/24, which is the subnet that I have connected to the router. We have a range of excluded addresses. This isn't mandatory, but oftentimes we put excluded addresses in our DHCP scope so that the server does not hand out these addresses, and we can use them either to statically assign to devices or to set up a reservation that's specific for a device. Usually these excluded addresses are reserved for static configuration, for things like servers or printers or other strange hardware that may not support DHCP very well. We need to include our gateway, or default gateway, or default router. All those are the same thing. We need to include that in our scope so that our devices know where the default gateway is so that when we're trying to send traffic off of our subnet, our workstations know how to route that traffic appropriately. We need to configure a DNS server. Here I've used Google's at 8.8.8.8. And then we have a lease time, which is the amount of time that our workstations are going to hold on to that IP address before asking for a new one. So, if we want to get an address, our workstation's going to send out a message called a discover message, and that discover message is going to be received by the DHCP server. The DHCP server is going to make an offer to the workstation. So here it's making an offer of 10.0.0.100/24. It's going to say what the default gateway is and what the DNS server is, as well as the lease time. So that then becomes the IP address of the workstation. The workstation will then respond back to the server saying, hey, I request this address that you've offered me, and the server will send back a message that says I acknowledge that. So we have a discover message, an offer, a request, and an acknowledgement. So now our workstation has this IP address, and inside our DHCP server, we're going to make something called a DHCP binding, and that binding is just going to be a table that lists out the address that we've handed out, along with the MAC address of the device that we are setting up. Here I've just listed the MAC address as A. That Mac address is the hardware, it's the layer 2 address of the network interface card on that device. So here we have MAC address A binded to 10.0.0.100/24. Now, if we have a printer on our network, this is where the excluded addresses come in. We may need to have a static address assigned to that. So, we can do that in a couple of ways. We might pop that into our DHCP binding database, or we may just statically assign that address to that printer. The reason we need printers to have a static IP address is all the devices on our network always need to know how to reach that printer, so we typically statically assign that address to our device. And here is our lease time of 7 days. That was the 10,080 hours of our lease time. That's equivalent to 7 days. So our workstation will be able to hold on to that address for 7 days before it needs to request a new one. In larger networks, we may have more than one subnet that we're working with, and in that case, our DHCP server can be someplace on the network. Doesn't have to be a local device anymore. And what we would use here is something called an IP helper address. And this IP helper address, what we're going to do is when we send out that DHCP discover message, when that message hits the router's interface, the router will say, oh, I know where to send this, the DHCP server is over here at 172.16.1.68, and it forwards the message across the network to a DHCP server that is someplace on the network that's not local to the subnet. This way we can have a single DHCP server for all of the subnets in our network.

Domain Name System (DNS)

Next, let's talk about DNS, or Domain Name System, or Domain Name Service. This is another application layer protocol operating on transport layer 53, can use either UDP or TCP, depending upon what we are trying to accomplish. Typically, if we are just resolving a hostname into an IP address, we're using UDP, and I'll show you when we use TCP later on in this section. So, to start off with DNS, we first need to understand what a URL is, or Uniform Resource Locator. You may have heard of URL before, because this is what you would be given to visit a certain website. So if we wanted to go to pluralsight.com, this is the URL, and it's broken into several components here, so let's take a look at what those are. First, we have the top-level domain, or TLD. The top-level domain is the last part of our URL, in this case, pluralsight.com, the .com is the top-level domain. There are many top-level domains, including .com, .edu, .org, .net, .gov, .mil. There are country codes like .ca for Canada, .jp for Japan, .uk for the United Kingdom, .in for India, .au for Australia. And in the last 10 years or so, the governing body for these URLs introduced all kinds of new top-level domain names that you can purchase for your own organization. So just remember that that last part of our URL is the top-level domain. Now each one of these top-level domains has a special service on the internet called a root DNS server that stores all the other URLs and tells them where the authority is for those particular domains. We're going to talk more about that in a little bit, but just know that each one of these has a special service running on a server on the internet that identifies where all the domains are and where we can find information about them for DNS. The second part here is the second-level domain, that is in purple in my pluralsight.com address here, and the second-level domain is our domain, or typically what we call our domain. So pluralsight.com, that's our second-level domain and top-level domain. So then our second-level domain here is as much like google.com, cisco.com, wikipedia.org, he.net, facebook.com, and so on. So that second-level domain is identifying the unique organization itself, or the unique URL for an organization's thing, whatever they're doing with this, right? Maybe they're setting up a special website, maybe it's for some services, like for an FTP service or whatnot. So that second-level domain is typically identifying the organization itself, but it does not necessarily have to identify the organization, it's just a name that identifies a specific domain. Next, we have a third-level domain, and in the case of pluralsight.com, the www is our third-level domain, and this is also the hostname of the server that we're trying to reach. So, www is a server at the domain Pluralsight at the top-level domain .com. So www is the third-level domain or the hostname, Pluralsight is our domain, and .com is our top level domain. Now that third-level domain doesn't always have to be a hostname. In this case, we have a top-level domain of .edu, a domain of university, a third-level domain of engineering, which is a specific organization within the university. There might also be philosophy, and English, and literature, and physics, and chemistry, and what not. So every single department within a university may have its own third-level domain name where they can add then a fourth-level domain, which will be the hostname here, in this case, www is specific to a web server. When we are using DNS, the idea here is that we need to have a DNS server configured on our workstation. Typically this is done through DHCP. If we take a look at our IPv4 properties, we can see that we can either manually configure a primary and secondary DNS server or we can ask our DHCP server for that information so that it's automatically configured. In this case, we have it automatically configured, our DHCP server is set to 8.8.8.8, which is Google's DNS server out on the public internet. So what I would do if I wanted to reach pluralsight.com is I would type that into my web browser, but before my workstation could send a message to pluralsight.com, it's going to have to do a DNS lookup, and what that means is that I need to send a message from my workstation to the DNS server that's configured on my workstation and say, hey, DNS server, what is the IP address of www.pluralsight.com, and then the DNS server will reply and say, hey, pluralsight.com is at

52.88.79.159. Then my workstation can create the packet and send the message to Pluralsight to get the website. We could also do a reverse DNS lookup. The reverse DNS lookup is going to say, hey, what is the domain at 52.88.79.159, and then the Google DNS server, if that record is configured, they're not always configured, but if that record is configured, we can reply back and say, hey, that's pluralsight.com. So our DNS can work in two directions. The primary direction, we're doing a forward lookup, we're saying, hey, what's the IP address of Pluralsight, that's what makes the internet work; however, if we want to do some other nifty utilities, we may want to take an IP address and find out what the URL or the domain associated with that is, the reverse lookups don't always work, because the record is not mandatory to have in DNS. So look at different DNS record types here. So first we have an A record. So a DNS A record is an IPv4 record for a forward lookup. It maps a URL to an IP address. An AAAA record is an IPv6 record, right? And if you remember right, IPv4 addresses were 32-bits long, IPv6 addresses are 128-bits long or four times the size of an IPv4 address, so, we just made the IPv6 record called an AAAA record, four As, four times as large. Next, we have a CNAME record, or a Canonical Name record, and this is a record where we can set it up in DNS and there's no IP addresses associated with it, but what we can do is we can say if we go to this certain URL, redirect it to this other URL, and sometimes you see this happening, for example, with Google. If you accidentally mistype Google and don't put two Os in Google and instead type G-O-G-L-E and hit Enter, DNS will automatically redirect that to google.com with the correct spelling. And what happened here is Google, what they did was they purchased the gogle domain name and just put a Canonical Name record in DNS so that any time you try to go to gogle.com, it automatically redirects you to google.com with the correct spelling. So that's a Canonical Name record. An MX record is a Mail Exchange record, this is what we're going to use when we're using email and our email client is going to do a DNS lookup for an MX record, so we know where to send our messages for certain domains. Next is an NS record, or a Name Server record, and this identifies an authoritative name server. So on our root DNS servers, which are going to host our top-level domains, the .coms, the .edus, and what not, that service on those servers is going to host all these Name Server records, or NS records, and when we do a query, when we don't know what the IP address of a device is, we may have to go all the way up to the root DNS server, which we can say, hey, do you know how to get to pluralsight.com? And that root DNS server will say, yes, I do, I don't have the address, but I can tell you where the authoritative DNS server is for Pluralsight, and it can redirect me to the right place so I can find out the records that I need. I have some illustrations of this that we're going to look at in just a little bit. We have a PTR record, or a Pointer record. The Pointer record is for the reverse lookup. So remember earlier I said that we can do a reverse lookup where we put in an IP address and it tells us the domain associated with that? If we want that to work, we need to set up a PTR record or a Pointer record. There's also a Service record, the SRV or Service record, that is going to specify the IP address of that domain, as well as a specific port number to use if it requires it. So the SRV record is specific for an IP address plus a port number. We also have a TXT record, and this is able to provide additional information about the IP address and URL association, if we need to send some specific information back to our clients about what this is, we can put this in human readable format. So we can say, hey, this is reserved for documentation, or this is reserved for this certain other thing. So we can send information through DNS using this TXT record if we choose to. Let's take a look at how this works a little bit better by talking about internal versus external DNS, and we can talk about an example here of what happens when we can't find a record in our local DNS server. So, in most organizations, we do not use this system that I have shown here where we're pointing to an external DNS server to resolve our hostnames or URLs into IP addresses. Instead what we have is an internal DNS server, and the internal DNS server serves many purposes, one, remember we're using private addresses in our internal network, and oftentimes we're going to have services and websites internally for our organization that require us to look up to an internal IP address or a private IP address, and in that case we are going to need to have an internal DNS server to do those lookups. So what we would do here is we would configure our clients to look at the internal DNS server to resolve internal hosts. So, let's take an example, though, of if we're trying to reach pluralsight.com. So if I was trying to reach pluralsight.com from my workstation, I would first send the request to my internal DNS server, because that's what would be configured on my workstation. If for some reason my internal DNS server does not know how to reach pluralsight.com, what it will do then is it will be configured to search out to a public DNS server to get that information. So it would then send that request up to an external DNS server, like Google. If for some reason Google does not know how to reach pluralsight.com, what Google's DNS server would do is it's going to reach out to the root DNS server for the top-level domain of .com. The root server then would look up the authoritative name server and send the authoritative name server information to Google, so that Google's DNS server now knows where the authoritative name server for pluralsight.com is. Now this authoritative name server, this is a service that will be run by Pluralsight, or more likely, contracted out to a third party to run the authoritative name server for Pluralsight, and that service then would have all of Pluralsight's domain name and all the hosts or third-level domain name information in it, so that any time somebody on the internet wants to reach one of those devices, it can go to the authoritative name server to look that up. Now, authoritative name servers never are alone or are almost never alone, there's almost always a secondary authoritative name server. And, the way that we get that to work is we'll configure the authoritative name server with all of our URLs and IP addresses, we'll do a zone transfer, and this is going to be using DNS port 53, but using TCP as our transport layer protocol as opposed to UDP, which we're using for all of our lookups. So we do the zone transfer to our secondary server, that way if one of these servers goes down, we always have an authoritative name server to use. So after the root DNS server tells Google DNS, hey, the authoritative name server for Pluralsight is over here, now Google's DNS server can query the authoritative name server for Pluralsight, it can return the proper IP address to get to www.pluralsight.com. Google's DNS can then report that back to our internal DNS server, and in that process, both the Google DNS server are going to cache that entry in a table and it's going to save that, so the next time somebody needs to look up pluralsight.com, we can just look it up directly on our DNS server versus having to go to the root server, then the authoritative name server, and then back to the client that's asking for it. So here, the Google DNS server is going to cache it, our internal DNS server is going to cache that entry for pluralsight.com in its table, and it's going to then send that information all the way back to my workstation to tell me where pluralsight.com is. Now, it's only going to keep that entry in the table for as long as the TTL will allow it, and the TTL is the time-to-live. Now the TTL is configured by the authoritative name server in something called a Start of Authority record, or SOA. So the Start of Authority record is something that the authoritative name server hosts, and it has configuration information including the TTL for that particular entry. Now the important part of using that TTL is that if we need to update the IP address of a server in our authoritative DNS server, we can force all of the other DNS servers to refresh their entries by setting that TTL to be very low, so that we automatically have to go all the way back up to the root server, then down to the authoritative name server in order to get the information.

Summary

So, this wraps up us talking about some network services. We talked about network address translation and saw how it replaces IP addresses in our packet from private IP to public IP so that we can surf on the public internet. We talked about DHCP and saw how we can assign IP addresses to devices using that DHCP scope on a server. We also looked at domain name system and identified lots of terms in DNS in order to understand how that operates to allow us to have records that identify a host name, which is a readable name like www.pluralsight.com in a URL and translate that into an IP address, which we can actually use in our packets to request information from those sites. I hope you learned something about network services in this module. The next place we're going to go to is talking about network topology, where I'm going to describe different types of network topologies, as well as introduce virtual networking and other terms.

Network Topologies and Types

Introduction

In this next module, I'd like to talk about network topologies and types. So we're first going to start off by talking about some different network topologies. Unfortunately, these are a bit antiquated terms, but that's okay, they are good to have in your knowledge bank. We're then going to talk about different network types that we can have that you may hear for terms, we'll talk about some different WAN technologies and some of the terminology that are important for that, and we'll wrap it up with some virtual networking concepts.

Network Topologies

Let's get started here with network topologies. Now, I've used these three kind of corny icons here, because these are kind of old terms and they're a little bit corny. These network topologies, we have a bus, a ring, and a star topology. There's also a hybrid topology, which would make use of more than one of these. The reality is is we only use one of these today, most of these are antiquated, and I'm about to show you that. However, the Net+ certification exam does have these network topologies as one of the exam objectives, which is why I've included it here. If it were a course that were just on networking, I would not include these terms. So a bus topology, what it is is we have these PCs all connected to the same wire, so we use some type of coax cable and we connect all of our devices to one single wire, much like you would do with cable TV, and that one wire is then used to transmit data. This is an old technology. I haven't used this technology since the late '90s, probably 97/98. The topologies here that we used were called 10Base5, which is thicknet, 10Base2, which is thinnet; and in the photos I've included here, you can see how old this technology is. It's quite massive. The picture on the left there shows a transceiver for the 10Base5 network, and it's quite large equipment. So, we don't use this anymore. Ring topology, similarly, we don't use anymore. What we did here was we connected a bunch of devices to some type of ring, whether it be using coaxial cable or twisted pair cabling, or even fiber optics had a ring option. Again, this technology is antiquated, you can see that this IBM Token Ring MAU unit, which is used for a ring topology, it's old, we don't use this anymore. Topology that we use most often is the star topology, that's where every device has a cable running to a central location where we usually plug it into a switch, and this is what a switch looks like. This is a modern switch you might find in a small office. So with the star topology, this is the topology you're pretty much going to see in every single home network. In fact, it is in your home network. You're going to see this in corporate networks, whether it be a small business, medium sized, large sized business, or a data center even for that matter, they all use star topologies. When we're talking about communication on data networks, we can have a peer-to-peer network, which is where we'd have two devices sitting on the network which would have direct communication with each other. In these cases, usually both of them are both the server for data and the client for data, it's a peer-to-peer network. The other option here is a client/server network. This is what we talk about most of the time. In fact, in all of the previous modules, I have always talked about client/server networks, where we have something like a web server and then a web client. The web client is our web browser. The web server is going to be some software application running on that server that when the clients ask for a website they hand it to it. So client server is typically what we're going to see as the type of communication we have in modern networks.

Blank Area Networks

Let's talk about different network types here. Now I've categorized these as blank area networks. And there are numerous types of networks here and we categorize these so that we can discuss where events are happening on a network. Not all these terms are used, some of them are used more than others. So we'll start with three very popular terms here, actually four very popular terms. We have LAN, WAN, and SAN. These are the terms you're going to hear most often, spoken by engineers or other professionals in IT, and in the LAN category there's a subset called WLAN, which is wireless LAN. So let's look at each one of these here. So local area network is a LAN, and this is going to be numerous devices all connected together to a switch. This is a local area network. Typically, it's a Layer 2 connection where all the devices on the network are coming into a central switch. We call that our LAN. There might be multiple iterations of this. Maybe there's five floors in your office building and each floor has its own LAN, and we can refer to that entire infrastructure as the LAN infrastructure. It's the local infrastructure inside of our building where all of our devices are communicating. The wireless local area network is the same thing, except we don't have wires, all right? So it's basically the same exact concept of having these numerous devices all connected to a central switch. In this case, our central switch is our wireless infrastructure, we just don't have wires going to it. Another type of connection that's important that we understand here is a wide area network, or WAN. And this is where we take two LANs and we connect them together through some technology over a large distance. I'm going to talk about the different technologies we use to connect WANs together in just a moment, but understand that we're taking two WANs and we're connecting them together, and typically, when we do this, we need some router infrastructure in order to allow that to happen. So these circles with the arrows that I've drawn in, these are Layer 3 devices called routers. We're going to learn more about routers, switches, and their operation in the next course in this series. The next one I want to talk about here is a storage area network, or a SAN. What a SAN is is a SAN is a place where we can store information on hard drives or solid-state drives, and that's the icon I have drawn there, and we take lots of these drives and we put them all into a server or a series of servers, and then we connect it to the data network with some type of high-speed network connection. And what this does is it allows us to store data from our workstations in our office, or we can use it to store information from virtual machines in a data center, or all kinds of other information, maybe it's application information for a web server. As a matter of fact, Google, Apple, Amazon, Microsoft, they all have massive data centers of just storage area networks so they can store all that data that we as users have to put up there like our pictures, emails, things like that. It might be used for streaming services for movies to get streamed to your local devices in your home or a whole host of other things. So storage area networks are a way to have a storage location on the data network and we can connect that with protocols like FTP and SMB, among other things. Two other network classification types that you might hear occasionally, but not incredibly often, as a matter of fact, I rarely use these terms. If you are on a large university campus or a large corporate campus, you might hear a CAN or a MAN, the campus area network or metropolitan area network. These are effectively a type of WAN, just depends typically on who owns the infrastructure in that wide area network. So if you're talking about a CAN or a MAN, oftentimes we are talking about a wide area network infrastructure where we have lots of different facilities connected together, but instead of purchasing a WAN connection between two buildings through a third party, instead the organization actually owns the infrastructure to connect those together. So maybe they laid fiber optics on the campus of a university, maybe they've laid fiber optics within the city. A CAN and a MAN are effectively wide area networks, and it's just a matter of who owns that infrastructure. A PAN, I've never heard anybody talk about a PAN, a personal area network. This is one of the objectives on the NET+ exam. Personal area network is effectively the network that is used to connect the devices around you. So maybe have a smartphone, you might have some Bluetooth earbuds, or Bluetooth headphones, you might have a smartwatch that connects to your smartphone, you might even have some glasses. I believe Google, years ago had some glasses that connected to your phone. Snapchat has some glasses and there's always talk about glasses coming out, so it's, the personal area network is really just these devices that exist around you and are all connected together through some wireless infrastructure typically.

WAN Technologies

Let's talk about some of the technologies we use now to create wide area networks. So wide area networks, we have a couple options. One of them we call a leased line, and that's usually a copper connection called a T1 connection. T1s are some of the first connections that we had in data networking that allowed facilities to connect themselves together at great distances. The T1 connection was initially developed to convert analog phone signals into digital phone signals so we could transport them extremely long distances without losing a lot of quality. Over time, the way we transport voice technology moved to fiber optics and we repurposed the T1s to use for mainly data networking connections. And then over time again they have eventually reverted back to using T1s for voice connections in order to get our voiceover IP networks to work. A lot of that has changed over the years. There's lots of different technologies and ways to use this. I'll tell you right now, we don't use T1s as much as we once did; however, they are still in data networks. Another option here is fiber optics. These are much more popular wide area network technologies, especially in modern networks. Fiber optics can have the ability to carry traffic up to 40 Gbps or more, so they're very high-speed, high-bandwidth network connections that we use in our WAN connections. We have a couple options with fiber optics. One is dark fiber where you lease some fiber optics or you install fiber optics in the ground, and that fiber is dark, and you, as the customer, have to add on your own lasers onto that dark fiber. Another option here is something called metro Ethernet where you'll use fiber optics for the communication, however, your service provider will actually have the lasers on the fiber already and they will provide some infrastructure for you to connect to their network. Metro Ethernet is nice because you can control the bandwidth and it's usually at a much more reasonable price than using dark fiber or installing your own dark fiber. Another option is just a plain old internet connection where we put a VPN on top of it. So you might have a DSL connection, DSL is a digital subscriber line. This is a pretty ancient technology; however, we're still using it to some degree. We also have fiber optics connections that we can get for the internet. I even have fiber optic in my home right now for my internet connection. And you can easily get fiber optic in a business setting for internet connections. We can use these internet connections to connect back to some central location. Another internet connection type here is satellite. Elon Musk has a new satellite internet service provider that he is launching, actually, in 2021 here. That satellite connection is supposed to be very fast, low latency, it's supposed to be a very nice network. There are currently satellite internet providers as well. One of my relatives lives in a far remote portion of the United States where there is no real internet service, so he has to use a satellite to connect. It's very slow, it's not super reliable; however, it does work. So, in remote settings we might have satellite internet as well. Another option here is a cable modem or cable internet, and this goes over the same infrastructure that you would use to get cable TV. As a matter of fact, since streaming services have become so popular, especially in the United States, a lot of cable providers are just using their cable infrastructure now to provide a data network connection. They're actually streaming the cable service over the data network instead of providing a separate signal over that cable. So, that cable TV infrastructure that we once used to get TV signals into our house changed, and now we typically use that just for data networking connections and an internet connection. So these are the different types that we might have to provide our WAN connection. We may even use a wireless point-to-point network for our WAN connection. And with these there are some terms that are important. So if we have a T1 link, or an E1 link if we're in Europe, these came out of Bell Labs, they can operate up to 1.544 Mbps. So this is a pretty slow connection; however, it works nicely if we have a voiceover IP system in our network. There's a faster connection here, a T3. These are numerous T1s all bundled together, and that can go up to 40, almost 45 Mbps. We also have ISDN. ISDN is an old-school connection, and this we can have a Primary Rate Interface, or PRI, and this is what we would use as a SIP trunk alternative. Now what is a SIP trunk? Well a SIP trunk is what we would use for connecting a voiceover IP system into what we call the POTS network, or the plain old telephone service. So it's the interface between a voiceover IP network in the inside of our building and connecting it to the rest of the world. We typically have something called a SIP trunk, which could be a data network connection. The other option here is to use ISDN with a PRI, or Primary Rate Interface. Now you may never encounter any of these connections in your career. These are becoming slightly older technologies now, so you may not encounter these in your career at all. And if you do, it's actually a pretty simple connection. It works just the same as any other connection for the most part as far as configuration and support. If we do have a T1 line here, a leased line, there is something that's important here that we call a demarcation point. Usually that demarcation point in modern networks is using something called a smartjack, and that smartjack is just a real simple interface for us to plug into and connect to our internal infrastructure, maybe a CSU/DSU or we may plug that right into a router. The CSU/DSU is a cable service unit/data service unit, and it's an interface between a T1's version of communication and the communication that our router is going to be able to talk on. That smartjack is just a way for us to connect into that network, it's a physical connection, and the demarcation point is the point in the network that divides which is the telephone company's equipment and which is the customer's equipment. So that demarcation point is important. It can be used in things beyond T1s where we have a demarcation point, even in fiber optics we may have a demarcation point that identifies which equipment is the fiber's provider's equipment and which is the customer's equipment. So when we talk about optical WAN now, we're talking about fiber optics. So this is when we have buildings that are far away from each other or far enough away from each other where copper connections just don't work. T1 likely won't work because if we have our main office as a data center, we're going to have a lot of data happening there. If these are a fair distance apart, we're going to need a high-speed network connection between these. So that's where we'd have fiber optics running to these buildings and possibly use a service like metro Ethernet. We could also use dark fiber to connect these. Dark fiber is more expensive. With dark fiber, we get to control the bandwidth and everything about that link. We control the lasers, we control all the configuration of it. However, if we use something like metro Ethernet, the metro Ethernet option is a little bit nicer sometimes because the cost is less and we can pay money for more bandwidth, and if we don't need a lot of bandwidth, we can pay less money. So usually using metro Ethernet is a more cost effective solution than it is to use dark fiber directly. And the reason for that is the metro Ethernet service provider can share that infrastructure, that fiber infrastructure, with other organizations so we'll have multiple organizations using the same infrastructure to move data across connections, and we can pay more money to get more bandwidth or less money to get less bandwidth. This lets us, as the customer, control our costs and our data network needs. There are many options for metro Ethernet. You can get 10 Mb, 100 Mb, gig, 10 Gb, 40 Gb, and oftentimes anything in between here as well. When we use metro Ethernet, it's very likely that we're going to have multiple facilities all connecting through this network service provider that's allowing us to have metro Ethernet. In addition to that, that network service provider giving us the metro Ethernet connection, they're likely also providing network connections for many other organizations as well. Now in order to provide a very fast, secure, and efficient network, one that allows my organization, as well as other organizations using the same network service provider to have a high-quality, low-cost experience, we're typically going to use another protocol here called Multiprotocol Label Switching, or MPLS. This is something we'll use inside of the service provider's network. That's a protocol that will allow us to keep each organization's traffic separate, as well as providing features like allowing my organization to have numerous VLANs on that service provider's network. Now we have not talked about VLANs yet, but VLANs are a way of separating network traffic to be organized into efficient and secure networks. So with MPLS, my organization can have a very effective and efficient path through a service provider, and the service provider can allow other customers to use that same network, keeping each organization's traffic separate and secure, which will keep the costs lower for all the customers. So we have with MPLS a fast, secure, efficient, and a low-cost networking option for wide area networks. Now, anytime we are working with many facilities connecting together through a wide area network, we might have internet connections like a cable modem, we may have T1 connections, there may be wireless point-to-point WAN connections, or it could be metro Ethernet, dark fiber or some combination of all of them. They're all going to connect into our central infrastructure to allow each of these facilities to communicate with each other. So one of these might be a data center for like a bank, and then all the other facilities could be the branch offices for the bank that need to connect back into that central office. And we're going to use different technologies to connect into that central office based on what's available in the local market. So if it's a bank branch up in a rural location in the world, they may not have a dark fiber or metro Ethernet option. Your only option may be a cable modem or a wireless point-to-point network or a T1. So, in order to make these connections work more effectively, we put another technology on top of all of our WAN connections and we use a centralized controller to help manage it. And it's something called SD-WAN, or software-defined WAN. And all of these lines that I have drawn on top of my drawing here, these colored lines going from building to building, these lines represent tunnels that the software-defined WAN infrastructure implements in order to facilitate high-speed, efficient communication that's resilient and robust and meets the needs of our organization. So SD-WAN is a software-defined networking where we use a server that then sets up a routing path and some tunnels that allow all the facilities in my organization to communicate in the most effective way possible. And these tunnels that we set up, these are virtual tunnels that connect one building to another building without worrying about the actual path through the network. These tunnels are called MGRE, or Multipoint GRE, or Multipoint Generic Routing Encapsulation. Really all a tunnel is is we take a packet, we put it inside of another packet, and send it across the network. It allows for secure and targeted communication, and it allows the lower network to re-converge itself as it's needed to become faster without disturbing the communication between our two facilities. So we're going to talk more about tunneling in the next course in this series when we talk about VPNs. For now, just understand that SD-WAN, the software-defined WAN infrastructure, it runs on top of a current WAN infrastructure of dark fiber, metro Ethernet, T1 cable, you name it, it runs on top of that in order to provide a way of organizing our network in an effective and efficient way.

Virtualized Networks

Let's talk next about virtualized networks. So network virtualization has happened in our data centers. Starting in 2007 to 2010, we really had this massive push to virtualize our data center. So what does that mean, exactly? So in a data center, we have a bunch of servers, and it used to be that we had these physical servers and each physical server had its own specific purpose. Now, a server is a pretty simple computing device in the sense that computing devices are nothing more than a processor, some memory, which hosts the transactions that we're processing, and then some storage, like a hard drive or an SSD, and that device is long-term storage for things like the operating system and data that we are serving up with the server. We also need a place to connect in a monitor, and a network interface card, and maybe a keyboard and mouse, but ultimately, those are all input/outputs, so we have a network cable coming into these so that we can serve up whatever it is that we're doing, like a website, Well, as our data networks grew, we had a specific server for each task, sometimes we could have multiple tasks on a single server. But when we did analysis of this in 2008 to 2009, especially, the organization that I work for, we saw that we kept adding more, and more, and more servers, and we needed more and more redundancy in our network, so we got to the point where we had about 700 to 1000 servers in our data center, and we were really only using about 10 or 15% of the processing and memory power available in those devices. So what we did instead was we made use of technologies that allowed us to virtualized the hardware. And what that means is we took a single very beefy server, which had lots of processor, lots of memory, and we hooked up a storage area network to it, a SAN, and that allowed us to put many virtualized servers on there. So a virtualized server takes a piece of the processor on the physical server, it takes a piece of the memory on the physical server, and it takes the network interface on the physical server, and it allows all these servers then to run simultaneously on one server, so we have multiple virtual servers all running on a single physical server. What this does is it allows us to be much more efficient with our processor utilization and memory utilization. It takes much less power, physical power, electricity to run these devices. It takes much less cooling power to run these devices. We need less backup power for these devices. We need less network infrastructure hardware to run these devices. So when we virtualize our servers and put multiple servers running on a single physical piece of hardware, what ends up happening is we have a much more efficient data center. So, in those data centers then, we still need ways to network these devices together. So when we have our network connection to our physical server, we are going to have a way of accommodating lots of bandwidth so that each server can have its own communication path. And each communication path that those servers have is going to be running on a special secure network called a VLAN. We're going to talk more about VLANs again in the next course that we have here. But for now, just understand VLANs are a way of connecting our virtual servers into our network infrastructure so that they have a secure and fast way to communicate. Inside of our virtual environment, we have network hardware as well, we have virtualized network hardware, and network hardware like a switch is nothing more than a processor, memory, and input/outputs. So it's really the same as a server. It just is a very specialized server that provides networking services. So in this case, we can virtualize a switch so that all of our devices run on the star topology to a central switch that's all running inside of this physical piece of hardware, so we have a virtualized switch inside of our physical hardware. Oftentimes, we're going to have redundant systems. In fact, anytime possible, we're going to want redundant systems where we have more than one physical server running these virtual machines. These virtual servers then, we can have redundancy built in so we have the same server maybe running on two different pieces of physical hardware. That way, if one of them starts on fire, we will still have a running server to work with. The switches that are inside of our network, those are our virtual network. In addition to that, we can have other virtualized network functions too, like a load balancer. So, in addition to having redundancy between our two pieces of physical hardware here, with our virtual servers having redundancy, where we have the same virtual server on two pieces of hardware, we can also run a load balancer, a virtual load balancer, and that virtual load balancer will have a virtual IP address, so the virtual IP address will be the IP address that we connect to. So if we want to browse to a website, we might browse to a website on a virtual IP address, like 10.0.0.10, and then the load balancer can automatically load balance between 2 virtual servers, which have their own unique IP addresses, maybe 10.10.0.20 and 10.10.0.30. So for me, as the end user, I browse to a single virtual IP Address, the load balancer then automatically routes the traffic between our different virtual servers. All of this is run with something called a hypervisor. The hypervisor runs both the data network, as well as the servers inside of our physical pieces of hardware. And this is all part of a function called network function virtualization, which includes our switches, there might be routing functions, there might be a load balancer, all this network hardware that we use in the physical world can be virtualized and configured via this hypervisor, so it's all controlled in a central box.

Summary

Let's wrap up what we've talked about here. We've talked about different network topologies that are mostly antiquated except for the star topology. We talked about network types, and these are the blank area networks, the LAN, the WLAN, the WAN, the SAN, the PAN, the MAN, the CAN; all these different types of area networks that we use to describe components of a large infrastructure. We talked about different ways of providing WAN technologies, focusing mainly on dark fiber and Metro Ethernet. We also talked about that SDWAN infrastructure, which allowed us to build a software-defined network on top of an existing WAN infrastructure. Last, we looked at some virtual network concepts and saw how we virtualized servers onto a single server, and then inside of there have virtualized network hardware as well. Now, all of that should hopefully make more sense as we move through these courses on networking and we start to talk more about network hardware and what its purpose is and how it works, and that's going to happen in the next course in this series. Let's move onto our next module, where we're going to talk more about data center network infrastructure and software-defined networks that we use inside of there. Much like that SDWAN that I mentioned here, we can do software-defined networking within our data center, and it has a little bit different concept than what we do for traditional networking, so let's go take a look.

Data Center Networks

Introduction

Welcome to this next module where we're going to talk about corporate and data center architectures. In this module, I'm going to introduce two new models for networking. This gets a little confusing in my mind, so I'm going to set the tone here in just a moment about how we're using each of these models of networking. So first, we're going to talk about the three-tier design model for networking that we use in corporate networks to create resilient networks. Next, we're going to talk about software-defined networks and the model that we use in SDN. And last, we're going to talk about storage area network connections, learning how we connect our SAN to our servers.

3-Tier Network Design Model

So first, let's start with this three-tier network model. I've already introduced the OSI model, so you might be asking the question, Ross, what's going on here? Don't we already have the seven-layer OSI model that we only use five layers of, and now you're telling me there's a three-tier model? What's up with this? Well, what's up with this is that the OSI model is specifically used to describe how protocols encapsulate network traffic so that traffic can use HTTP, and TCP, and IP, and Ethernet to move across a network from one device to another. The three-tier network model that I'm about to describe, this is a design model, this is what we use when we're building corporate networks in order to provide a resilient, redundant, and high-speed network where traffic can move.

So here in this design model, this network design model, we're talking about how we design and organize the hardware used to move traffic, versus the OSI model, which is talking about how traffic is encapsulated and the hierarchy of protocols that we use in order to get traffic to move from one device to another. So here we're really talking about design and not network traffic encapsulation. So this is a different model for a different purpose. So the three-tier design here, the three-tier network design model, this is typically used for Cisco gear; however, this model is applied pretty much universally because it's a really nice way to provide for redundancy, resiliency, and high-speed traffic movement in our network. Here's how it works. There's three layers, there's the core layer, there is the distribution layer, and there is the access layer. Let's start with the access layer here. In the access layer, so if we imagine we have two buildings here, a data center and an office building, inside of our office building we're going to have lots of workstations and clients. They might be wired, or wireless, or handhelds, or tablets, regardless there's going to be lots of devices in our office building which are going to require access to the servers in our data center, which are also going to be numerous and need to be connected to the network. So both locations require us to connect the devices to the network in some way. And this is the access layer. So we're often going to have lots, and lots, and lots of devices in these buildings. So in our access layer, we're running the actual cables from the device to a switch, or typically we're running it to a switch, this could also be connected via wireless connections, especially in our office, not so much wireless in our data center. So our access layer here is the layer of the network model we are connecting the devices to a switch. So the purpose here is to provide network access to the devices and also to control access to the network. So we may have some policies on these switches that when we plug a device into that switch, it verifies that that device is authorized to be on the network, so that we don't end up with a device that's owned by an attacker or some nefarious agent plugging into our network and trying to take control of the network. The next layer here, our distribution layer. The distribution layer is responsible for connecting all of the access layer devices together and starting to build this connection between our offices and our data center. So here we're going to be using some Layer 3 switches or some routers to take all of our access layer devices and connect them together usually to some redundant devices. Whether that be multiple Layer 3 routers or multiple Layer 3 switches, we're going to be building our network here and connecting all those devices to that so that we have high-speed and redundant access now to this distribution layer of our network. The distribution layer here is responsible for distributing network access to the access layer, and hopefully that's done in a very redundant way. Additionally, at the distribution layer, we're going to filter traffic. So if there is traffic that we find undesirable on our network, we're going to filter that out at the distribution layer so that we can keep the next layer nice and clean so traffic can move very fast. Additionally, here's where we're going to have routing policies. So here we're going to implement routing protocols to distribute our network so that we have high-speed paths that are redundant and resilient through our network. So the distribution layer here, this is responsible for getting network access to our devices in a redundant way, keeping unwanted traffic off of our network, and have routing policies that allow for moving traffic at high speeds, as well as some resiliency and redundancy so that if something breaks in our network, our end users don't even notice that the network broke. The last layer of our three-tier model is the core layer. Now, the core layer is oftentimes called the backbone of our network, and the backbone is going to connect all of the distribution layer nodes together. It's going to be a high-speed option here. There's going to be lots of high-speed connections. It's going to quickly move traffic from one part of the distribution layer to another. It's going to have few to no policies so that there's almost never changes at the core layer so that we end up limiting the changes we make so that traffic can move very fast over this core layer. It's kind of like thinking about how your neighborhood might be designed. You have these small roads in your neighborhood, and you have a driveway into your house or maybe an alley that leads to your house. It's kind of like the access layer. It's the path you drive to get from your house to some place. You drive out your driveway onto the small street, and then typically we move to a bigger street that may have some stop lights, may have two or three lanes on it, and then we moved to an even bigger road, like a freeway, which has lots of lanes, no stop lights, and high speeds. Right? So the three-tier network model is very much similar to how you might experience driving a car in a neighborhood, moving from the low-speed roads, to the higher-speed roads, to even higher-speed, higher-volume roads.

So this three-tier network model, this is how we typically like to design the hardware and the connections of the hardware in order to make a highly resilient, redundant network. So core distribution access, this is the three-tier design model we're using for our hardware design in networks.

Software Defined Networking Model

Now software-defined networks are a whole new beast in data networking. They have been slow to get adopted, but now they are here, and they're here to stay. We typically see software-defined networking used in data centers, but it does not have to be exclusive to data centers, and we're seeing it slowly move to accommodate the entire network. A lot of people ask me, they say, Ross, like with software-defined networking, does that make the rest of the OSI model or the three-tier design model or other forms of networking obsolete? Does it make IP and TCP obsolete? Absolutely not. What software-defined networking is going to do is it's going to make use of all of those existing technologies and just build on top of it. So let's take a look at how this works. In software-defined networking, or SDN, we have three layers here. Now the three layers that we're talking about here are specifically to describe how software-defined networking works. So let's see if I can make some sense of this for you. And don't forget, remember the OSI model is still relevant here because the OSI model is describing that client to server communication and how we package up data. The software-defined networking model here is going to describe the overlay we put on our infrastructure on our hardware to describe how SDN is working to move traffic nice and quick for us across our network. So let's take a look. We'll start with the infrastructure layer. So the infrastructure layer is going to consist of routers, switches, Layer 3 switches, that are all going to be laid out and connected together with certain policies, with routing policies, with switching policies. So, the infrastructure layer is going to be a layer of the network that already has networking applied to it. As a matter of fact, we can use the infrastructure layer just to pass traffic normally and talk from a client to a server on the network. However, what we end up doing here is we create something called the management plane. The management plane is a new mechanism of configuring, and controlling, and monitoring all the devices on our network. So what we can do as an administrator is we can use that device on the lower left-hand corner of the screen, that PC there connected to that orange switch, and we can use that device to then configure something called a software-defined network controller, or an SDN controller. That SDN controller, then is going to host all the configuration that will get applied as an overlay to the rest of our hardware to make it easier to control and manipulate traffic. The idea here is that the infrastructure layer is the hardware itself that we're going to be using to move traffic across our network. We're then going to use a workstation to actually go configure a controller, and that controller is going to push out additional configs to these devices to make them work more effectively. Now, that configuration that we apply is all applied at the control layer, and the control layer is responsible for actually moving this traffic around. It's responsible for accepting the configuration from the administrator from the workstation there. It's responsible for a multitude of things that all revolve around moving the traffic, giving devices the right information like an IP address via DHCP service, applying some access control rules to limit where traffic can go, and ultimately actually responsible for moving the traffic to the right location at the most efficient way possible. The control layer is going to adapt to the network as we get information from the higher layer, the application layer. So the application layer here in software-defined networking, this is going to be where administrators and developers can create utilities and rulesets that will allow the network to operate in a dynamic way. So if there's a lot of traffic coming from one device going to a different device, the application layer can have policies written for it that get pushed down to the control layer that say, hey, when this event happens, reconfigure the network in this way so that we can have a very efficient path for this additional traffic load. And then when that additional traffic load is done, reconfigure it again so that we can restore the original configuration. So the application layer is all about creating utilities that can be used to push down to the control layer to better manage how that traffic works. What software-defined networking is is it's using all of that existing infrastructure to put on a whole other layer on top of that so that we can better control how traffic moves through our network. As networks get larger and larger, software-defined networking is going to allow us to better control and manage all of that gear in our network. Now when we talk about software-defined networking, most of this is happening inside of a data center. So let's take away the client for a second here, at least minimize the client for a second, and talk about what's happening inside of a data center where software-defined networking is used most often. So first off, we're going to talk about these switches that we use here. So in software-defined networking, we're typically going to have a switch at the top of a rack. What on earth is this? Alright, so here's an image of a data center or what a data center might look like. This is a nice, super clean, very pretty, heavily digitized data center. I don't expect yours to look like this. However, the idea here is that all these devices in here are some type of server or storage area network. So they're going to be servers or servers hosting virtual servers, storage area networks, and other network appliances. And in order to connect these all together to the network via the access layer, they need some type of switch. So what we do is at the top of each one of these racks, we'll oftentimes place a network switch, and that network switch will then be the point where all the servers in the rack connect to the top of the rack via this switch. So that top-of-rack switch is often called a leaf, and that top-of-rack switch then connects to another switch called the spine switch, which is responsible for aggregating that. So we can kind of look at this as like the top of rack switch is like our access layer, and the spine switch is kind of like our distribution layer. However, this is going to be applied in software-defined networking instead. So the concept is the same here. We're just using slightly different terminology. When we're talking about how communication moves within a data center, we have both East/West and North/South communication. East/West communication inside of a data center is server-to-server communication. When we browse to a website, we might browse to a server that hosts the website. However, that server may not host all the images and other content that we're looking at, and it may have to go to another server to get the images to populate in our website. So in order to do that, our web server is going to say, hey, image server, serve up these images to the client. So that's East/West communication, these servers talking to other servers to tell them what to do. The North/South communication is going to be communication that comes from a client, goes into the data center and back out of the data center. So East/West is within our data center, North/South is in and out of our data center.

SAN Connections

Now the last component we're going to talk about here is connecting our storage area network to our servers. So our servers become these appliances on our network that are literally just processor and memory. It's just processor and memory. They don't know anything else but how to process things and how to store them temporarily in memory, in RAM. What they need is they need some storage, they need a hard drive or an SSD drive to connect to it. So, we do that now with a storage area network. So on our laptops, our laptops are basically similar to this, right? Our laptop has a processor, it has RAM, or memory, and then it has a hard drive, usually an SSD nowadays. And those three components work together, we store our operating system and our files on our hard drive, the processor processes the information, , it's the mathematics engine of our our laptop, and then the memory is a place to temporarily store operations as they run through the processor. Well, servers don't have hard drives typically anymore, or SSD drives, usually now we use a storage area network. So the servers need a way to connect to the SAN in order to get all the data, the operating system, the files, images, things like that. So the way we do this here is we run connections from the SAN to our servers, and typically we're doing that with some type of switch. So we connect our SAN to a switch and then we connect our servers to a switch. Well, there's three ways we typically do this. Method one is a method called Fibre Channel, and if you notice, Fibre looks like it's spelled strangely, there, F-i-b-r-e. It's not a spelling error. That's how we talk about the connection when we're talking about a SAN. Fibre Channel is actually using fiber optics to connect to a fiber optic switch to distribute our storage area network to our servers, and that's one method we can use. Fibre Channel over Ethernet uses the same protocol as Fibre Channel, however, instead of operating it exclusively over fiber optic connections, now we can run it over high-speed ethernet connections and we don't necessarily need to have a separate fiber infrastructure for our SAN. We can run our Ethernet connection to our servers on the same connection that we run our SAN connection. That's called FCOE, or Fibre Channel over Ethernet. Both Fibre Channel and Fibre Channel over Ethernet are considered to be layer 2 protocols, meaning we don't need to worry about IP or TCP or anything above the data link layer of the OSI model. ISCSI, on the other hand, is another method we can use to connect our SAN to our servers. ISCSI is going to use TCP/IP to make a TCP/IP connection between the SAN and the servers in order to pass that traffic. So, it's still a very high-speed connection, it just uses a slightly different technology to do it. So, when we're talking about SAN and connecting them to servers, you will likely hear about one of these three options in order to connect our devices together.

Summary

So, let's wrap up what I've talked about here. I've talked about two new design models here, or two new networking models. One was this 3 tier design model for networking where we talked about how we can have the core, distribution, and access layers to help us figure out how to properly design a redundant resilient network. We talked about the software-defined networks model, which talks about how we apply this overlay on our network so that we can have a more resilient, more easily configured network, especially as networks grow very large. And then last, we talked about how Storage Area Network connections work, talking about the three different types, Fibre Channel, FCoE, and iSCSI. Hopefully this brings some insight into networking technology, specifically how we design networks and how we talk about software-defined networks. My hope was that we didn't introduce more confusion by talking about more layers that don't end up conflicting with the OSI model, which is talking about something unique about how we're moving traffic from one device to another device, versus these models, which are talking about how we design and explain the operations of networks. Let's move on to the last module in this course, where we talk about cloud computing concepts and how we connect to them.

Cloud Concepts

Introduction

Well, it's been a long journey of a previous 12 modules of content before we got to this Cloud Services module. This is the last module in our Network Concepts and Protocols course. We're going to go deeper into all of these concepts we learned in this course in the next four courses while we cover the rest of the Network+ certification exam objectives. If you're not taking that Net+ certification exam, that's okay. All of these courses are designed to give you all the language, terminology, and understanding of how data networks work so you can be successful in any IT career that you are undertaking. Let's get started talking about cloud services here. Our goals for this module are first going to be to introduce cloud services. Then we're going to talk about the different types: public, private, hybrid, and community clouds. Then we'll talk about Software as a Service, which is a cloud-based function, as is Platform as a Service, Infrastructure as a Service, Desktop as a Service, and some of the benefits of using these cloud technologies.

Introduction to Cloud Computing

Now cloud services in general, this isn't a new concept that we're doing. What we're doing is we're taking all of the stuff that we used to do in our own private organization's data center that was right on-premise, and we're taking all the functions of that and we're moving it into some place on the internet. So those functions that we needed in our own internal data center, we needed servers that had processors and memory, we needed storage that showed up as disk drives or as storage area networks, and we needed operating systems to run those servers, as well as the services we were providing. So we might need a web server, or an email server, or some type of file server. All these services traditionally existed within the private company's own data center. But as the internet connections grew faster, as technology grew, as Amazon, Google, Apple, and Microsoft grew their cloud-based service's offerings, a lot of those services could be offloaded from the private organization up into a cloud. So what we did here was we took all the services that we needed in this private organization and we moved them to some other organization's data center. So really what it meant is that instead of supporting our own local data center, we paid a monthly service fee to use somebody elses. And somebody else's data center, they orchestrated it in a way that made it super efficient so that lots of organizations could share that same data center and use all the applications. And what happened was, is it became more efficient to use this and the cost became cheaper. Additionally, we found out that we could scale our business much faster by using somebody else's network because we didn't have to go buy more servers when we wanted to expand, we could just ask our cloud service provider to give us more servers, which they could typically do at the flip of a switch. Those cloud services that we ask for are going to typically be defined of one of three or four services here, Software as a Service, Platform as a Service, Infrastructure as a Service, and then sometimes Desktop as a Service. So when we talk about cloud services then, suddenly cloud became a name that we used for our own local data center, a private cloud, and the internet's data center, a public cloud. So we might have a data center and have some cloud services in there, and then we might purchase additional cloud services from someplace on the internet, like through Apple, or Google, or Amazon, or Microsoft, or maybe a local cloud service provider in your area. So private cloud is our internal services. Public cloud is what we purchase from others. Typically, most organizations are going to use some combination of this, which we call a hybrid cloud, where some of the services reside inside our organization, some of them reside inside of the cloud. So another type of cloud we can have here is one where our organization may have a private cloud offering some specialized services. In healthcare, oftentimes there are specialized doctors that focus on very unique specialties of medicine. One of them is determining if a patient had a stroke or not. That is a very specialized area of medicine, it requires a lot of training, and a lot of local organizations, especially in rural communities, may not have those services. So a healthcare organization may reach out to those small, rural community healthcare organizations and say, hey, we can offer you some services via telemedicine, where we set up a video camera and some other specialized equipment, where a doctor at a large healthcare facility that has all the training that's required, can remotely diagnose a patient so they can be better treated for medical conditions. In this case, we don't really need some public cloud. Amazon and Google aren't going to be able to offer this service.

However, this private healthcare organization will be. So what this is called is a community cloud. When we have a private organization offering a specialized service, and we're going use their cloud infrastructure, their data center, to allow other smaller organizations to connect to it and use those services. So that is a community cloud.

Cloud Services

Let's talk about our different cloud service options here. We have Software as a Service, Platform as a Service, Infrastructure as a Service, and Desktop as a Service. So let's start with Software as a Service. So Software as a Service here is popular in consumer markets. We use it for things like Microsoft Office or Google's office suite, Adobe has products that are available in the cloud. There are many, many, many other software products that we can use, oftentimes right through a web browser, and it's all based in the cloud. The software oftentimes isn't even installed locally on our device, we just use it through an internet connection. These have been used in commercial operations for a long time. Oftentimes we may have used Software as a Service to host our customer information databases, to host our email systems, things like that. Another one here is we can actually host things like DNS, Domain Name System, which isn't an application like Microsoft Office; however, it is a service that can be offered as a software that we can ask a cloud service provider to provide for us as a software. The next one here is Platform as a Service. Platform as a Service offers us hardware and software where we can connect to our hardware and software remotely. It's basically our server side services. So a cloud service provider might offer us a server and say, yeah, here is a Linux server running these specific services and it has these specific properties of processor and memory and storage space. Oftentimes databases are offered as Platform as a Service. We can have some type of SQL database server set up for us, and all we do is we send our data in and out of that database and we don't have to worry about supporting the database and the database hardware itself. That's all done as the platform. Basically, we are given processing power for Platform as a Service to run the applications that we need. And a lot of times, we can do this as colocation where we have services set up in somebody else's data center to offer us redundancy to our own systems. So Platform as a Service offers us something like a database platform or some hardware along with an operating system that we do minimal support of, and it allows us to make use of somebody else's resources instead of using our own internal resources. The next one here is Infrastructure as a Service. This would be where we just rent a piece of server hardware, and that server hardware becomes our own organization's server hardware that we use exclusively. We put our own operating system on it, we put our own applications on it. They just provide us the place to put it, and they provide the power, the redundancy, and all the other services that a data center has to offer. We typically will choose how much disk and storage we need, and we can also have some automation built into this. So if we suddenly grow our business and now we need two servers or five servers, we can ask that organization that we're buying this Infrastructure as a Service from to automatically expand our service needs as our organization grows. So Infrastructure as a Service is providing the server hardware itself and some disk and storage. This is different from Platform as a Service where that cloud service provider is actually supporting the operating system of that machine itself. The last one here, Desktop as a Service, this is a newer one and this is basically providing a virtual desktop to a user. So virtual desktops are a way of providing a working environment for a user that's highly secure and easy to manage. The workstation is mobile. Oftentimes you can use any computer that you want in order to log into your work's workstation, so you can do your work from anywhere really as long as you're using a Desktop as a Service Provider. This makes it very easy to deploy desktops to users. It's also very easy to support these desktops because if you have an issue with your desktop, you can call into a help desk and they should be able to remotely configure and support the workstation you're working on.

Cloud Benefits

The benefits to using these cloud services is one that's called multitenancy, Meaning that, when we use a cloud service, something like Amazon, Amazon has data centers all over the world. And it means that when we set up some services with service providers like Amazon, or Google, or Microsoft, we can have servers all over the world that are hosting the same application, they get all synchronized, and it means that we have multiple servers in multiple locations. This adds redundancy, it allows us to have a worldwide audience, and it allows us to grow our organization rapidly when we need to because our servers are in many, many data centers. Elasticity is another benefit here. And elasticity means that we can grow and shrink our needs of our IT infrastructure as we need to, and we can oftentimes do this right from our desktop, as an administrator, without even calling into a salesperson or any other technician. We can just grow our environment or shrink it as we need to. Another one is scalability, and this is related to elasticity. Scalability, meaning that as we get more customers, we can grow the number of services we need rapidly. The last one here is security. Security is always a concern with cloud services because we're usually handing our data over to some third-party provider. However, oftentimes those third-party providers have a much larger security team and much greater infrastructure to manage our security needs. So when we talk about cloud-based security, really what's happening here is we are offloading our own security risk onto another organization and letting them handle the security issues that we may encounter in our organization. So there's lots of benefits to using the cloud here, largely the benefits are that we get to have servers deployed around the world and we get to grow and shrink the quantity of servers we need at the drop of a hat.

Summary

So to wrap up this module in this course, what we've done here is we've introduced cloud services, talked about a public cloud, private cloud, a hybrid cloud, and community clouds. We've looked at Software as a Service, Platform as a Service, Infrastructure as a Service, Desktop as a Service, and then looked at the benefits of using these clouds. I hope this introduction to networking with Network Concepts and Protocols was beneficial for you. We have a lot more to cover about data networks. We have barely touched the surface on networking hardware, and we haven't even started the conversation about how traffic gets moved across our networking hardware. That's what the focus of our next course is going to be. I hope you got a lot of value out of this, and I hope you learned a tremendous number of terms that will benefit you in your career in your IT profession.