Lecture Notes

Networking Fundamentals [CIT03]

Centre of Excellence
in
Information Technology

Papua New Guinea

Sunday 19th July, 2020

Contents

Chapter 1

Introduction to Networking

Outcome

1. You'll be able to explain all five layers of network model.
2. You'll be able to identify and describe each layer and what purpose it serves.
3. You'll be able to identify and describe various networking cables and devices.
4. You'll be able to have deep understanding in Physical and Datalink Layer.

1.1 Introduction

Networking skills are critical. As need to understand not just how applications work on a single system, but how they interact with all other systems in the company and even externally. Computers communicate with each other a lot like how humans do. Take verbal communication as an example. Two people need to speak the same language and be able to hear each other to communicate effectively. If there are loud noises, one person might have to ask the other person to repeat themselves. If one person only somewhat understands an idea being explained to them, that person might ask for clarification. One person might address only one other person or they may be speaking to a group. And there's usually a greeting and a way to close the conversation. The point is that humans follow a series of rules when they communicate, and computers have to do the same.

Protocol
The defined set of standards that computers must follow in order to communicate properly is called a protocol .

Computer networking is the name we've given to the full scope of how computers communicate with each other. Networking involves ensuring that computers can hear each other, that they speak and use protocols that other computers can understand. They have to repeat if messages are not fully delivered, and a couple other things, just like how humans communicate. There are lots of models used to describe the different layers at play with computer networking. TCP-IP five-layer model, the other primary network model, the OSI Model, which has seven layers. It's super important to know these types of layered models to learn about computer networking because it's a really layered affair. The protocols at each layer carry the ones above them in order to get data from one place to the next. Think of the protocol used to get data from one end of a networking cable to the other. It's totally different from the protocol you use to get data from one side of the planet to the other. But both of these protocols are required to work at the same time in order for things like the internet and business networks to work the way they do. Sometimes, there are problems when computers on the internet or on these business networks try to communicate with each other and often, it's up to an IT support specialist to fix these problems. This is why understanding computer networking is so important.

1.2 The TCP/IP Five-Layer Network Model

#	*Layer Name*	*Protocol*	*Protocol Data Unit*	*Addressing*
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

To really understand networking, we need to understand all of the components involved. We're talking about everything from the cables that connect devices to each other to the protocols that these devices use to communicate. There are a bunch of models that help explain how network devices communicate, but in this course, we will focus on a five-layer model.

Let's start at the bottom of our stack, where we have what's known as the physical layer . The physical layer is a lot like what it sounds.

Physical Layer
It represents the physical devices that interconnect computers.

This includes the specifications for the networking cables and the connectors that join devices together along with specifications describing how signals are sent over these connections. The second layer in our model is known as the data link layer . Some sources will call this layer the network interface or the network access layer. At this layer, we introduce our first protocols. While the physical layer is all about cabling, connectors and sending signals, the data link layer is responsible for defining a common way of interpreting these signals, so network devices can communicate.

Datalink Layer
It is responsible for defining a common way of interpreting these signals, so network devices can communicate.

Lots of protocols exist at the data link layer, but the most common is known as Ethernet, although wireless technologies are becoming more and more popular. Beyond specifying physical layer attributes,

Ethernet
Ethernet standards also define a protocol responsible for getting data to nodes on the same network or link.

The third layer, the network layer is also sometimes called the Internet layer.

Network Layer
It allows different networks to communicate with each other through devices known as routers.

A collection of networks connected together through routers is an internetwork, the most famous of these being the Internet. While the data link layer is responsible for getting data across a single link, the network layer is responsible for getting data delivered across a collection of networks. Think of when a device on your home network connects with a server on the Internet. It's the network layer that helps gets the data between these two locations. The most common protocol used at this layer is known as IP or Internet Protocol.

Internet Protocol
IP is the heart of the Internet and most small networks around the world.

Network software is usually divided into client and server categories, with the client application initiating a request for data and the server software answering the request across the network. A single node may be running multiple client or server applications. So, you might run an email program and a web browser, both client applications, on your PC at the same time, and your email and web server might both run on the same server. Even so, emails end up in your email application and web pages end up in your web browser. That's because our next layer, the transport layer. While the network layer delivers data between two individual nodes, the transport layer sorts out which client and server programs are supposed to get that data.

Transport Layer
It sorts out which client and server programs are supposed to get that data.

The protocol most commonly used in the fourth layer, the transport layer , is known as TCP or Transmission Control Protocol. While often said together as the phrase TCP IP, to fully understand and troubleshoot networking issues, it's important to know that they're entirely different protocols serving different purposes. Other transfer protocols also use IP to get around, including a protocol known as UDP or User Datagram Protocol.

The big difference between the two is that TCP provides mechanisms to ensure that data is reliably delivered while UDP does not. the network layer, in our case IP, is responsible for getting data from one node to another. the transport layer, mostly TCP and UDP, is responsible for ensuring that data gets to the right applications running on those nodes. The fifth layer is known as the application layer . There are lots of different protocols at this layer, and

as you might have guessed from the name, they are application-specific. Protocols used to allow you to browse the web or send receive email are some common ones. The protocols at play in the application layer will be most familiar to you, since they are ones you probably interacted with directly before even if you didn't realize it.

You can think of layers like different aspects of a package being delivered. The physical layer is the delivery truck and the roads. The data link layer is how the delivery trucks get from one intersection to the next over and over. The network layer identifies which roads need to be taken to get from address A to address B. The transport layer ensures that delivery driver knows how to knock on your door to tell you your package has arrived. And the application layer is the contents of the package itself.

1.3 The Basic of Networking Devices

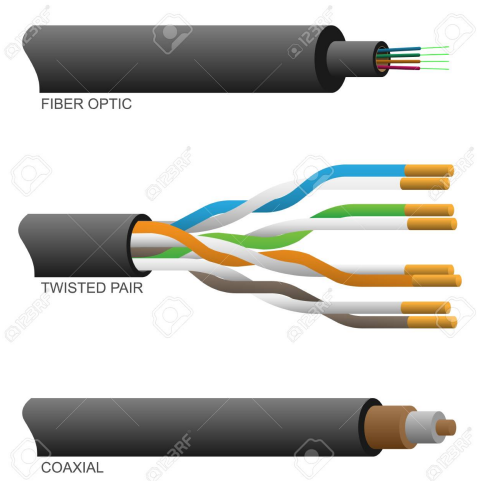
Lots of different cables and network devices can be used to allow computers to properly communicate with each other.

1.3.1 Cables

Let's start with the most basic component of a wired network: cables.

Cables
Connects different devices to each other allowing data to be transmitted over them.

Most network cables used today can be split into two categories, copper and fiber.



Copper cables are the most common form of networking cable. They're made up of multiple pairs of copper wires inside plastic insulator. You may already know that computers communicate in binary which people represent with

ones and zeros. The sending device communicates binary data across these copper wires by changing the voltage between two ranges. The system at the receiving end is able to interpret these voltage changes as binary ones and zeros which can then be translated into different forms of data.

Copper Twisted-Pair Cables
The most common forms of copper twisted pair cables used in networking are CAT5, CAT5E, and CAT6 cables.

These are all shorthand ways of saying category five or category six cables. These categories have different physical characteristics like the number of twists in the pair of copper wires that result in different usable lengths and transfer rates. CAT5 is older and has been mostly replaced by CAT5E and CAT6 cables. From the outside, they all look about the same, and even internally, they're very similar to the naked eye. The important thing to know is that differences in how the twisted pairs are arranged inside these cables can drastically alter how quickly data can be sent across them and how resistant these signals are to outside interference. CAT5E cables have mostly replaced those older CAT5 cables because their internals reduce crosstalk.

Crosstalk
Crosstalk is when an electrical pulse on one wire is accidentally detected on another wire. So, the receiving end isn't able to understand the data causing a network error .

Higher level protocols have methods for detecting missing data and asking for the data a second time. But of course, this takes up more time. The higher quality specifications of a CAT5E cable make it less likely that data needs to be re-transmitted. That means, on average, you can expect more data to be transferred in the same amount of time. CAT6 cables follow an even more strict specification to avoid crosstalk, making those cables more expensive. CAT6 cables can transfer data faster and more reliably than CAT5E cables can, but because of their internal arrangement, they have a shorter maximum distance when used at higher speeds.

The second primary form of networking cable is known as Fiber, short for fiber optic cables.

Fiber cables
Fiber cables contain individual optical fibers which are tiny tubes made out of glass about the width of a human hair.

These tubes of glass can transport beams of light. Unlike copper, which uses electrical voltages, fiber cables use pulses of light to represent the ones and zeros of the underlying data. Fiber is even sometimes used specifically in environments where there's a lot of electromagnetic interference from outside sources because this can impact data being sent across copper wires. Fiber cables can generally transport data quicker than copper cables can, but they're much more expensive and fragile. Fiber can also transport data over much longer distances than copper can without suffering potential data loss.

1.3.2 Hubs and Switches

Hub Switch

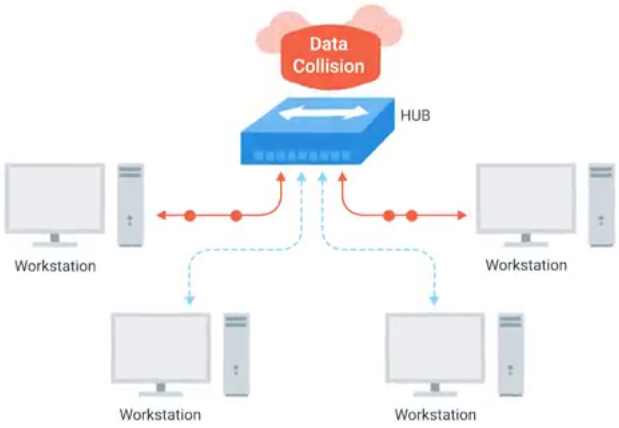
Cables allow you to form point-to-point networking connections. These are networks where only a single device at each end of the link exists. Not to knock point-to-point networking connections, but they're not super useful in a

world with billions of computers. Luckily, there are network devices that allow for many computers to communicate with each other. The most simple of these devices is a hub.

Hub

A hub is a physical layer device that allows for connections from many computers at once.

All the devices connected to a hub will end up talking to all other devices at the same time. It's up to each system connected to the hub to determine if the incoming data was meant for them, or to ignore it if it isn't. This causes a lot of noise on the network and creates what's called a collision domain.



Collision Domain

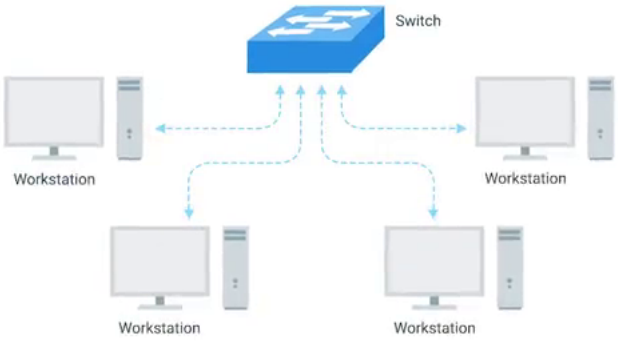
A collision domain is a network segment where only one device can communicate at a time.

If multiple systems try sending data at the same time, the electrical pulses sent across the cable can interfere with each other. This causes these systems to have to wait for a quiet period before they try sending their data again. It really slows down network communications, and is the primary reason hubs are fairly rare. They're mostly a historical artifact today. A much more common way of connecting many computers is with a more sophisticated device, known as a network switch, originally known as a switching hub. A switch is very similar to a hub, since you can connect many devices to it so they can communicate. The difference is that while a hub is a layer 1 or physical layer device, a switch is a level 2 or data link device.

#	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

Switch

A network switch is a hardware device that channels incoming data from multiple input ports to a specific output port that will take it toward its intended destination



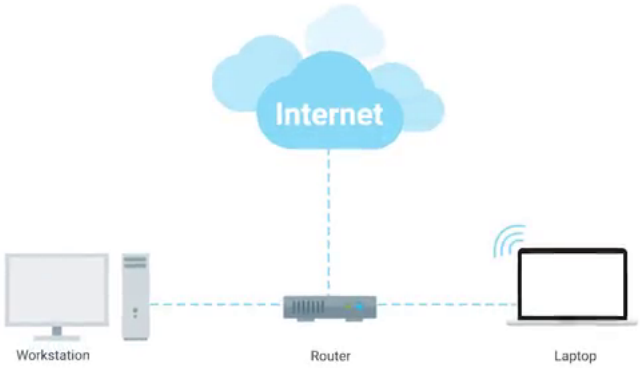
This means that a switch can actually inspect the contents of the Ethernet protocol data being sent around the network, determine which system the data is intended for and then only send that data to that one system. This reduces or even completely eliminates the size of collision domains on a network. This will lead to fewer retransmissions and a higher overall throughput,

1.3.3 Routers

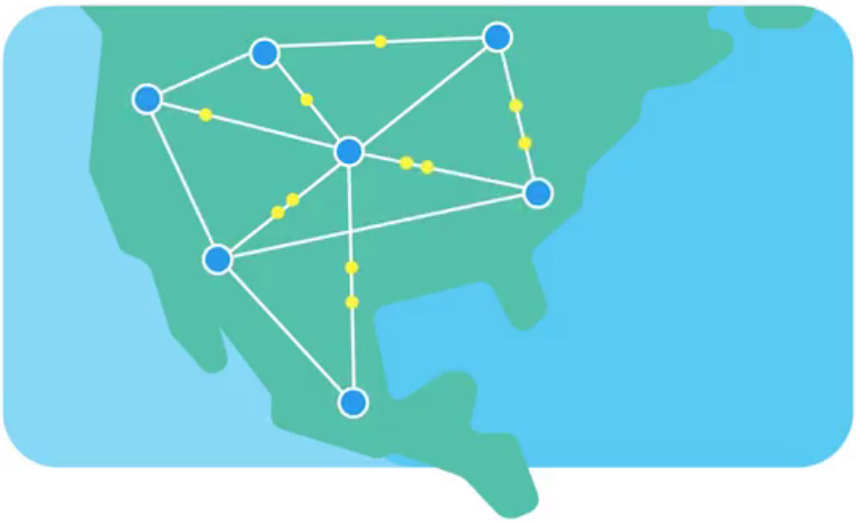
Hubs and switches are the primary devices used to connect computers on a single network, usually referred to as a LAN, or local area network. But we often want to send or receive data to computers on other networks. This is where routers come into play.

Router
A router is a device that knows how to forward data between independent networks.

While a hub is a layer one device and a switch is a layer two device. A router operates at layer three, a network layer. Just like a switch can inspect Ethernet data to determine where to send things, a router can inspect IP data to determine where to send things.



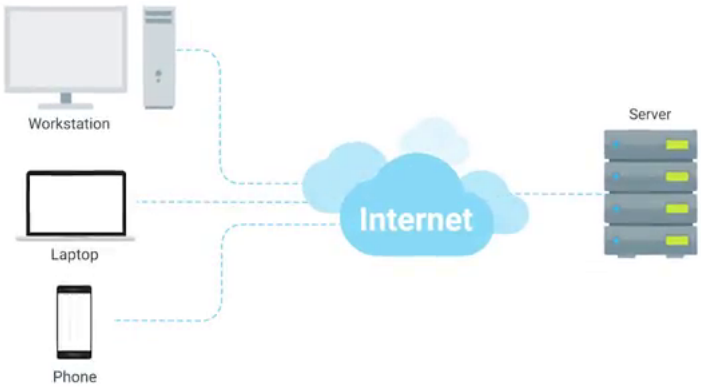
Routers store internal tables containing information about how to route traffic between lots of different networks all over the world. The most common type of router you'll see is one for a home network, or a small office. These devices generally don't have very detailed routing tables. The purpose of these routers is mainly just to take traffic originating from inside the home, or office LAN, and to forward it along to the ISP, or Internet Service Provider. Once traffic is at the ISP, a way more sophisticated type of router takes over. These core routers form the backbone of the Internet and are directly responsible for how we send and receive data all over the Internet every single day. Core ISP routers don't just handle a lot more traffic than a home or a small office router. They also have to deal with much more complexity in making decision about where to send traffic.



A core router usually has many different connections to many other routers. Routers share data with each other via a protocol known as **BGP, or Border Gateway Protocol .** That lets them learn about the most optimal paths to forward traffic. When you open a web browser and load a webpage, the traffic between computers and the web servers could have travelled over dozens of different routers. The Internet is incredibly large and complicated and routers are global guides for getting traffic to the right places.

1.3.4 Server and Clients

All of the network devices you've just learned about, exist so that computers can communicate with each other, whether they're in the same room or thousands of miles apart. We've been calling these devices nodes, but it's also important to understand the concepts of servers and clients. The simplest way to think of a server is as something that provides data to something requesting that data. The thing receiving the data is referred to as a client.



Individual computer programs running on the same node can be servers and clients to each other too. It's also important to call out that most devices aren't purely a server or a client. Almost all nodes are both at some point in time. Quite the multitasking overachievers. That all being said, in most network topographies, each node is primarily either a server or a client. Sometimes we refer to an email server as an email server, even though it's itself a client of a DNS server. Because its primary reason for existing is to serve data to clients. Likewise, if a desktop machine occasionally acts as a server in the sense that it provides data to another computer, its primary reason for existing is to fetch data from servers, so that the user at the computer can do their work. To sum up, a server is anything that can provide data to a client, but we also use the words to refer to the primary purpose of various nodes on our network.

1.4 The Physical Layer

In some ways, the physical layer of our network stack model is the most complexible.

1.4.1 Moving Bits across Wire

Its main focus is on moving ones and zeros from one end of the link to the next. But, very complicated mathematics, physics, and electrical engineering principles are at play to transmit huge volumes of data across tiny wires at incredible speeds. Luckily for us, most of that falls within a different realm. What you, an aspiring IT support specialist needs to know about the physical layer is much more approachable. By the end of this lesson, you should have a solid foundation in aspects of the physical layer that will allow you to properly troubleshoot networking issues, and set up new networks.

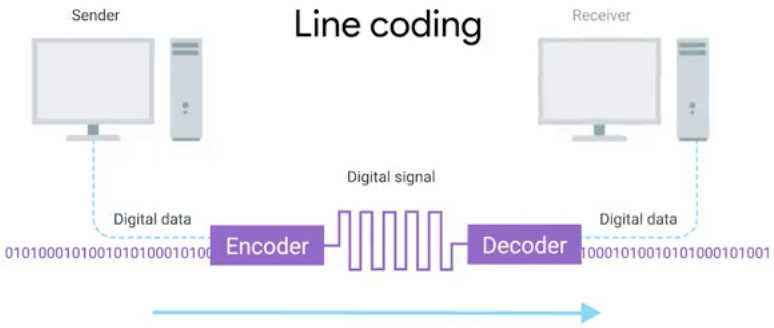
The physical layer consists of devices and means of transmitting bits across computer networks.

Bit
A bit is the smallest representation of data that a computer can understand. It's a one or a zero.

These ones and zeros sends across networks at the lowest level are what make up the frames and packets of data The takeaway is that it doesn't matter whether you're streaming your favourite song, emailing your boss, or using an ATM, what you're really doing is sending ones and zeros across the physical layer of the many different networks between you and the server you're interacting with. A standard copper network cable, once connected to devices on both ends, will carry a constant electrical charge.

Modulation
Ones and zeros are sent across those network cables through a process called modulation.

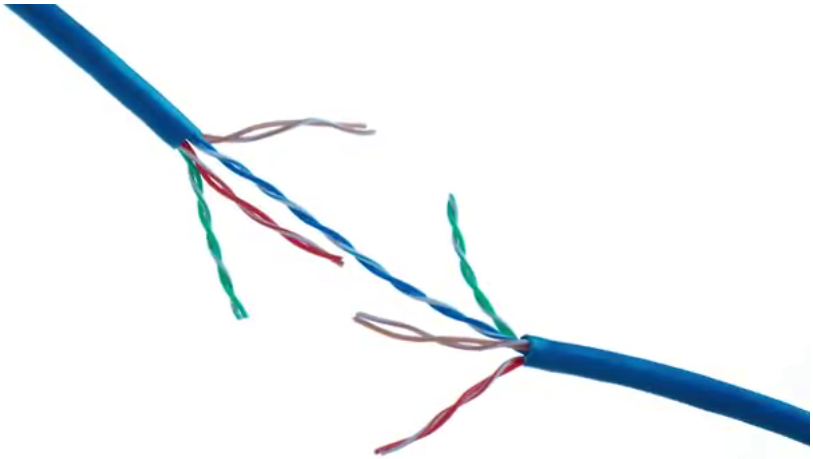
Modulation is a way of varying the voltage of this charge moving across the cable. When used for computer networks, this kind of modulation is more specifically known as line coding.



It allows devices on either end of a link to understand that an electrical charge in a certain state is a zero, and in another state is a one. Through this seemingly simple techniques, modern networks are capable of moving 10 billion ones and zeros across a single network cable every second.

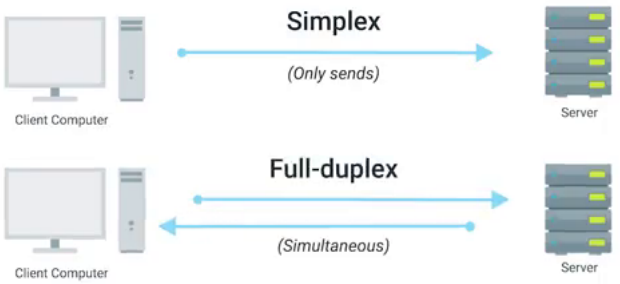
1.4.2 Twisted Pair Cabling and Duplexing

The most common type of cabling used for connecting computing devices is known as twisted pair. It's called a twisted pair cable because it features pairs of copper wires that are twisted together.



These pairs act as a single conduit for information, and their twisted nature helps protect against electromagnetic interference and crosstalk from neighbouring pairs. A standard cat six cable has eight wires consisting of four twisted pairs inside a single jacket. Exactly how many pairs are actually in use depends on the transmission technology being used. But in all modern forms of networking, it's important to know that these cables allow for duplex communication.

Duplex communication
Duplex communication is the concept that information can flow in both directions across the cable. On the flip side, a process called simplex communication is unidirectional .

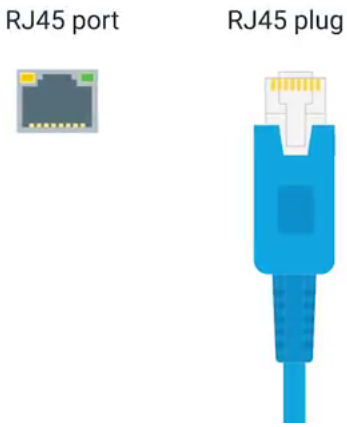


Think about a baby monitor, where the transmission of data only goes in one direction making it a simplex communication. A phone call on the other hand is duplex since both parties can listen and speak. The way networking cables ensure that duplex communication is possible is by reserving one or two pairs for communicating in one direction.

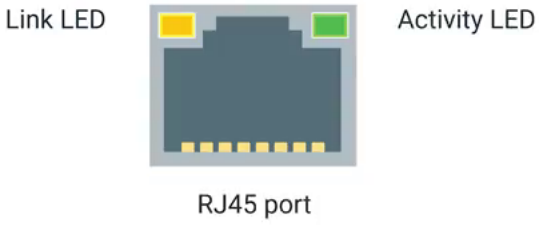
They then use the other one or two pairs for communicating in the other direction. So, devices on either side of a networking link can both communicate with each other at the exact same time. This is known as full duplex . Half-duplex means that, while communication is possible in each direction, only one device can be communicating at a time.

1.4.3 Network Ports and Patch Panels

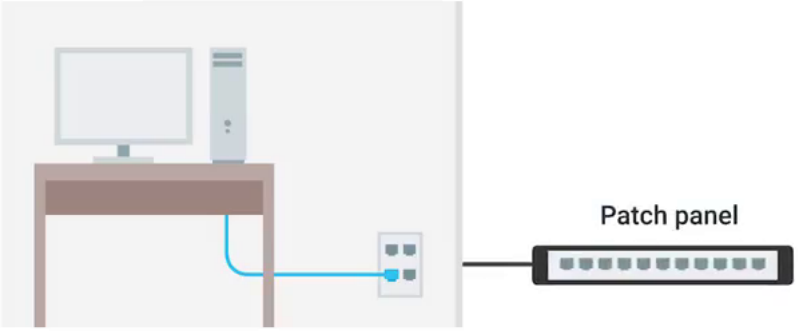
The final steps of how the physical layer works take place at the end points our network links. Twisted Pair network cables are terminated with a plug that takes the individual internal wires and exposes them. The most common plug is known as an RJ-45 or Registered Jack 45.



It's one of many cable plugs specifications but by far the most common in computer networking. A network cable with an RJ-45 plug can connect to an RJ-45 network port. Network ports are generally directly attached to the devices that make up a computer network. Switches would have many network ports because their purpose is to connect many devices. But servers and desktops usually only have one or two.



Most network ports have two small LEDs. One is the link light and the other is the activity light. The link light will be light when a cable is properly connected to two devices that are both powered on. The activity light will flash when data is actively transmitted across the cable. A long time ago, the flashing in the activity light corresponded directly to the one's and zero's being sent. Today, computer networks are so fast that the activity light doesn't really communicate much other than if there's any traffic or not. On switches, sometimes the same LED is used for both link and activity status. It might even indicate other things like links speed. You'll have to read up on a particular piece of hardware you're working with but there will almost always be some troubleshooting data available to you through port lights.



Sometimes a network port isn't connected directly to a device. Instead, there might be network ports mounted in a wall or underneath your desk. These ports are generally connected to the network via cables ran through the walls that eventually end at a patch panel.

Patch panel
A Patch panel is a device containing many net ports but it does no other work.

It's just a container for the endpoints of many runs of cable. Additional cables are then generally ran from a patch panel to switches or routers to provide network access to the computers at the other end of those links.

1.5 The Data Link Layer

1. You'll be able to explain what MAC addresses are and how they're used to identify computers .
2. You'll also know how to describe the various components that make up an Ethernet frame. And you'll be able to differentiate between unicast, multicast and broadcast addresses.
3. You'll be able to explain how cyclical redundancy checks help ensure the integrity of data sent via Ethernet.

1.5.1 Ethernet and MAC Addresses

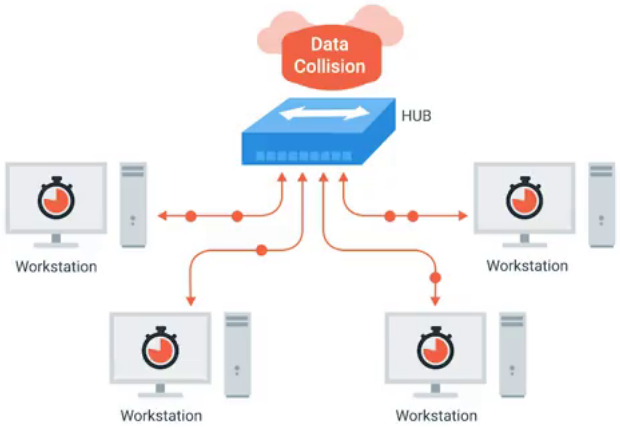
Wireless and cellular internet access are quickly becoming some of the most common ways to connect computing devices to networks, So you might be surprised to hear that traditional cable networks are still the most common option you find in the workplace and definitely in the data center.

Ethernet
The protocol most widely used to send data across individual links is known as Ethernet .

Ethernet and the data link layer provide a means for software at higher levels of the stack to send and receive data. One of the primary purposes of this layer is to essentially abstract away the need for any other layers to care about the physical layer and what hardware is in use. By dumping this responsibility on the data link layer, the Internet, transport and application layers can all operate the same no matter how the device they're running on is connected.
So, for example, your web browser doesn't need to know if it's running on a device connected via a twisted pair or a wireless connection. It just needs the underlying layers to send and receive data for it.

Ethernet is a fairly old technology. It first came into being in 1980 and saw its first fully polished standardization in 1983. Since then, a few changes have been introduced primarily in order to support ever-increasing bandwidth needs. For the most part though, the Ethernet in use today is comparable to the Ethernet standards as first

published all those years ago. In 1983, computer networking was totally different than it is today. One of the notable differences in land topology was that the switch or switchable hub hadn't been invented yet. This meant that frequently, many or all devices on a network shared a single collision domain.



You might remember from our discussion about hubs and switches that a collision domain is a network segment where only one device can speak at a time. This is because all data in a collision domain is sent to all the nodes connected to it. If two computers were to send data across the wire at the same time, this would result in literal collisions of the electrical current representing our ones and zeros, leaving the end result unintelligible.

Ethernet, as a protocol, solved this problem by using a technique known as **Carrier Sense Multiple Access with Collision Detection**. Generally abbreviate as CSMA/CD. CSMA/CD is used to determine when the communications channels are clear and when the device is free to transmit data. If there's no data currently being transmitted on the network segment, a node will feel free to send data. If it turns out that two or more computers end up trying to send data at the same time, the computers detect this collision and stop sending data. Each device involved with the collision then waits a random interval of time before trying to send data again. This random interval helps to prevent all the computers involved in the collision from colliding again the next time they try to transmit anything.

When a network segment is a collision domain, it means that all devices on that segment receive all communication across the entire segment. This means we need a way to identify which node the transmission was actually meant for. This is where something known as a media access control address or **MAC Address** comes into play.

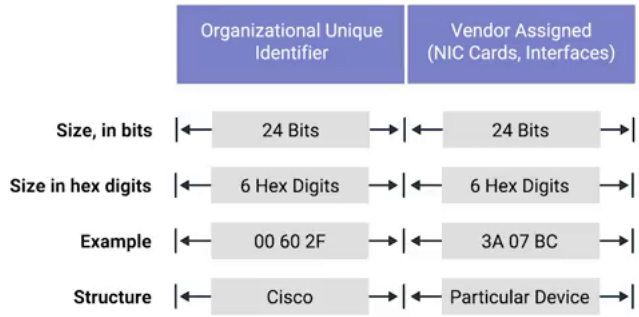
MAC address
A MAC address is a globally unique identifier attached to an individual network interface.

It's a 48-bit number normally represented by six groupings of two hexadecimal numbers. Just like how binary is a way to represent numbers with only two digits, hexadecimal is a way to represent numbers using 16 digits.

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Since we don't have numerals to represent any individual digit larger than nine, hexadecimal numbers employed the letters A, B, C, D, E, and F to represent the numbers 10, 11, 12, 13, 14, and 15. Another way to reference each group of numbers in a MAC address is an octet. An octet, in computer networking, is any number that can be represented by 8 bits. In this case, two hexadecimal digits can represent the same numbers that 8 bits can.

Now, you may have noticed that we mentioned that MAC addresses are globally unique, which might have left you wondering how that could possibly be. The short answer is that a 48-bit number is much larger than you might expect. The total number of a possible MAC addresses that could exist is 2 to the power 48 or 281,474,976,710,656 unique possibilities.



A MAC address is split into two sections.

- The first three octets of a MAC address are known as the organizationally unique identifier or OUI. These are assigned to individual hardware manufacturers by the IEEE or the Institute of Electrical and Electronics Engineers. It means that you can always identify the manufacturer of a network interface purely by its MAC address.
- The last three octets of MAC address can be assigned in any way that the manufacturer would like with the condition that they only assign each possible address once to keep all MAC addresses globally unique.

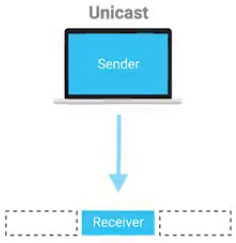
Ethernet uses MAC addresses to ensure that the data it sends has both an address for the machine that sent the transmission, as well as the one that the transmission was intended for. In this way, even on a network segment, acting as a single collision domain, each node on that network knows when traffic is intended for it.

1.5.2 Unicast, Multicast, and Broadcast

A unicast transmission is always meant for just one receiving address. At the Ethernet level, this is done by looking at a special bit in the destination MAC address. One device to transmit data to one other device. This is what's known as unicast.

Unicast
Unicast addresses represent a single LAN interface. A unicast frame will be sent to a specific device, not to a group of devices on the LAN. The unicast address will have the value of the MAC address of the destination device.

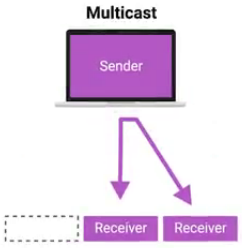
If the least significant bit in the first octet of a destination address is set to zero, it means that Ethernet frame is intended for only the destination address. This means it would be sent to all devices on the collision domain, but only actually received and processed by the intended destination.



If the least significant bit in the first octet of a destination address is set to one, it means you're dealing with a multicast frame.

Multicast Frame
These are frames that are transmitted to a select group of destinations. This would be any frame with the least significant bit of the destination address set to 1, except for broadcast, where all bits of the MAC destination address are set to 1. One example of an Ethernet multicast address would be 01:00:0C:CC:CC:CC, which is the address used by CDP (Cisco Discovery Protocol).

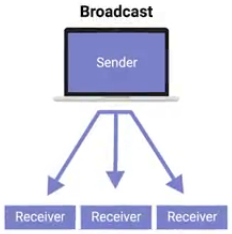
What's different is that it will be accepted or discarded by each device depending on criteria aside from their own hardware MAC address. Network interfaces can be configured to accept lists of configured multicast addresses for these sort of communication.



The third type of Ethernet transmission is known as broadcast .

Broadcast
An Ethernet broadcast is sent to every single device on a LAN.

This is accomplished by using a special destination known as a broadcast address. The Ethernet broadcast address is all Fs. Ethernet broadcasts are used so that devices can learn more about each other.



1.5.3 Dissecting an Ethernet Frame

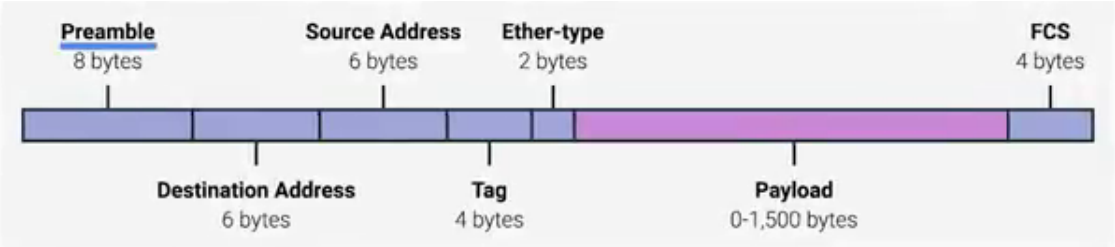
Understanding the networking basics is the first step in building a really strong foundation of networking knowledge that you'll need in IT support. So it's important to know what is inside the Ethernet Frame.

Data Packet
A data packet is an all-encompassing term that represents any single set of binary data being sent across a network link.

The term data packet isn't tied to any specific layer or technology. It just represents a concept. One set of data being sent from point A to Point B. Data packets at the Ethernet level are known as Ethernet frames.

Ethernet frame
An Ethernet frame is a highly structured collection of information presented in a specific order.

This way network interfaces at the physical layer can convert a string of bits, travelling across a link into meaningful data or vice versa. Almost all sections of an Ethernet frame are mandatory and most of them have a fixed size.



1. A preamble is 8 bytes or 64 bits long and can itself be split into two sections. The first seven bytes are a series of alternating ones and zeros. These act partially as a buffer between frames and can also be used by the network interfaces to synchronize internal clocks they use, to regulate the speed at which they send data. This last byte in the preamble is known as the SFD or start frame delimiter.

SFD
This signals to a receiving device that the preamble is over and that the actual frame contents will now follow.

2. Destination MAC address : This is the hardware address of the intended recipient.
3. Source MAC address, or where the frame originated from.

MAC Address Size
Each MAC address is 48 bits or 6 bytes long.

4. EtherType field: It's 16 bits long and used to describe the protocol of the contents of the frame. It's worth calling out that instead of the EtherType field, you could also find what's known as a VLAN header. It indicates that the frame itself is what's called a VLAN frame. If a VLAN header is present, the EtherType field follows it.

VLAN
VLAN stands for virtual LAN. It's a technique that lets you have multiple logical LANs operating on the same physical equipment. Any frame with a VLAN tag will only be delivered out of a switch interface configured to relay that specific tag. This way you can have a single physical network that operates like it's multiple LANs. VLANs are usually used to segregate different forms of traffic. So you might see a company's IP phones operating on one VLAN, while all desktops operate on another.

5. Data payload of an Ethernet frame : A payload in networking terms is the actual data being transported, which is everything that isn't a header. The data payload of a traditional Ethernet frame can be anywhere from 46 to 1500 bytes long. This contains all of the data from higher layers such as the IP, transport and application layers that's actually being transmitted.
6. Frame Check Sequence (FCS) : This is a 4-byte or 32-bit number that represents a checksum value for the entire frame. This checksum value is calculated by performing what's known as a cyclical redundancy check against the frame.

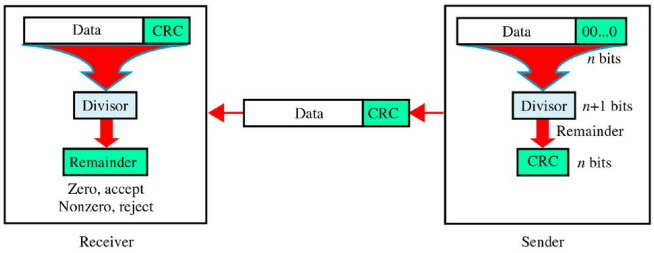
Cyclical Redundancy Check

A cyclical redundancy check or CRC, is an important concept for data integrity and is used all over computing, not just network transmissions. A CRC is basically a mathematical transformation that uses polynomial division to create a number that represents a larger set of data. Anytime you perform a CRC against a set of data, you should end up with the same checksum number. The reason it's included in the Ethernet frame is so that the receiving network interface can infer if it received uncorrupted data.

When a device gets ready to send an Internet frame, it collects all the information we just covered, like the destination and originating MAC addresses, the data payload and so on. Then it performs a CRC against that data and attaches the resulting checksum number as the frame check sequence at the end of the frame. This data is then sent across a link and received at the other end. Here, all the various fields of the Ethernet frame are collected and now the receiving side performs a CRC against that data. If the checksum computed by the receiving end doesn't match the checksum in the frame check sequence field, the data is thrown out. This is because some amount of data must have been lost or corrupted during transmission. It's then up to a protocol at a higher layer to decide if that data should be retransmitted. Ethernet itself only reports on data integrity. It doesn't perform data recovery.

CRC (Cyclic Redundancy Check)

~ is based on binary division.



Chapter 2

Network Layer

Outcome

1. You'll be able to describe the IP addressing scheme and how Subnetting works.
2. You'll also be able to demonstrate how encapsulation works, and how protocols such as ARP allow different layers of the network to communicate.
3. You'll gain an understanding of the basics behind routing, routing protocols and how the internet works.

2.1 Introduction

Computers are able to communicate across massive distances at near instant speeds. It's a remarkable technical advancement at the root of how billions of people use the internet every single day. Earlier in this chapter, we learned about how computers communicate with each other over short distances or on a single network segment or LAN. In this lesson, we'll focus on the technologies that allow data to cross many networks facilitating communications over great distances.

2.2 Network Layer

On a local area network or LAN, nodes can communicate with each other through their physical MAC addresses. This works well on small scale because switches can quickly learn the MAC addresses connected to each other ports to forward transmissions appropriately. But MAC addressing isn't a scheme that scales well, every single network interface on the planet has a unique MAC address and they aren't ordered in any systematic way.
There is no way of knowing where on the planet a certain MAC address might be at any one point in time, so it's not ideal for communicating across distances. when we introduce ARP or Address Resolution Protocol, you'll see that the way that nodes learn about each other's physical addressing isn't translatable to anything besides a single network signet anyway.

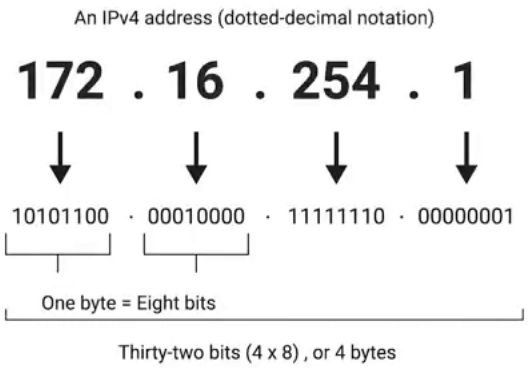
#	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

Clearly we need another solution, and that is the network layer, and the internet protocol or IP in the IP addresses that come along with it. By the end of this lesson you'll be able to take identify an IP address, describe how IP datagrams are encapsulated inside the payload of an ethernet frame and correctly identify and describe the many fields of an IP datagram header.

2.2.1 IP Addresses

IP addresses are 32-bit long numbers made up of 4 octets, and each octet is normally described in decimal numbers. 8 bits of data, or a single octet, can represent all decimal numbers from 0 to 255.

Valid IP address
For example, 12.34.56.78 is a valid IP address. But 123.456.789.100 would not be, because it has numbers larger than could be represented by 8 bits.



This format is known as dotted decimal notation. The important thing to know for now is that IP addresses are distributed in large sections to various organizations and companies, instead of being determined by hardware vendors. This means that IP addresses are more hierarchical, and easier to store data about than physical addresses are. Think of IBM (IT Company), which owns every single IP that has the number 9 as the first octet. At a very high level, this means that if an Internet router needs to figure out where to send a data packet intended for the IP address 9.0.0.1, that router only has to know to get it to one of IBM's routers. That router can handle the rest of the delivery process from there.

Note
IP addresses belong to the networks, not the devices attached to those networks. The network may represent single or multiple nodes.

So your laptop will always have the same MAC address, no matter where you use it. But it'll have a different IP address assigned to it at an Internet cafe than it would when you're at home. The LAN at the Internet cafe or the LAN at your house would each be individually responsible for handing out an IP address to your laptop if you power it on there. On a day-to-day basis, getting an IP address is usually a pretty invisible process. For now, remember that on many modern networks, you can connect a new device. And an IP address will be assigned to it automatically through a technology known as **Dynamic Host Configuration Protocol**. An IP address assigned this way is known as a dynamic IP address. The opposite of this is known as a static IP address, which must be configured on a node, manually.

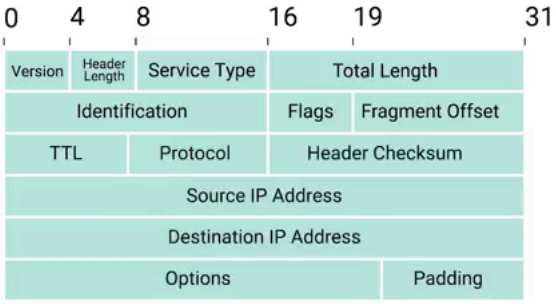
Note
In most cases, static IP addresses are reserved for servers and network devices, while dynamic IP addresses are reserved for clients. But there are certainly situations where this might not be true..

2.2.2 IP Datagrams and Encapsulation

Just like all the data packets at the Ethernet layer have a specific name, Ethernet frames, so do packets at the network layer. Under the IP protocol, a packet is usually referred to as an IP datagram.

IP datagram
IP datagram is a highly structured series of fields that are strictly defined.

The two primary sections of an IP datagram are the header and the payload. You'll notice that an IP datagram header contains a lot more data than an Ethernet frame header does.



1. The very first field is four bits, and indicates what version of Internet protocol is being used.

IPv4 / v6
The most common version of IP is version four or IPv4. Version six or IPv6, is rapidly seeing more widespread adoption.

2. Header Length field : This is also a four bit field that declares how long the entire header is. This is almost always 20 bytes in length when dealing with IPv4. In fact, 20 bytes is the minimum length of an IP header.
3. Service Type field : These eight bits can be used to specify details about quality of service or QoS technologies.

Note
The important takeaway about QoS is that there are services that allow routers to make decisions about which IP datagram may be more important than others.

4. Total Length field It's a 16 bit field to indicate the total length of the IP datagram it's attached to.
5. Identification field, is a 16-bit number that's used to group messages together

Note
IP datagrams have a maximum size Since the Total Length field is 16 bits, and this field indicates the size of an individual datagram, the maximum size of a single datagram is the largest number you can represent with 16 bits: 65,535. If the total amount of data that needs to be sent is larger than what can fit in a single datagram, the IP layer needs to split this data up into many individual packets. When this happens, the identification field is used so that the receiving end understands that every packet with the same value in that field is part of the same transmission.

6. The flag field and the Fragmentation Offset field. The flag field is used to indicate if a datagram is allowed to be fragmented, or to indicate that the datagram has already been fragmented.

Note
Fragmentation is the process of taking a single IP datagram and splitting it up into several smaller datagrams. While most networks operate with similar settings in terms of what size an IP datagram is allowed to be, sometimes, this could be configured differently. If a datagram has to cross from a network allowing a larger datagram size to one with a smaller datagram size, the datagram would have to be fragmented into smaller ones. The fragmentation offset field contains values used by the receiving end to take all the parts of a fragmented packet and put them back together in the correct order.

7. The Time to Live or TTL field. This field is an 8-bit field that indicates how many router hops a datagram can traverse before it's thrown away.

> **Note**
>
> Every time a datagram reaches a new router, that router decrements the TTL field by one. Once this value reaches zero, a router knows it doesn't have to forward the datagram any further. The main purpose of this field is to make sure that when there's a misconfiguration in routing that causes an endless loop, datagrams don't spend all eternity trying to reach their destination. An endless loop could be when router A thinks router B is the next hop, and router B thinks router A is the next hop.

8. Protocol field: This is another 8-bit field that contains data about what transport layer protocol is being used.

> **Note**
>
> The most common transport layer protocols are TCP and UDP.

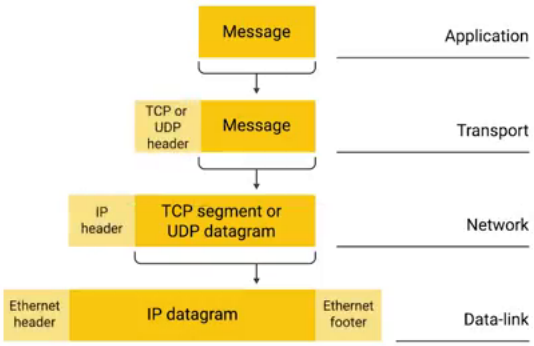
9. Header Checksum Field : This field is a checksum of the contents of the entire IP datagram header. It functions very much like the Ethernet checksum field.

> **Note**
>
> Since the TTL field has to be recomputed at every router that a datagram touches, the checksum field necessarily changes, too.

10. The source and destination IP address fields. (IP address is a 32 bit number so, it should come as no surprise that these fields are each 32 bits long.)
11. IP options field : This is an optional field and is used to set special characteristics for datagrams primarily used for testing purposes.

> **Note**
>
> The IP options field is usually followed by a padding field. Since the IP options field is both optional and variable in length, the padding field is just a series of zeros used to ensure the header is the correct total size.

Now that you know about all of the parts of an IP datagram, you might wonder how this relates to what we've learned so far. You might remember that in our breakdown of an Ethernet frame, we mentioned a section we described as the data payload section. This is exactly what the IP datagram is, and this process is known as **Encapsulation**.



The entire contents of an IP datagram are encapsulated as the payload of an Ethernet frame. You might have picked up on the fact that our IP datagram also has a payload section. The contents of this payload are the entirety of a TCP or UDP packet . Hopefully, this helps you better understand why we talk about networking in terms of layers. Each layer is needed for the one above it.

2.2.3 IP Address Classes

IP addresses can be split into two sections, the network ID, and the host ID. Earlier, we mentioned that IBM owns all IP addresses that having a nine as the value of the first octet in an IP address. If we take an example IP address of 9.100.100.100, the network ID would be the first octet, and the host ID, would be the second, third and fourth octets.

The address class system is a way of defining how the global IP address space is split up.
There are three primary types of address classes : Class A, class B, and class C.

- Class A addresses are those where the first octet is used for the network ID, and the last three are used for the host ID.
- Class B addresses are where the first two octets are used for the network ID, and the second two, are used for the host ID.
- Class C addresses, are those where the first three octets are used for the network ID, and only the final octet is used for the host ID.

IP address classes

Class	Range	Max Hosts
A	0-126	16 Million
B	128-191	64,000
C	192-224	254
D	224-239	N/A
E	240-255	N/A

Each address class represents a network of vastly different size. For example, since a class A network has a total of 24 bits of host ID space, this comes out to two to the twenty power to fourth, or 16,777,216 individual addresses. Compare this with a class C network, which only has eight bits of host ID space. For a class C network, this comes out to two to the eighth, or 256 addresses.

You can also tell exactly what address class an IP address belongs to just by looking at it :

class A: if the very first bit of an IP address is a zero, it belongs to a class A network.

class B: if the first bits are 10, it belongs to a class B network.

class C: if the first bits are 110, it belongs to a class C network.

Since humans aren't great at thinking in binary, it's good to know that this also translates nicely to how these addresses are represented in dotted decimal notation. As each octet in an IP address is eight bits, which means each octet can take a value between 0 and 255. If the first bit has to be a zero, as it is with the class A address, the possible values for the first octet are zero through 127. This means that any IP address with a first octet with one of those values is a class A address. Similarly, class B addresses are restricted to those that begin with the first octet value of 128 through 191, and class C addresses begin with the first octet value of 192 through to 223. You might notice that this doesn't cover every possible IP address. That's because there are two other IP address classes, but they're not quite as important to understand.
Class D addresses always begin with the bits 1110, and are used for multicasting, which is how a single IP datagram can be sent to an entire network at once. These addresses begin with decimal values between 224 and 239.
Lastly, class C addresses make up all of the remaining IP addresses, but they're unassigned and only used for testing purposes.
In practical terms, this class system has mostly been replaced by a system known as CIDR or Classless inter-domain routing.

2.2.4 Address Resolution Protocol

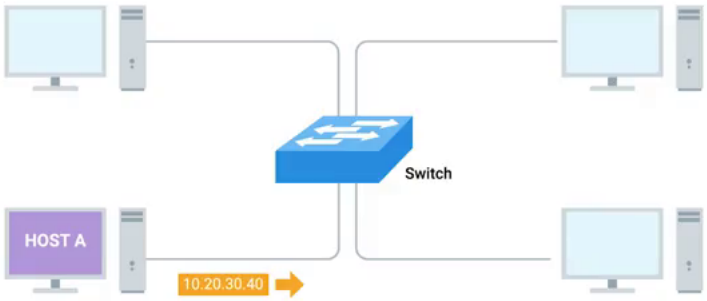
MAC addresses are used at the Data Link Layer and IP addresses are used at the network layer. Now we need to discuss how these two separate addresses types relate to each other. This is where Address Resolution Protocol or ARP comes into play.

ARP
ARP is a protocol used to discover the hardware address of a node with a certain IP address.

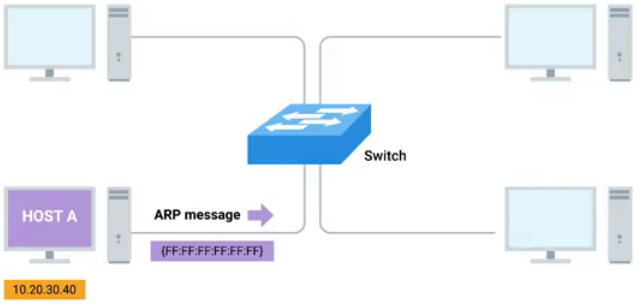
Once an IP datagram has been fully formed, it needs to be encapsulated inside an Ethernet frame. This means, that the transmitting device needs a destination MAC address to complete the Ethernet frame header. Almost all network connected devices while retaining local ARP table.

ARP Table
ARP table is just a list of IP addresses and the MAC addresses associated with them

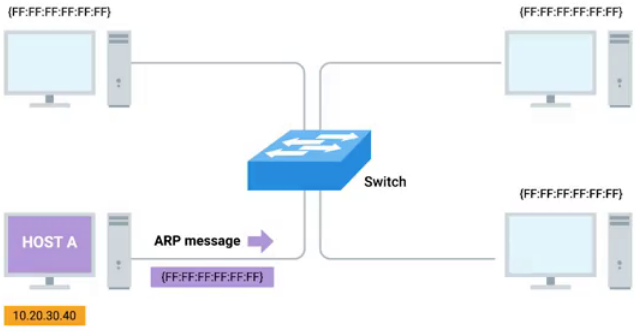
Let's say we want to send some data to the IP address 10.20.30.40. It might be the case that this destination doesn't have an entry in the ARP table.



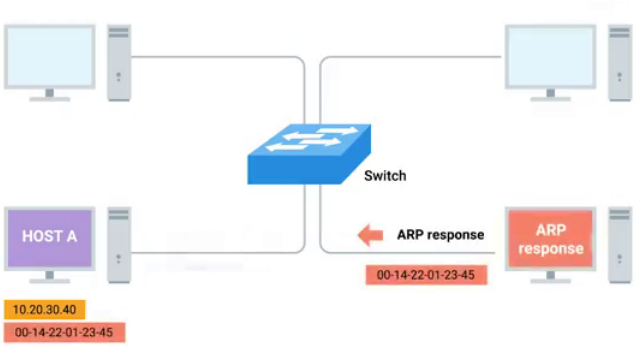
When this happens, the node that wants to send data sends a broadcast ARP message to the Mac broadcast address which is all FFs.



These kinds of broadcast ARP messages are delivered to all computers on the local network.



When the network interface that's been assigned an IP of 10.20.30.40 receives this ARP broadcast, it sends back what's known as an ARP response. This response message will contain the MAC address for the network interface in question.



Now, the transmitting computer knows what MAC address to put in the destination hardware address field and the Ethernet frame is ready for delivery. It will also likely store this IP address in its local ARP table so that it won't have to send an ARP broadcast the next time he needs to communicate with this IP.

Note
ARP table entries generally expire after a short amount of time to ensure changes in the network are accounted for.

2.3 Subnetting

In the most basic of terms, **Subnetting is the process of taking a large network and splitting it up into many individual smaller subnetworks or subnets**.

1. You'll be able to explain why subnetting is necessary and describe how subnet masks extend what's possible with just network and host IDs.
2. You'll also be able to discuss how a technique known as CIDR allows for even more flexibility than plain subnetting.
3. You'll be able to apply some basic binary math techniques to better understand how all of this works.

Address classes give us a way to break the total global IP space into discrete networks. If you want to communicate with the IP address 9.100.100.100, core routers on the Internet know that this IP belongs to the 9.0.0.0 class A network. They then route the message to the gateway router responsible for the network by looking at the network ID. A gateway router specifically serves as the entry and exit path to a certain network.
You can contrast this with core Internet routers, which might only speak to other core routers. Once your packet gets to the gateway router for the 9.0.0.0 class A network, that router is now responsible for getting that data to the proper system by looking at the host ID. This all makes sense until you remember that a single class A network contains 16,777,216 individual IPs. That's just way too many devices to connect to the same router. This is where subnetting comes in. With subnets, you can split your large network up into many smaller ones. These individual subnets will all have their own gateway routers serving as the ingress and egress point for each subnet.

2.3.1 Subnet Masks

We've learned about network IDs, which are used to identify networks, and host IDs, which are used to identify individual hosts. You might remember that an IP address is just a 32-bit number. In a world without subnets, a certain number of these bits are used for the network ID, and a certain number of the bits are used for the host ID. In a world with subnetting, some bits that would normally comprise the host ID are actually used for the subnet ID. With all three of these IDs representable by a single IP address, we now have a single 32-bit number that can be accurately delivered across many different networks. At the internet level, core routers only care about the network ID and use this to send the datagram along to the appropriate gateway router to that network. That gateway router then has some additional information that it can use to send that datagram along to the destination machine or the next router in the path to get there. Finally, the host ID is used by that last router to deliver the datagram to the intended recipient machine. Subnet IDs are calculated via what's known as a subnet mask.

Subnet Masks
Just like an IP address, subnet masks are 32-bit numbers that are normally written now as four octets in decimal.

The easiest way to understand how subnet masks work is to compare one to an IP address. Subnet masks are often glossed over as magic numbers.

Let's work with the IP address 9.100.100.100 again. You might remember that each part of an IP address is an octet, which means that it consists of eight bits. The number 9 in binary is just 1001. But since each octet needs eight bits, we need to pad it with some zeros in front. As far as an IP address is concerned, having a number 9 as the first octet is actually represented as 0000 1001. Similarly, the numeral 100 as an eight-bit number is 0110 0100. So, the entire binary representation of the IP address 9.100.100.100 is a lot of ones and zeros.

IP address	9	100	100	100
IP address *(in binary)*	0000 1001	0110 0100	0110 0100	0110 0100
Subnet mask *(in binary)*	1111 1111	1111 1111	1111 1111	0000 0000

255.255.255.0

A subnet mask is a binary number that has two sections. The beginning part, which is the mask itself is a string of ones just zeros come after this, the subnet mask, which is the part of the number with all the ones, tells us what we can ignore when computing a host ID. The part with all the zeros tells us what to keep. Let's use the common subnet mask of 255.255.255.0. This would translate to 24 ones followed by eight zeros. The purpose of the mask or the part that's all ones is to tell a router what part of an IP address is the subnet ID.

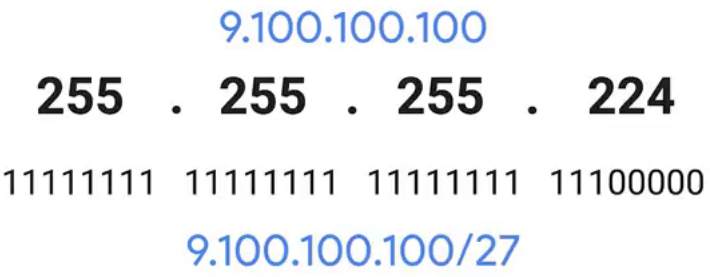
You might remember that we already know how to get the network ID for an IP address. For 9.100.100.100, a Class A network, we know that this is just the first octet. This leaves us with the last three octets. Let's take those remaining octets and imagine them next to the subnet mask in binary form. The numbers in the remaining octets that have a corresponding one in the subnet mask are the subnet ID. The numbers in the remaining octets that have a corresponding zero are the host ID. The size of a subnet is entirely defined by its subnet mask. So for example, with the subnet mask of 255.255.255.0, we know that only the last octet is available for host IDs, regardless of what size the network and subnet IDs are.

A single eight-bit number can represent 256 different numbers, or more specifically, the numbers 0-255. This is a good time to point out that, in general, a subnet can usually only contain two less than the total number of host IDs available. Again, using a subnet mask of 255.255.255.0, we know that the octet available for host IDs can contain the numbers 0-255, but zero is generally not used and 255 is normally reserved as a broadcast address for the subnet. This means that, really, only the numbers 1-254 are available for assignment to a host.

255 . 255 . 255 . 224

11111111 11111111 11111111 11100000

While this total number less than two approach is almost always true, generally speaking, you'll refer to the number of host available in a subnet as the entire number. So, even if it's understood that two addresses aren't available for assignment, you'd still say that eight bits of host IDs space have 256 addresses available, not 254. This is because those other IPs are still IP addresses, even if they aren't assigned directly to a node on that subnet. Now, let's look at a subnet mask that doesn't draw its boundaries at an entire octet or eight bits of address. The subnet mask 255.255.255.224 would translate to 27 ones followed by five zeros. This means that we have five bits of host ID space or a total of 32 addresses.



This brings up a shorthand way of writing subnet masks. Let's say we're dealing with our old friend 9.100.100.100 with a subnet mask of 255.255.255.224. Since that subnet mask represents 27 ones followed by five zeros, a quicker way of referencing this is with the notation /27. The entire IP and subnet mask can be written now as 9.100.100.100/27. Neither notation is necessarily more common than the other, so it's important to understand both

2.3.2 Basic Binary Math

Binary numbers can seem intimidating at first, since they look so different from decimal numbers. But, as far as the basics go the math behind counting, adding, or subtracting binary numbers is exactly the same as with decimal numbers. It's important to call out that there aren't different kinds of numbers. Numbers are universal. There are only different notations for how to reference them. Humans most likely because most of us have ten fingers and ten toes decided on using a system with 10 individual numerals used to represent all numbers. The numerals zero, one, two, three, four, five, six, seven, eight and nine can be combined in ways to represent any whole number in existence. Because there are 10 total numerals in use in a decimal system, another way of referring to this is as base 10. Because of the constraints of how logic gates work inside of a processor, it's way easier for computers to think of things only in terms of zero and one. This is also known as binary or base two.

Binary						Decimal	
32	16	08	04	02	01	10	01
					1		1
				1	0		2
				1	1		3
			1	0	0		4
			1	0	1		5
			1	1	0		6
			1	1	1		7
		1	0	0	0		8
		1	0	0	1		9
		1	0	1	0	1	0
		1	0	1	1	1	1

You can represent all whole numbers in binary in the same way you can in decimal, it just looks a little different. When you count in decimal you move through all of the numerals upward until you run out then you add a second column with a higher significance. Let's start counting at zero until we get to nine. Once we get to nine, we basically just start over we add a one to a new column then start over zero in the original column. We repeat this process over and over in order to count all whole numbers. Counting in binary is exactly the same, it's just that you only have two numerals available. You start with zero, which is the same as zero in decimal. Then you increment once. Now you have one, which is the same as one in decimal since we've already run out of numerals to use. It's time to add a new column. So now we have the number one zero which is the same as two in decimal. One one is three, one zero zero is four, one zero one is five, one one zero is six, one one one is seven, etc. It's the exact same thing we do with decimal, just with fewer numerals at our disposal.

8 bit	$2^8 = 256$	0-255
4 bit	$2^4 = 16$	
16 bit	$2^{16} = 65536$	

When working with various computing technologies, you'll often run into the concept of bits or ones and zeros. There's a pretty simple trick to figure out how many decimal numbers can be represented by a certain number of bits. If you have an eight bit number you can just perform the math two to the power of eight, this gives you 256 which lets you know that an eight bit number can represent 256 decimal numbers, or put another way the numbers zero through 255. A 4 bit number would be two to the power of four, or 16 total numbers. A 16 bit number would be two to the power of 16 or 65,536 numbers.

In order to tie this back to what you might already know, this trick doesn't only work for binary, it works for any number system, it's just the base changes. You might remember, that we can also refer to binary as base two and decimal as base 10. All you need to do is swap out the base for what's being raised to the number of columns. For example, let's take a base 10 number with two columns of digits. This would translate to 10 to the power of two, 10 and the power two equals 100, which is exactly how many numbers you can represent with two columns of decimal digits or the numbers zero then 99. Similarly,10 to the power three is 1,000 which is exactly how many numbers you can represent with three columns of decimal digits or the numbers 0 through 999.

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10$$

Not only is counting in different bases the same, so as simple arithmetic like addition. In fact, binary addition is even simpler than any other base since you only have four possible scenarios. Zero plus zero equals zero just like in decimal. Zero plus one equals one, and one plus zero equals one should also look familiar. One plus one equals one zero looks a little different, but should still make sense. You carried digit to the next column once you reached 10 in doing decimal edition, you carry a digit to the next column once you reach 2 when doing binary edition. Addition is what's known as an operator and there are many operators that computers use to make calculations. Two of the most important operators are OR and AND.

Note

In computer logic, a one represents true and a zero represents false.

The way the OR operator works is you look at each digit, and if either of them is true, the result is true. The basic equation is X or Y equals Z. Which could be read as, if either X or Y is true then Z is true, otherwise, it's false. Therefore one or zero equals one, but zero or zero equals zero.

X OR Y = Z

"If either X or Y is true, then Z is true;
otherwise, it's false."

The operator AND does what it sounds like it does, it returns true if both values are true. Therefore, one and one equals one, but one and zero equals zero, and zero and zero equals zero, and so on.

1 AND 1 = 1
1 AND 0 = 0
0 AND 0 = 0

1 OR 0 = 1

0 OR 0 = 0

A subnet mask is a way for a computer to use and operators to determine if an IP address exists on the same network. This means that the host ID portion is also known, since it will be anything left out. Let's use the binary representation of our IP address 9.100.100.100 and our subnet mask 255.255.255.0. Once you put one on top of the other and perform a binary and operator on each column, you'll notice that the result is the network ID and subnet ID portion of our IP address or 9.100.100.

IP address	9	100	100	100
	AND	AND	AND	AND
Subnet mask *(in binary)*	1111 1111	1111 1111	1111 1111	0000 0000

9.100.100

The computer that just performed this operation can now compare the results with its own network ID to determine if the address is on the same network or a different one.

2.3.3 CIDR

Address classes were the first attempt at splitting up the global Internet IP space. Subnetting was introduced when it became clear that address classes themselves weren't as efficient way of keeping everything organized. But as the Internet continued to grow, traditional subnetting just couldn't keep up. With traditional subnetting and the address classes, the network ID is always either 8 bit for class A networks, 16 bit for class B networks, or 24 bit for class C networks.

Network ID

8 bit	Class A
16 bit	Class B
24 bit	Class C

This means that there might only be 254 classing networks in existence, but it also means there are 2,970,152 potential class C networks. That's a lot of entries in a routing table. To top it all off, the sizing of these networks aren't always appropriate for the needs of most businesses. 254 hosts in a class C network is too small for many use cases, but the 65,534 hosts available for use in a class B network is often way too large. Many companies ended up with various adjoining class C networks to meet their needs. That meant that routing tables ended up with a bunch of entries for a bunch of class C networks that were all actually being routed to the same place. This is where CIDR or classless inter-domain routing comes into play.

Subnet masks and IP address

Class		Mask short name	Max Hosts
A	255.0.0.0 11111111.00000000.00000000.00000000	/8	16,777,214
B	255.255.0.0 11111111.11111111.00000000.00000000	/16	65,534
C	255.255.255.0 11111111.11111111.11111111.00000000	/24	254
	255.255.240.0 11111111.11111111.11110000.00000000	/20	4,094
	255.255.255.224 11111111.11111111.11111111.11100000	/27	30
	255.255.255.252 11111111.11111111.11111111.11111100	/30	2

CIDR is an even more flexible approach to describing blocks of IP addresses. It expands on the concept of subnetting by using subnet masks to demarcate networks. To demarcate something means to set something off. When discussing computer networking, you'll often hear the term demarcation point (to describe where one network or system ends and another one begins). In our previous model, we relied on a network ID, subnet ID, and host ID to deliver an IP datagram to the correct location. With CIDR, the network ID and subnet ID are combined into one.

9.100.100.100

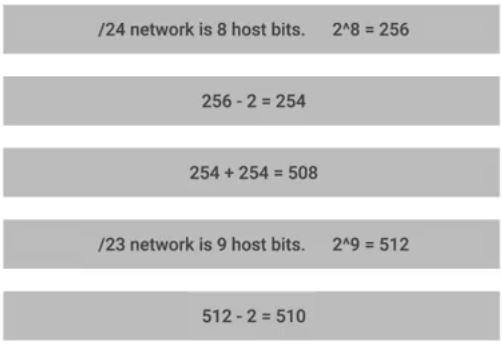
255.255.255.0

9.100.100.10/24

This slash notation is also known as CIDR notation. CIDR basically just abandons the concept of address classes entirely, allowing an address to be defined by only two Individual IDs.

Let's take 9.100.100.100 with a net mask of 255.255.255.0. Remember, this can also be written as 9.100.100.100/24. In a world where we no longer care about the address class of this IP, all we need is what the

network mask tells us to determine the network ID. In this case, that would be 9.100.100, the host ID remains the same. This practice not only simplifies how routers and other network devices need to think about parts of an IP address, but it also allows for more arbitrary network sizes. Before, network sizes were static. Think only class A, class B or, class C, and only subnets could be of different sizes. CIDR allows for networks themselves to be differing sizes. Before this, if a company needed more addresses than a single class C could provide, they need an entire second class C. With CIDR, they could combine that address space into one contiguous chunk with a net mask of /23 or 255.255.254.0. This means, that routers now only need to know one entry in their routing table to deliver traffic to these addresses instead of two. It's also important to call out that you get additional available host IDs out of this practice.



Remember that you always lose two host IDs per network. So, if a /24 network has two to the eight or 256 potential hosts, you really only have 256 minus two, or 254 available IPs to assign. If you need two networks of this size, you have a total of 254 plus 254 or 508 hosts. A single /23 network, on the other hand, is two to the nine or 512. 512 minus two, 510 hosts.

2.4 Basic Routing Concepts

The Internet is an incredibly impressive technological achievement. It meshes together millions of individual networks and allows communications to flow between them. From almost anywhere in the world, you can now access data from almost anywhere else. Often in just fractions of a second. The way communications happen across all these networks, allowing you to access data from the other side of the planet, is through routing.

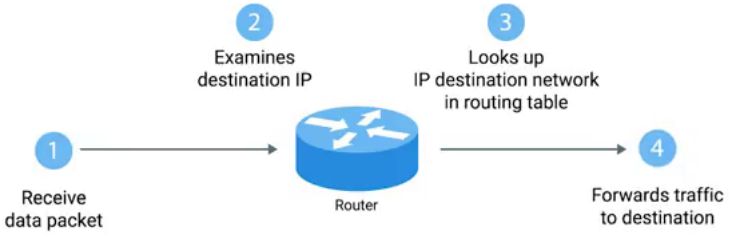
1. You'll be able to describe the basics of routing and how routing tables work.
2. You'll be able to define some of the major routing protocols and what they do and identifying non-routable address space and how it's used.
3. You'll also gain an understanding of the RFC system and how it made the Internet what it is today.

At a very high level, what routing is and how routers work is actually pretty simple. But underneath the hood, routing is a very complex and technologically advanced topic. Today most intensive routing issues are almost exclusively handled by ISPs and only the largest of companies.

Router
A router is a network device that forwards traffic depending on the destination address of that traffic.

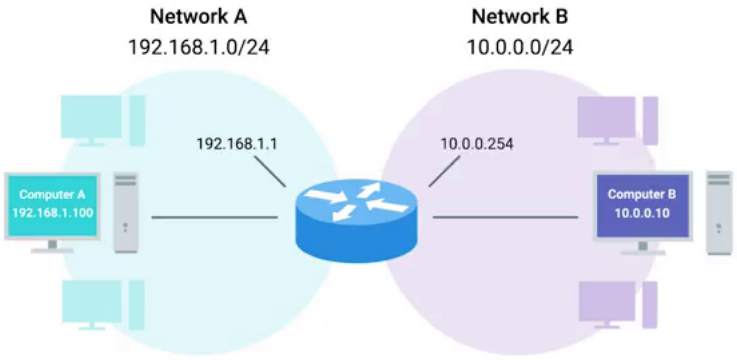
A router is a device that has at least two network interfaces, since it has to be connected to two networks to do its job. Basic routing has just a few steps :

Basic routing:



1. A router receives a packet of data on one of its interfaces.
2. The router examines the destination IP of this packet.
3. The router then looks up the destination network of this IP in its routing table.
4. The router forwards that out though the interface that's closest to the remote network.

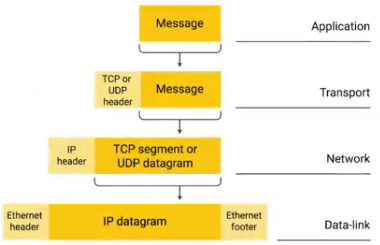
As determined by additional info within the routing table. These steps are repeated as often as needed until the traffic reaches its destination.



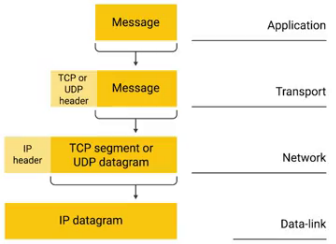
Let's imagine a router connected to two networks. We'll call the first network, Network A and give it an address space of 192.168.1.0/24. We'll call the second network, Network B and give it an address space of 10.0.0.0/24. The router has an interface on each network. On Network A, it has an IP of 192.168.1.1 and on Network B, it has an IP of 10.0.254.

Note
IP addresses belong to networks, not individual nodes on a network.

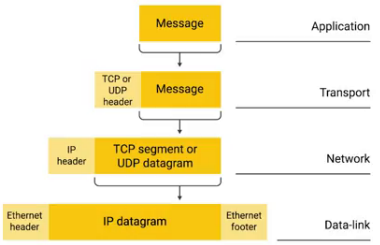
A computer on Network A with an IP address of 192.168.1.100 sends a packet to the address 10.0.0.10. This computer knows that 10.0.0.10 isn't on its local subnet. So it sends this packet to the MAC address of its gateway, the router.



The router's interface on Network A receives the packet because it sees that destination MAC address belongs to it. The router then trips away the Data Link Layer encapsulation, leaving the network layer content, the IP datagram.

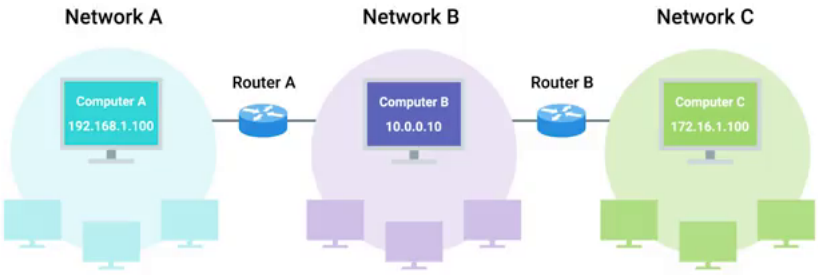


Now, the router can directly inspect the IP datagram header for the destination IP field. It finds the destination IP of 10.0.0.10. The router looks at it's routing table and sees that Network B, or the 10.0.0.0/24 network, is the correct network for the destination IP. It also sees that, this network is only one hop away. In fact, since it's directly connected, the router even has the MAC address for this IP in its ARP table. Next, the router needs to form a new packet to forward along to Network B. It takes all of the data from the first IP datagram and duplicates it. But decrements the TTL field by one and calculates a new checksum. Then it encapsulates this new IP datagram inside of a new Ethernet frame. This time, it sets its own MAC address of the interface on network B as the source MAC address.



Since it has the MAC address of 10.0.0.10 in its ARP table, it sets that as the destination MAC address. Lastly, the packet is sent out of its interface on Network B and the data finally gets delivered to the node living at 10.0.0.10.

That's a pretty basic example of how routing works, but let's make it a little more complicated and introduce a third network. Everything else is still the same. We have network A whose address space is 192.168.1.0/24. We have network B whose address space is 10.0.0/24. The router that bridges these two networks still has the IPs of 192.168.1.1 on Network A and 10.0.0.254 on Network B. But let's introduce a third network, Network C. It has an address space of 172.16.1.0/23.



There is a second router connecting network B and network C. It's interface on network B has an IP of 10.0.0.1 and its interface on Network C has an IP of 172.16.1.1. This time around our computer at 192.168.1.100 wants to send some data to the computer that has an IP of 172.16.1.100. We'll skip the Data Link Layer stuff, but it's still happening,

1. The computer at 192.168.1.100 knows that 172.16.1.100 is not on its local network, so it sends a packet to its gateway, the router between Network A and Network B.
2. Again, the router inspects the content of this packet. It sees a destination address of 172.16.1.100 and through a lookup of its routing table, it knows that the quickest way to get to the 172.16.1.0/23 network is via another router.
3. With an IP of 10.0.0.1. The router decrements the TTL field and sends it along to the router of 10.0.0.1.
4. This router then goes through the motions, knows that the destination IP of 172.16.1.100 is directly connected and forwards the packet to its final destination. That's the basics of routing.

The only difference between our examples and how things work on the Internet is scale. Routers are usually connected to many more than just two networks. Very often, your traffic may have to cross a dozen routers before it reaches its final destination. And finally, in order to protect against breakages, core Internet routers are typically connected in a mesh, meaning that there might be many different paths for a packet to take. Still, the concepts are all the same. Routers inspect the destination IP, look at the routing table to determine which path is the quickest and forward the packet along the path. This happens over and over. Every single pocket making up every single bit of traffic all over the Internet at all times.

2.4.1 Routing Tables

During the earlier topic on the basics of routing, you might have noticed a bunch of references to something known as a routing table. Routing itself is pretty simple concept and you'll find that routing tables aren't that much more complicated. The earliest routers were just regular computers of the era. They had two network interfaces, bridge to networks, and auto-routing table that was manually updated. In fact, all major operating systems today, still have a routing table that they consult before transmitting data. You could still build your own router today, if you had a computer with two network interfaces and it manually updated routing table.

Routing tables can vary a on depending on the version and class of the router, but they all share a few things in common. The most basic routing table will have four columns. **Destination network**, this column would contain a row for each network that the router knows about, this is just the definition of the remote network, a network ID, and the net mask. These could be stored in one column inside a notation, or **the network ID and net mask** might be in a separate column.

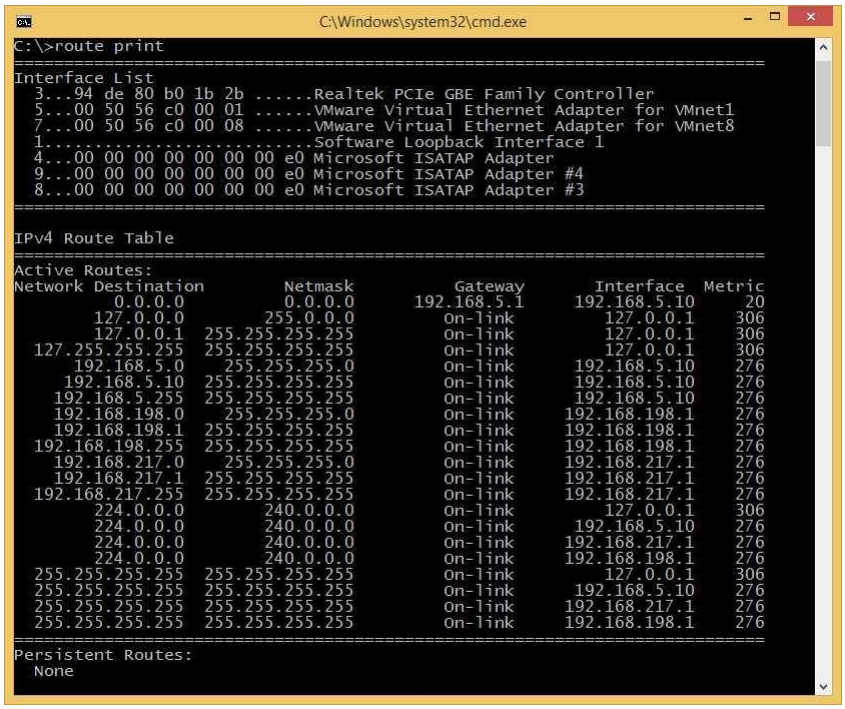
IP: 192.168.1.1

Subnet Mask: 255.255.255.0

CIDR: 192.168.1.1/24

Either way, it's the same concept, the router has a definition for a network and therefore knows what IP addresses might live on that network. When the router receives an incoming packet, it examines the destination IP address and determines which network it belongs to. A routing table will generally have a cache entry, that matches any IP address that it doesn't have an explicit network listing for.

Next hop, this is the IP address of the next router that should receive data intended for the destination networking question or this could just state the network is directly connected and that there aren't any additional hops needed. **Total hops**, this is the crucial part to understand routing and how routing tables work, on any complex network like the Internet, there will be lots of different paths to get from point A to point B.



Routers try to pick the shortest possible path at all times to ensure timely delivery of data but the shortest possible path to a destination network is something that could change over time, sometimes rapidly, intermediary routers could go down, links could become disconnected, new routers could be introduced, traffic congestion could cause certain routes to become too slow to use.

We'll get to know how routers know the shortest path in an upcoming topic. For now, it's just important to know that for each next hop and each destination network, the router will have to keep track of how far away that destination currently is. That way, when it receives updated information from neighboring routers, it will know if it currently knows about the best path or if a new better path is available.

Interface, the router also has to know which of its interfaces it should for traffic matching the destination network out of. In most cases, routing tables are pretty simple. The really impressive part is that, many core Internet routers have millions of rows in the routing tables. These must be consulted for every single packet that flows through a router on its way to its final destination.

2.4.2 Interior Gateway Protocols

We've covered the basics of how routing works and how routing tables are constructed. They're both really pretty basic concepts. The real magic of routing is in the way routing tables are always updated with new information about the quickest path to destination networks. The protocols we'll be learning about in this session will help you identify routing problems on any network you might support. In order to learn about the world around them, routers use what are known as routing protocols. These are special protocols the routers use to speak to each other in order to share what information they might have. This is how a router on one side of the planet can eventually learn about the best path to a network on the other side of the planet.

Routing Protocols
Routing protocols fall into two main categories: Interior Gateway Protocols and Exterior Gateway Protocols. Interior Gateway Protocols are further split into two categories: Link state routing protocols and Distance-vector protocols.

Interior gateway protocols are used by routers to share information within a single autonomous system.

Autonomous System
An autonomous system is a collection of networks that all fall under the control of a single network operator.

The best example of this would be a large corporation that needs to route data between their many offices, and each of which might have their own local area network. Another example is the many routers employed by an Internet service provider whose reaches are usually national in scale.

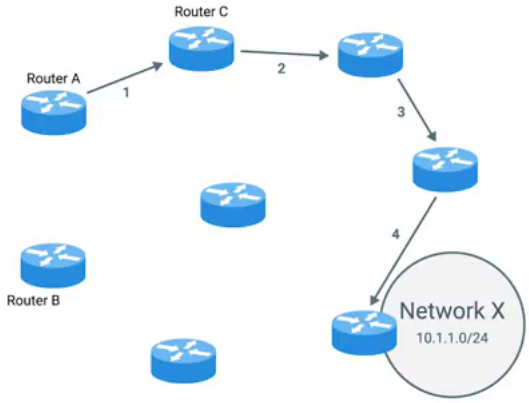
You can contrast this with exterior gateway protocols which are used for the exchange of information between independent autonomous systems. The two main types of interior gateway protocols are link state routing protocols and distance-vector protocols. Their goals are similar but the routers that employ them share different kinds of data to get the job done.

Distance vector protocols are an older standard. A router using a distance vector protocol basically just takes its routing table which is a list of every network known to it and how far away these networks are in terms of hops. Then the router sends this list to every neighboring router, which is basically every router directly connected to it.

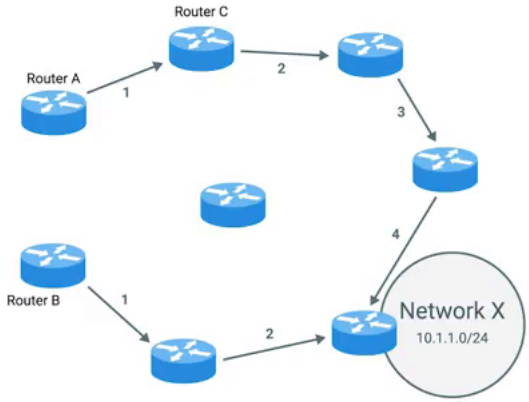
Note

In computer science, a list is known as a vector. This is why a protocol that just sends a list of distances to networks, is known as a distance vector protocol.

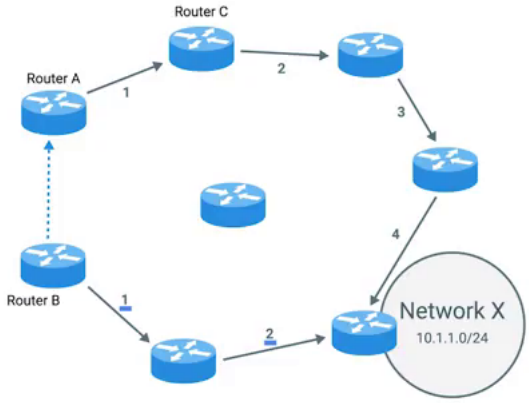
With a distance vector protocol, routers don't really know that much about the total state of an autonomous system. They just have some information about their immediate neighbors.



For a basic glimpse into how distance vector protocols work, let's look at how two routers might influence each other's routing tables. Router A has a routing table with a bunch of entries. One of these entries is for 10.1.1.0/24 network which we'll refer to as Network X. Router A believes that the quickest path to network X is through its own interface two, which is where router C is connected. Router A knows that sending data intended for Network X through interface two to Router C means it'll take four hops to get to the destination.



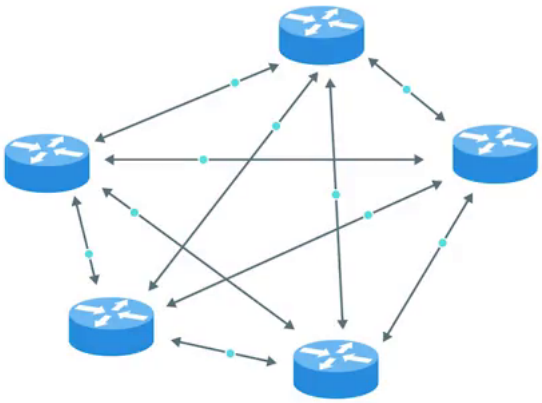
Meanwhile, Router B is only two hops removed from Network X and this is reflected in its routing table. Router B, using a distance vector protocol, sends the basic contents of it's routing table to Router A. Router A sees that Network X is only two hops away from Router B.



Even with the extra hop to get from Router A to Router B, this means that network X is only three hops away from router A if it forwards data to router B instead of router C. Armed with this new information, router A updates its routing table to reflect this. In order to reach network X in the fastest way, it should forward traffic through its own interface 1 to router B.

Now, distance vector protocols are pretty simple. But they don't allow for a router to have much information about the state of the world outside of their own direct neighbors. Because of this, a router might be slow to react to a change in the network far away from it. This is why link-state protocols were eventually invented.

Routers using a link-state protocol take a more sophisticated approach to determining the best path to a network. Linked state protocols got their name because each router advertises the state of the link of each of its interfaces. These interfaces can be connected to other routers or they could be direct connections to networks. The information about each router is propagated to every other router on the autonomous system. This means that every router on the system knows every detail about every other router in the system.



Each router then uses this much larger set of information and runs complicated algorithms against it to determine what the best path to any destination network might be.
Link state protocols require both more memory in order to hold all of this data and also much more processing power. This is because it has to run algorithms against this data in order to determine the quickest path to update the routing tables. As computer hardware has become more powerful and cheaper over the years, link state protocols have mostly made distance vector protocols outdated.

2.4.3 Exterior Gateway Protocols

Exterior gateway protocols are used to communicate data between routers representing the edges of an autonomous system. Since routers sharing data using interior gateway protocols are all under control of the same organization. Routers use exterior gateway protocols when they need to share information across different organizations. Exterior gateway protocols are really key to the Internet operating how it does today.

The Internet is an enormous mesh of autonomous systems. At the highest levels, core Internet routers need to know about autonomous systems in order to properly forward traffic. Since autonomous systems are known and defined collections of networks, getting data to the edge router of an autonomous system is the number one goal of core Internet routers.

IANA
The IANA or the Internet Assigned Numbers Authority, is a non-profit organization that helps manage things like IP address allocation.

The Internet couldn't function without a single authority for these sorts of issues. Otherwise, anyone could try and use any IP space they wanted, which would cause total chaos online. Along with managing IP address allocation, the IANA is also responsible for ASN, or Autonomous System Number allocation. ASNs are numbers assigned to individual autonomous systems. Just like IP addresses, ASNs are 32-bit numbers. But, unlike IP addresses, they're normally referred to as just a single decimal number, instead of being split out into readable bits. There are two reasons for this. First, IP addresses need to be able to represent a network ID portion and a host ID portion for each number. This is more easily accomplished by splitting the number in four sections of eight bits, especially back in the day when address classes ruled the world.

An ASN, never needs to change in order for it to represent more networks or hosts. Its just the core Internet routing tables that need to be updated to know what the ASN represents. Second, ASNs are looked at by humans, far less often, than IP addresses are. So because it can be useful to be able to look at the IP 9.100.100.100 and know that 9.0.0.0/8 address space is owned by IBM, ASNs represent entire autonomous systems. Just being able to look up the fact that AS19604 belongs to IBM is enough. Unless you one day end up working at an Internet service provider, understanding more details about how exterior gateway protocols work, is out of scope for most people in IT. But grasping the basics of autonomous systems, ASNs, and how core Internet routers route traffic between them, is important to understand some of the basic building blocks of the Internet.

2.4.4 Non-Routable Address Space

And now, a brief history lesson. Even as far back as 1996, it was obvious that the internet was growing at a rate that couldn't be sustained. When IP was first defined, it defined an IP address as a single 32-bit number. A single 32-bit number can represent 4,294,967,295 unique numbers which definitely sounds like a lot, but as of 2017, there are an estimated 7.5 billion humans on earth. **This means that the IPv4 standard doesn't even have enough IP addresses available for every person on the planet.**

It also can account for entire data centers filled with thousands and thousands of computers required for large scale technology companies to operate. So, in 1996, RFC 1918 was published. RFC stands for Request for Comments, and has a long standing way for those responsible for keeping the internet running to agree upon the standard requirements to do so. RFC 1918, outlined a number of networks that would be defined as non-routable address space.

Non-routable address space is basically exactly what it sounds like. They are ranges of IPs set aside for use by anyone that cannot be routed to. Not every computer connected to the internet needs to be able to communicate with every other computer connected to the internet. Non-routable address space allows for nodes on such a network to communicate with each other but no gateway router will attempt to forward traffic to this type of network. This might sound limiting, and in some ways, it is.

In a future module, we'll cover a technology known as NAT or Network Address Translation. It allows for computers on non-routable address space to communicate with other devices on the internet. But for now, let's just discuss non-routable address space in a vacuum. RFC 1918 defined three ranges of IP addresses that will never be routed anywhere by co-routers. That means that they belong to no one and that anyone can use them. In fact, since they are separated from the way traffic moves across the internet, there's no limiting to how many people might use these addresses for their internal networks.

Note
The primary three ranges of non-routable address space are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

These ranges are free for anyone to use for their internal networks. It should be called out that interior gateway protocols will route these address spaces. So, they are appropriate for use within an autonomous system but exterior gateway protocols will not.

2.5 IPv6

IPv6 is the newest version of the IP protocol. IPv6 was developed to overcome many deficiencies of IPv4, most notably the problem of IPv4 address exhaustion. Unlike IPv4, which has only about 4.3 billion (2 raised to power 32) available addresses, IPv6 allows for 3.4 × 10 raised to power 38 addresses.
Here is a list of the most important features of IPv6:

Large address space: IPv6 uses 128-bit addresses, which means that for each person on the Earth there are 48,000,000,000,000,000,000,000,000,000 addresses!

Enhanced security: IPSec (Internet Protocol Security) is built into IPv6 as part of the protocol . This means that two devices can dynamically create a secure tunnel without user intervention.

Header improvements: The packed header used in IPv6 is simpler than the one used in IPv4. The IPv6 header is not protected by a checksum so routers do not need to calculate a checksum for every packet.

No need for NAT: Since every device has a globally unique IPv6 address, there is no need for NAT.

Stateless address autoconfiguration: IPv6 devices can automatically configure themselves with an IPv6 address.

Chapter 3

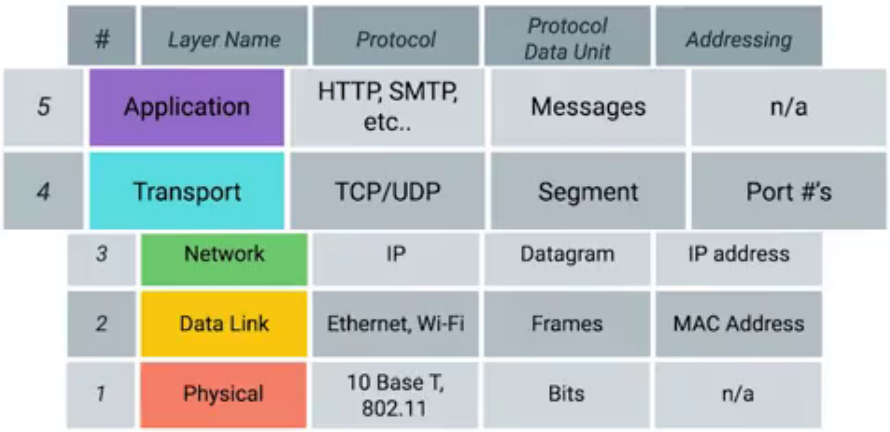
Transport and Application Layer

Outcome

1. You'll be able to have deep understanding in Transport and Application Layer.
2. You'll be able to describe TCP ports and sockets and identify the different components of a TCP header.
3. You'll also be able to show the difference between connection oriented, and connection lists protocols, and explain how TCP is used to ensure data integrity.
4. You should be able to describe what multiplexing and demultiplexing are, and how they work.
5. You'll be able to identify the differences between TCP and UDP, explain the three way handshake, and understand how TCP flags are used in this process.
6. You'll be able to describe the basics of how firewalls keep networks safe.

3.1 Intro to Transport and Application Layer

The first three layers of our Network Model have helped us describe how individual nodes on a network can communicate with other nodes on either their own network or a remote one. Now how individual computer programs can communicate with each other We network computers together, not just so they can send data to each other, but because we want programs running on those computers to be able to send data to each other. This is where the Transport and Application layers of networking model come into play.



#	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

Transport layer
Transport layer allows traffic to be directed to specific network applications.

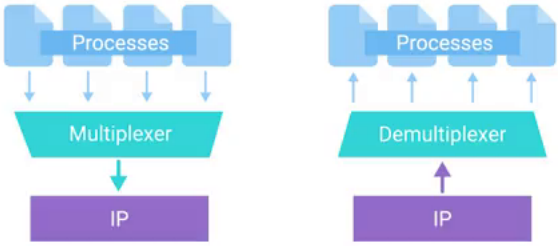
Application layer
Application layer allows these applications to communicate in a way they understand.

3.2 The Transport Layer

The transport layer is responsible for lots of important functions of reliable computer networking. These include multiplexing and demultiplexing traffic, establishing long running connections and ensuring data integrity through error checking and data verification. The transport layer has the ability to multiplex and demultiplex, which sets this layer apart from all others.

Multiplexing in the transport layer means that nodes on the network have the ability to direct traffic toward many different receiving services.

Demultiplexing is the same concept, just at the receiving end, it's taking traffic that's all aimed at the same node and delivering it to the proper receiving service.



Port

The transport layer handles multiplexing and demultiplexing through ports. A port is a 16-bit number that's used to direct traffic to specific services running on a networked computer.

A server or service is a program running on a computer waiting to be asked for data. A client is another program that is requesting this data. Different network services run while listening on specific ports for incoming requests. For example, the traditional port for HTTP or unencrypted web traffic is port 80. If we want tor request the webpage from a web server running on a computer listening on IP 10.1.1.100, the traffic would be directed to port 80 on that computer. Ports are normally denoted with a colon after the IP address.

Note

Full IP and port in this scenario could be described as 10.1.1.100:80 . When written this way, it's known as a socket address or socket number.

The same device might also be running an FTP or file transfer protocol server. FTP is an older method used for transferring files from one computer to another, but you still see it in use today. FTP traditionally listens on port 21, so if you wanted to establish a connection to an FTP server running on the same IP that our example web server was running on, you direct traffic to 10.1.1.100 port 21. In the environments, a single server could host almost all of the applications needed to run a business. The same computer might host an internal website, the mail server for the company, file server for sharing files, a print server for sharing network printers, etc. This is all possible because of multiplexing and demultiplexing, and the addition of ports to our addressing scheme.

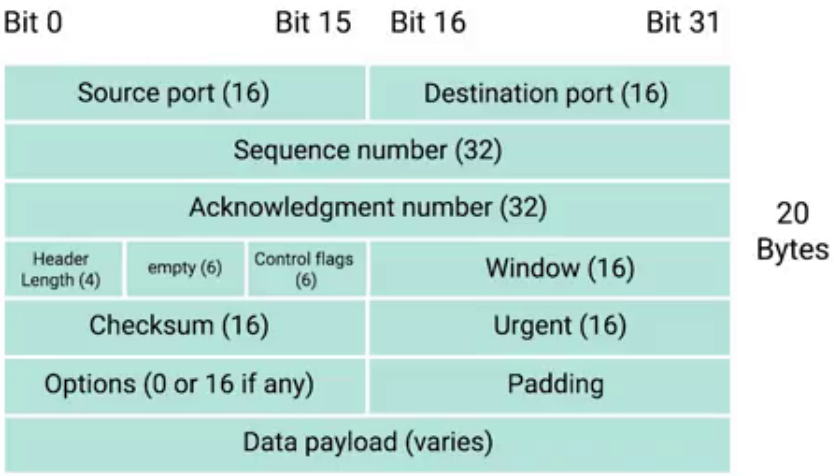
3.2.1 Dissection of a TCP Segment

In IT support, if network traffic isn't behaving as users expect it to, you might have to analyze it closely to troubleshoot. Just like how an Ethernet frame encapsulates an IP datagram, an IP datagram encapsulates a TCP segment. Remember that an Ethernet frame has a payload section which is the entire contents of an IP datagram. Remember also that an IP datagram has a payload section and this is made up of what's known as a TCP segment.

Note

A TCP segment is made up of a TCP header and a data section.

This data section is just another payload area for where the application layer places its data. A TCP header itself is split into lots of fields containing lots of information.



1. The source port and the destination port fields. The destination port is the port of the service the traffic is intended for. A source port is a high numbered port chosen from a special section of ports known as ephemeral ports.

Note
A source port is required to keep lots of outgoing connections separate. You know how a destination port, say port 80, is needed to make sure traffic reaches a web server running on a certain IP. Similarly, a source port is needed so that when the web server replies, the computer making the original request can send this data to the program that was actually requesting it. It is in this way that when it web server responds to your requests to view a webpage that this response gets received by your web browser and not your word processor.

2. The sequence number : This is a 32-bit number that's used to keep track of where in a sequence of TCP segments this one is expected to be.

Note
You might remember that lower on our protocol stack, there are limits to the total size of what we send across the wire. In Ethernet frame, it's usually limited in size to 1,518 bytes, but we usually need to send way more data than that. At the transport layer, TCP splits all of this data up into many segments. The sequence number in a header is used to keep track of which segment out of many this particular segment might be.

3. The acknowledgment number, is a lot like the sequence number. The acknowledgment number is the number of the next expected segment.

Note
A sequence number of one and an acknowledgement number of two could be read as this is segment one, expect segment two next.

4. The data offset field : This field is a four-bit number that communicates how long the TCP header for this segment is.

> **Note**
> This is so that the receiving network device understands where the actual data payload begins.

5. Six bits that are reserved for the six TCP control flags.
6. 16-bit number known as the TCP window. A TCP window specifies the range of sequence numbers that might be sent before an acknowledgement is required.

> **Note**
> TCP is a protocol that's super reliant on acknowledgements. This is done in order to make sure that all expected data is actually being received and that the sending device doesn't waste time sending data that isn't being received.

7. 16-bit checksum. It operates just like the checksum fields at the IP and Ethernet level.

> **Note**
> Once all of this segment has been ingested by a recipient, the checksum is calculated across the entire segment and is compared with the checksum in the header to make sure that there was no data lost or corrupted along the way.

8. The Urgent pointer field is used in conjunction with one of the TCP control flags to point out particular segments that might be more important than others.

> **Note**
> This is a feature of TCP that hasn't really ever seen adoption.

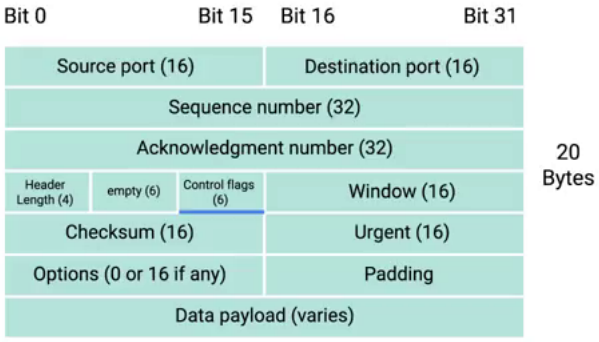
9. The options field.

> **Note**
> Like the urgent pointer field, this is rarely used in the real world, but it's sometimes used for more complicated flow control protocols.

10. Padding which is just a sequence of zeros to ensure that the data payload section begins at the expected location.

3.2.2 TCP Control Flags and the Three-way Handshake

As a protocol, TCP establishes connections used to send long chains of segments of data. You can contrast this with the protocols that are lower in the networking model. These include IP and Ethernet, which just send individual packets of data. As an IT support specialist, you need to understand exactly how that works, so you can troubleshoot issues, where network traffic may not be behaving in the expected manner. The way TCP establishes a connection, is through the use of different TCP control flags, used in a very specific order.



Before we cover how connections are established and closed, let's first define the six TCP control flags. We'll look at them in the order that they appear in a TCP header. This isn't necessarily in the same order of how frequently they're set, or how important they are.

1. URG, this is short for Urgent. A value of one here indicates that the segment is considered urgent and that the urgent pointer field has more data about this. This feature of TCP has never really had wide spreaded option and isn't normally seen.

2. ACK, short for acknowledge. A value of one in this field means that the acknowledgment number field should be examined.

3. PSH, which is short for Push. This means, that the transmitting device wants the receiving device to push currently- buffered data to the application on the receiving end as soon as possible.

   Buffer

   A buffer is a computing technique, where a certain amount of data is held somewhere, before being sent somewhere else. This has lots of practical applications. In terms of TCP, it's used to send large chunks of data more efficiently. By keeping some amount of data in a buffer, TCP can deliver more meaningful chunks of data to the program waiting for it. But in some cases, you might be sending a very small amount of information, that you need the listening program to respond to immediately. This is what the push flag does.

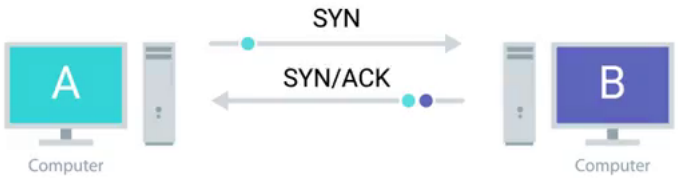
4. RST, short for Reset. This means, that one of the sides in a TCP connection hasn't been able to properly recover from a series of missing or malformed segments. It's a way for one of the partners in a TCP connection to basically say, "Wait, I can't put together what you mean, let's start over from scratch."

5. s SYN, which stands for Synchronize. It's used when first establishing a TCP connection and make sure the receiving end knows to examine the sequence number field.

6. FIN, which is short for Finish. When this flag is set to one, it means the transmitting computer doesn't have any more data to send and the connection can be closed.

For a good example of how TCP control flags are used, let's check out how a TCP connection is established. Computer A will be our transmitting computer and computer B will be our receiving computer. To start the process off, computer A, sends a TCP segment to computer B with this SYN flag set. This is computer A's way of saying, "Let's establish a connection and look at my sequence number field, so we know where this conversation starts."

Computer B then responds with a TCP segment, where both the SYN and ACK flags are set. This is computer B's way of saying, "Sure, let's establish a connection and I acknowledge your sequence number."



Then computer A responds again with just the ACK flag set, which is just saying, "I acknowledge your acknowledgement. Let's start sending data." This exchange involving segments that have SYN, SYN/ACK and ACK sets, happens every single time a TCP connection is established anywhere.

It is so famous that it has a nickname. The three way handshake.

Handshake
A handshake is a way for two devices to ensure that they're speaking the same protocol and will be able to understand each other.

Once the three way handshake is complete, the TCP connection is established. Now, computer A is free to send whatever data it wants to computer B and vice versa. Since both sides have now sent SYN/ACK pairs to each other, a TCP connection in this state is operating in full duplex. Each segment sent in either direction should be responded to by TCP segment with the ACK field set. This way, the other side always knows what has been received.

Once one of the devices involved with the TCP connection is ready to close the connection, something known as a four way handshake happens. The computer ready to close the connection, sends a FIN flag, which the other computer acknowledges with an ACK flag.

The Four-Way Handshake

Then, if this computer is also ready to close the connection, which will almost always be the case. It will send a FIN flag. This is again responded to by an ACK flag.

Hypothetically, a TCP connection can stay open in simplex mode with only one side closing the connection. But this isn't something you'll run into very often. Up next, I'll sock it to you, with information on TCP socket states.

3.2.3 TCP Socket States

TCP sockets require actual programs to instantiate them. You can contrast this with a port which is more of a virtual descriptive thing.

Note
A socket is the instantiation of an endpoint in a potential TCP connection. An instantiation is the actual implementation of something defined elsewhere.

In other words, you can send traffic to any port you want, but you're only going to get a response if a program has opened a socket on that port. TCP sockets can exist in lots of states. And being able to understand what those mean will help you troubleshoot network connectivity issues as an IT support specialist.

1. LISTEN : Listen means that a TCP socket is ready and listening for incoming connections. it's seen on the server side only.
2. SYN_SENT : This means that a synchronization request has been sent, but the connection hasn't been established yet. it's seen this on the client side only.
3. SYN_RECEIVED : This means that a socket previously in a listener state, has received a synchronization request and sent a SYN_ACK back. But it hasn't received the final ACK from the client yet. it's seen on the server side only.
4. ESTABLISHED : This means that the TCP connection is in working order, and both sides are free to send each other data. it's seen on both the client and server sides of the connection.
5. CLOSE_WAIT : This means that the connection has been closed at the TCP layer, but that the application that opened the socket hasn't released its hold on the socket yet.
6. CLOSED : This means that the connection has been fully terminated, and that no further communication is possible.

There are other TCP socket states that exist. Additionally, socket states and their names, can vary from operating system to operating system. That's because they exist outside of the scope of the definition of TCP itself. TCP, as a protocol, is universal in how it's used since every device speaking to TCP protocol has to do this in the exact same way for communications to be successful. Choosing how to describe the states of a socket at the operating system level isn't quite as universal. When troubleshooting issues at the TCP layer, make sure you check out the exact socket state definitions for the systems you're working with.

3.2.4 Connection-oriented and Connectionless Protocols

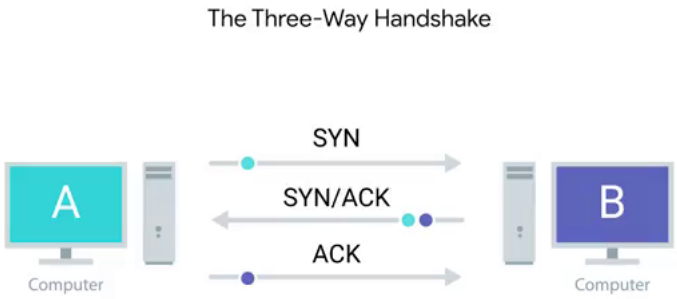
So far, we have mostly focused on TCP which is a connection-oriented protocol.

Connection Oriented Protocol

A connection-oriented protocol is one that establishes a connection, and uses this to ensure that all data has been properly transmitted.

A connection at the transport layer implies that every segment of data sent is acknowledged. This way, both ends of the connection always know which bits of data have definitely been delivered to the other side and which haven't. Connection-oriented protocols are important because the Internet is a vast and busy place, and lots of things could go wrong while trying to get data from point A to point B. You might remember from our lesson about the physical air that even some minor crosstalk from a neighboring twisted pair in the same cable can be enough to make a cyclical redundancy check fail. This could cause the entire frame to be discarded. . If even a single bit doesn't get transmitted properly, the resulting data is often incomprehensible by the receiving end. And remember that at the lowest level, a bit is just an electrical signal within a certain voltage range. But there are plenty of other reasons why traffic might not reach its destination beyond liners. It could be anything. Pure congestion might cause a router to drop your traffic in favour of forwarding more important traffic, or a construction company could cut a fiber cable connecting two ISPs.

Connection-oriented protocols like TCP protect against this by forming connections and through the constant stream of acknowledgements. Our protocols at lower levels of our network model like IP and Ethernet do use checksums to ensure that all the data they received was correct. But did you notice that we never discussed any attempts to resending data that doesn't pass this check. That's because that's entirely up to the transport layer protocol. At the IP or Ethernet level, if a checksum doesn't compute, all of that data is just discarded. It's up to TCP to determine when to resend this data since TCP expects an ACK for every bit of data it sends, it's in the best position to know what data successfully got delivered and it can make the decision to resend a segment if needed.



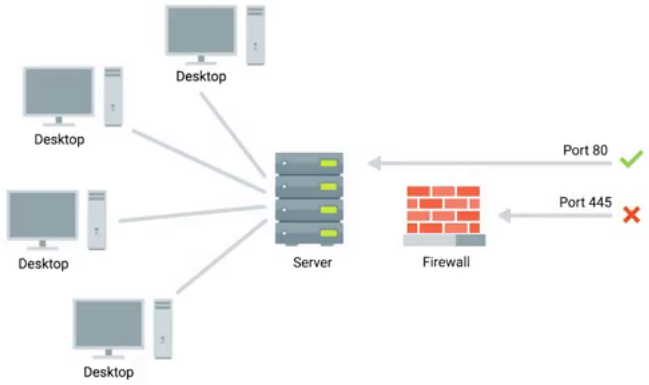
This is another reason why sequence numbers are so important. While TCP will generally send all segments in sequential order, they may not always arrive in that order. If some of the segments had to be resend due to errors at lower layers, it doesn't matter if they arrive slightly out of order. This is because sequence numbers allow for all of the data to be put back together in the right order. Now, as you might have picked up on, there's a lot of overhead with connection-oriented protocols like TCP such as to establish the connection, to send a stream of constant streams of acknowledgements, to tear the connection down at the end. That all accounts for a lot of extra traffic. While this is important traffic, it's really only useful if you absolutely positively have to be sure your data reaches its destination.

You can contrast this with connectionless protocols. The most common of these is known as UDP, or User Datagram Protocol. Unlike TCP, UDP doesn't rely on connections and it doesn't even support the concept of an acknowledgement. With UDP, you just set a destination port and send the packet. This is useful for messages that aren't super important. A great example of UDP is streaming video. Let's imagine that each UDP datagram is a single frame of a video. For the best viewing experience, you might hope that every single frame makes it to the viewer but it doesn't really matter if a few get lost along the way. A video will still be pretty watchable unless it's missing a lot of its frames. By getting rid of all the overhead of TCP, you might actually be able to send higher quality video

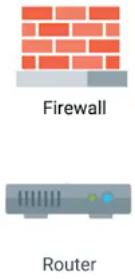
with UDP. That's because you'll be saving more of the available bandwidth for actual data transfer instead of the overhead of establishing connections and acknowledging delivered data segments.

3.2.5 Firewalls

A firewall is just a device that blocks traffic that meets certain criteria. Firewalls are a critical concept to keeping a network secure since they are the primary way you can stop traffic you don't want from entering a network. Firewalls can actually operate at lots of different layers of the network. There are firewalls that can perform inspection of application layer traffic, and firewalls that primarily deal with blocking ranges of IP addresses. The reason we cover firewalls here is that they're most commonly used at the transportation layer. Firewalls that operate at the transportation layer will generally have a configuration that enables them to block traffic to certain ports while allowing traffic to other ports.



Let's imagine a simple small business network. The small business might have one server which hosts multiple network services. This server might have a web server that hosts the company's website, while also serving as the file server for a confidential internal document A firewall placed at the perimeter of the network could be configured to allow anyone to send traffic to port 80 in order to view the web page. At the same time, it could block all access for external IPs to any other port. So that no one outside of the local area network could access the file server. Firewalls are sometimes independent network devices, but it's really better to think of them as a program that can run anywhere.



For many companies and almost all home users, the functionality of a router and a firewall is performed by the same device. And firewalls can run on individual hosts instead of being a network device. All major modern operating systems have firewall functionality built-in. That way, blocking or allowing traffic to various ports and therefore to specific services can be performed at the host level as well.

3.3 The Application Layer

We're almost done covering all aspects of our networking module, which means you've already learned how computers process electrical or optical signals to send communication across a cable, at the physical layer. We've also covered how individual computers can address each other and send each other data using ethernet at the data link layer. We've discussed how the network layer is used by computers and routers to communicate between different networks using IP. And in our last section, we covered how the transportation layer ensures that data is received and sent by the proper applications. Now, we can finally describe about how those actual applications send and receive data using the application layer.

#	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc..	Messages	n/a
4	Transport	TCP/UDP	Segment	Port #'s
3	Network	IP	Datagram	IP address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

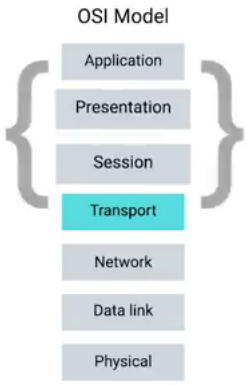
Just like with every other layer, TCP segments have a generic data section to them. As you might have guessed, this payload section is actually the entire contents of whatever data applications want to send to each other. It could be contents of a web page, if a web browser is connecting to a web server. This could be the streaming video content of your Netflix app on your PlayStation connecting with the Netflix servers. It could be the contents of a document your word processor is sending to a printer. And many more things.

There are a lot of protocols used at the application layer, and they are numerous and diverse. At the data link layer, the most common protocol is ethernet. Wireless technologies do use other protocols at this layer, which we'll cover in a future module. At the network layer, use of IP is everywhere you look. At the transport layer, TCP and UDP cover most of the use cases. But at the application layer, there are just so many different protocols in use.

Even so, one concept you can take away about application layer protocols is that they're still standardized across application types. Let's dive a little deeper into web servers and web browsers for an example. There are lots of different web browsers. You could be using Chrome, Internet Explorer, Safari, etc. They'll need to speak the protocol. The same thing is true for web servers. In this case, the web browser would be the client, and the web server would be the server. The most popular web servers are Microsoft IIS, Apache, and nginx. But they also need to speak the same protocol. This way, you ensure that no matter which browser you're using, you'd still be able to be speak to any server. For web traffic, the application layer protocol is known as HTTP. All of these different web browsers and web servers have to communicate using the same HTTP protocol specifications in order to ensure interoperability. The same is true for most other classes of application. You might have dozens of choices for an FTP client, but they all need to speak the FTP protocol in the same way.

3.3.1 The Application Layer and the OSI Model

In our opening module, we talked about how there are lots of competing network layer models. We've been working from a five-layer model, but you'll probably run into various other models Some models might combine the physical and data link layers into one and only talk about four layers. Then, there is the OSI or Open Systems Interconnection model. This model is important to understand with our five-layer model because it's the most rigorously defined. That means it's often used in academic settings or by various network certification organizations. The OSI model has seven layers, and introduces two additional layers between our transport layer and our application layer.



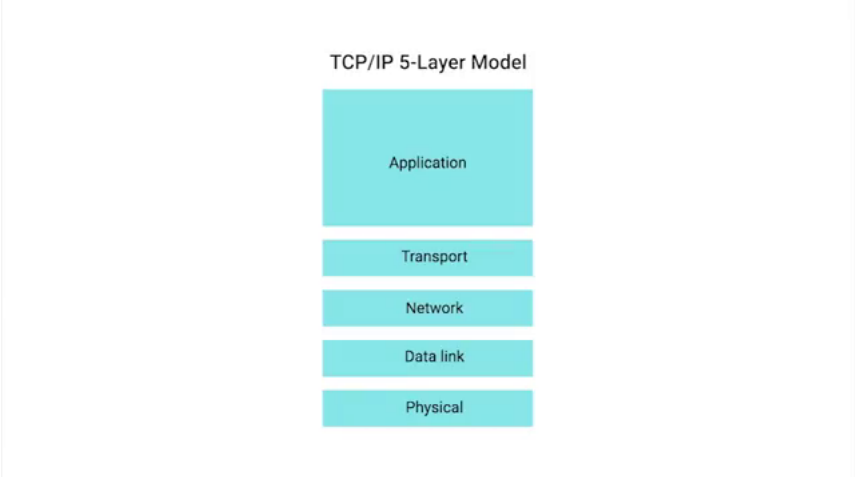
The fifth layer in the OSI model is the session layer.

Session Layer
The concept of a session layer is that it's responsible for things like facilitating the communication between actual applications and the transport layer.

It's the part of the operating system that takes the application layer data that's been unencapsulated from all the layers below it, and hands it off to the next layer in the OSI model, the presentation layer.

Presentation Layer
The presentation layer is responsible for making sure that the unencapsulated application layer data is actually able to be understood by the application in question.

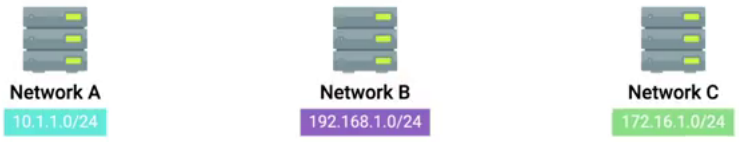
This is the part of an operating system that might handle encryption or compression of data. While these are important concepts to keep in mind, you'll notice that there isn't any encapsulation going on.



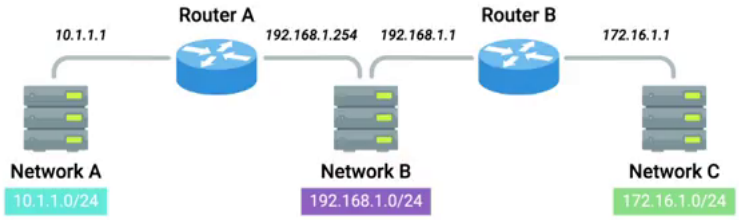
That's why in our model we lump all of these functions into the application layer. We believe a five-layer model is the most useful when it comes to the day-to-day business of understanding networking, but the seven-layer OSI model is also prevalent.

3.4 All Layers Working in Unison

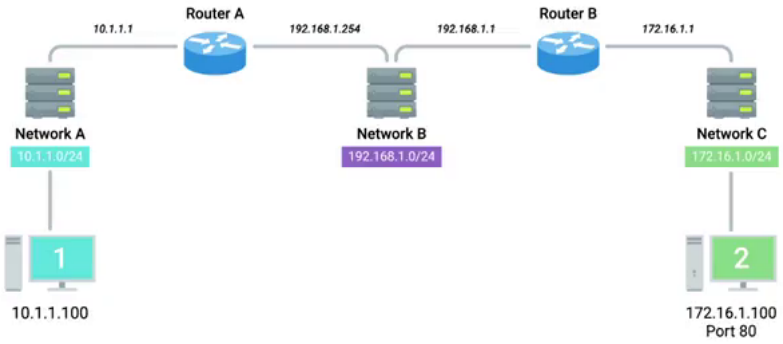
Now that you know the basics of how every layer of out network model works, let's go through an exercise to look at how everything works at every step of the way. Imagine three networks, network A will contain address space 10.1.1.0/24. Network B Will contain address space 192.168.1.0/24, and network C will be 172.16.1.0/24.



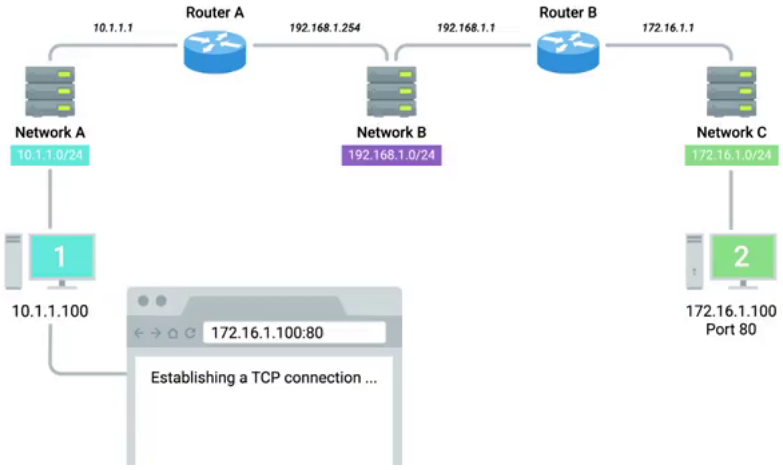
Router A sits between network A and network B. With an interface configured with an IP of 10.1.1.1 on network A, and an interface at 192.168.1.254 on network B. There's a second router, router B, which connects networks B and C. It has an interface on network B with an IP address of 192.168.1.1, and an interface on network C with an IP address of 172.16.1.1.



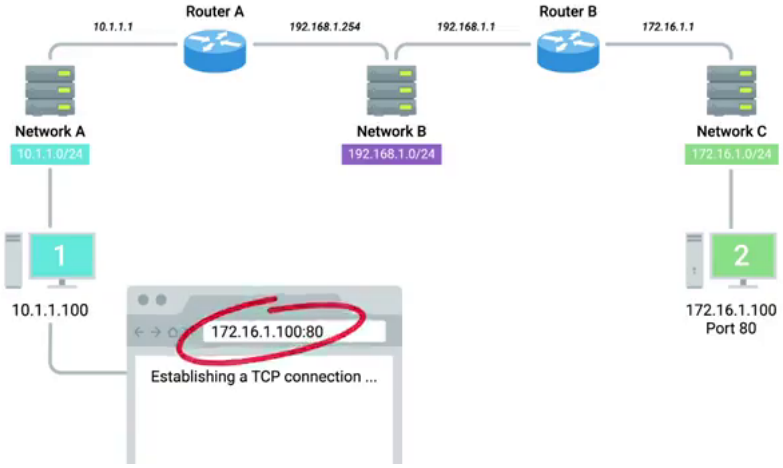
Now let's put a computer on one of the networks. Imagine it's a desktop, sitting on someone's desk at the workplace. It'll be our client in this scenario, and we'll refer to it as computer 1. It's part of Network A and has been assigned an IP address of 10.1.1.100. Now, let's put another computer on one of our other networks. This one is a server in a data center, it'll act as our server in this scenario, and we'll refer to it as computer 2. It's part of network C, and has been assigned an IP address of 172.16.1.100, and has a web server listening on port 80.



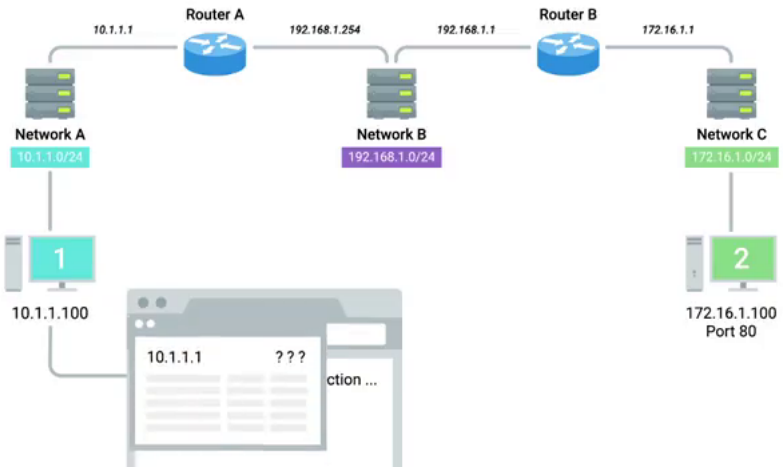
An end user sitting at computer 1 opens up a web browser and enters 172.16.1.100 into the address bar, let's see what happens. The web browser running on computer 1 knows it's been ordered to retrieve a web page from 172.16.1.100. The web browser communicates with the local networking stack, which is the part of the operating system responsible for handling networking functions.



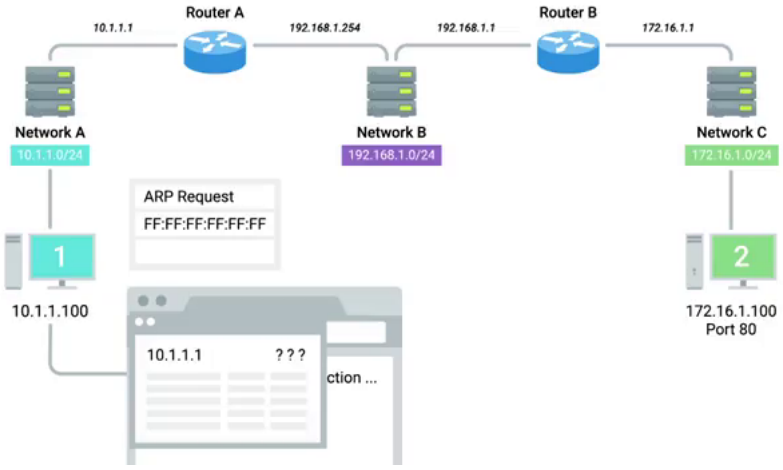
The web browser explains that it's going to want to establish a TCP connection to 172.16.1.100, port 80. The networking stack will now examine its own subnet. It sees that it lives on the network 10.1.1.0/24, which means that the destination 172.16.1.100 is on another network.



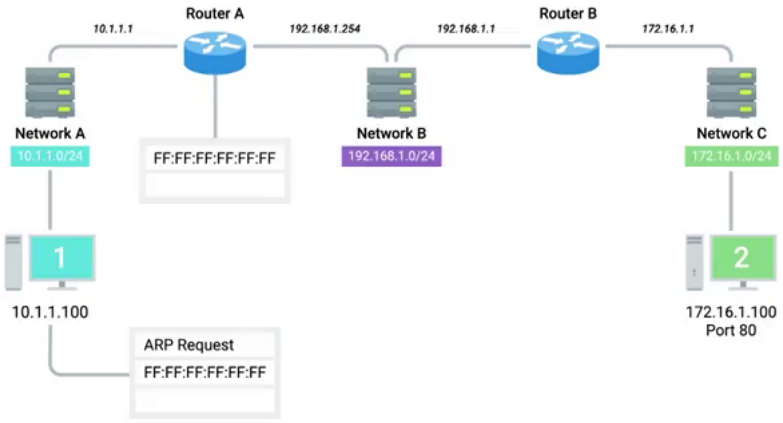
At this point, computer 1 knows that it'll have to send any data to its gateway for routing to a remote network. And it's been configured with a gateway of 10.1.1.1. Next, computer 1 looks at its ARP table to determine what MAC address of 10.1.1.1 is, but it doesn't find any corresponding entry.



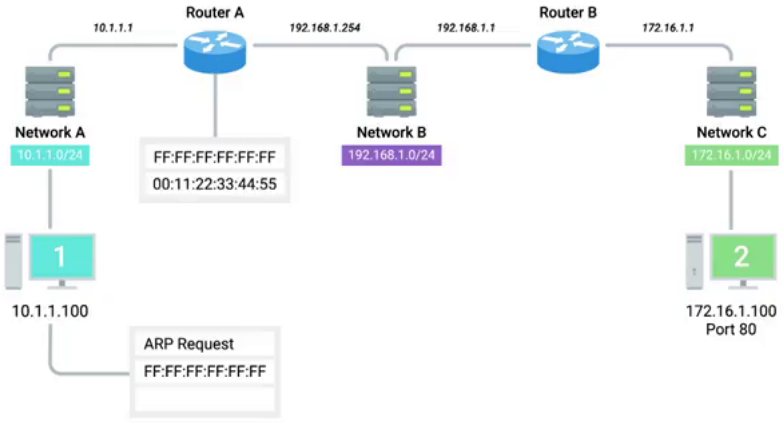
So computer A crafts an ARP request for an IP address of 10.1.1.1, which it sends to the hardware broadcast address of all Fs.



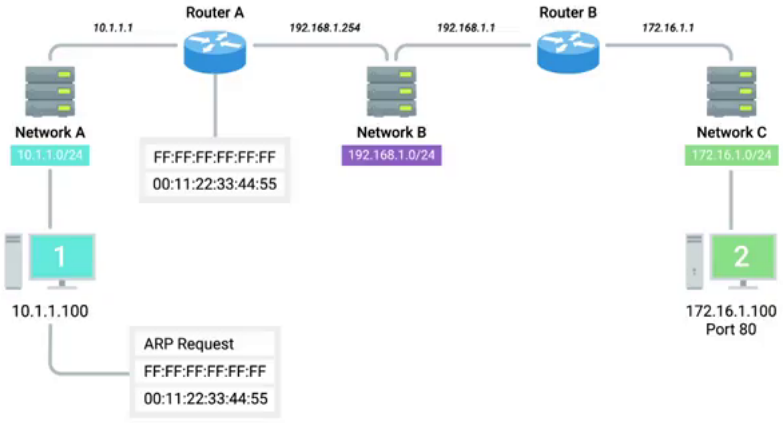
This ARP discovery request is sent to every node on the local network.



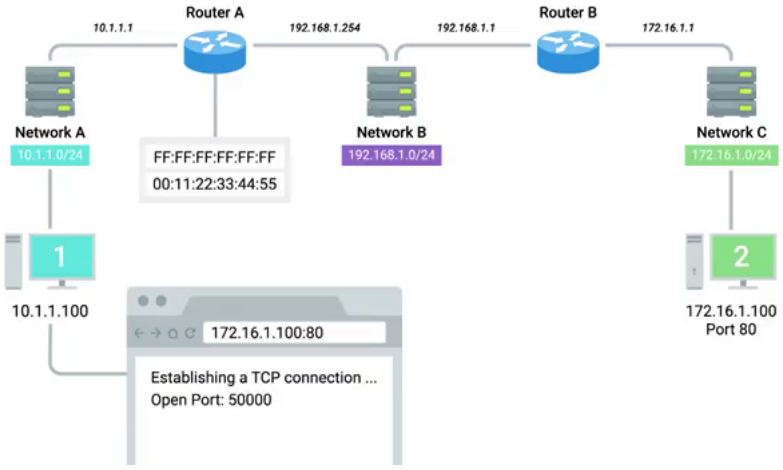
When router A receives this ARP message, it sees that it's the computer currently assigned the IP address of 10.1.1.1.



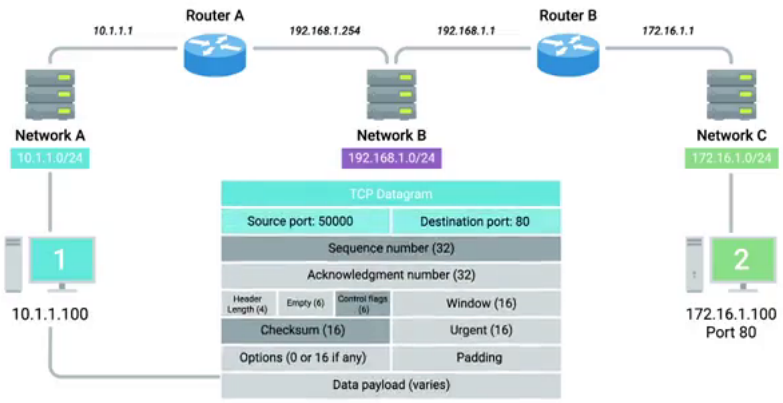
So it responds to computer 1 to let it know about its own MAC address of 00:11:22:33:44:55. Computer 1 receives this response and now knows the hardware address of its gateway.



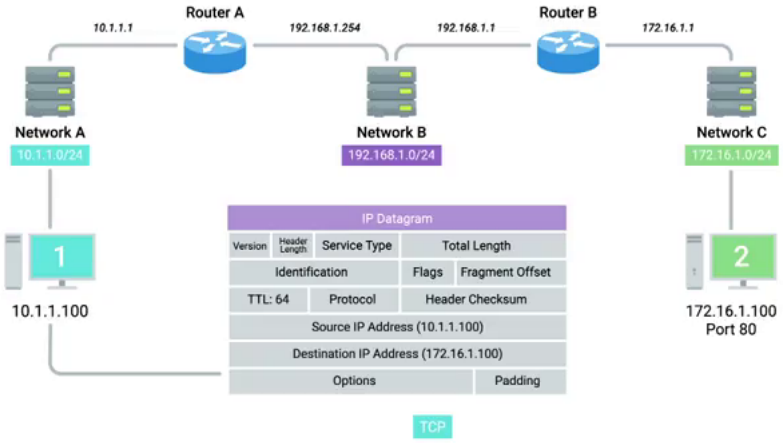
This means that it's ready to start constructing the outbound packet. Computer 1 knows that it's being asked by the web browser to form an outbound TCP connection, which means it'll need an outbound TCP port. The operating system identifies the ephemeral port of 50000 as being available, and opens a socket connecting the web browser to this port.



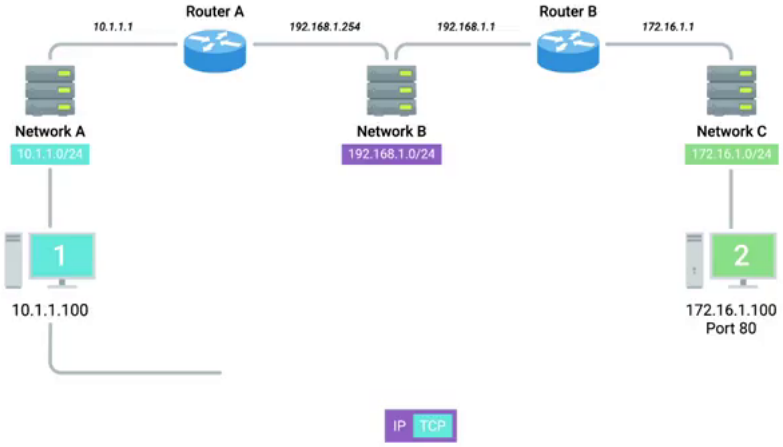
Since this is a TCP connection, the networking stack knows that before it can actually transmit any of the data the web browser wants it to, it'll need to establish a connection. The networking stack starts to build a TCP segment. It fills in all the appropriate fields in the header, including a source port of 50000 and a destination port of 80. A sequence number is chosen and is used to fill in the sequence number field. Finally, the SYN flag is set, and a checksum for the segment is calculated and written to the checksum field.



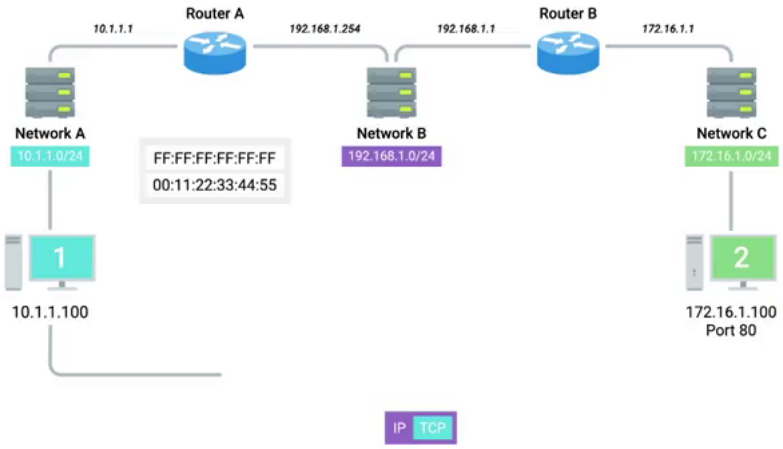
Our newly constructed TCP segment is now passed along to the IP layer of the networking stack. This layer constructs an IP header. This header is filled in with the source IP, the destination IP, and a TTL of 64, which is a pretty standard value for this field.



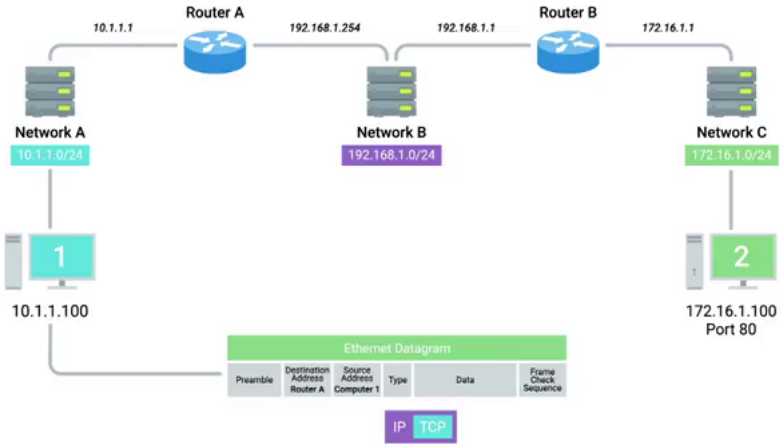
Next, the TCP segment is inserted as the data payload for the IP datagram. And a checksum is calculated for the whole thing.



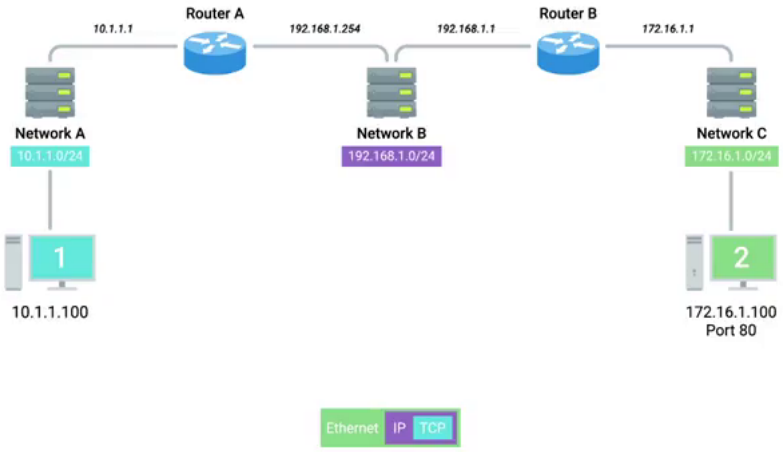
Now that the IP datagram has been constructed, computer 1 needs to get this to its gateway, which it now knows has a MAC address of 00:11:22:33:44:55.



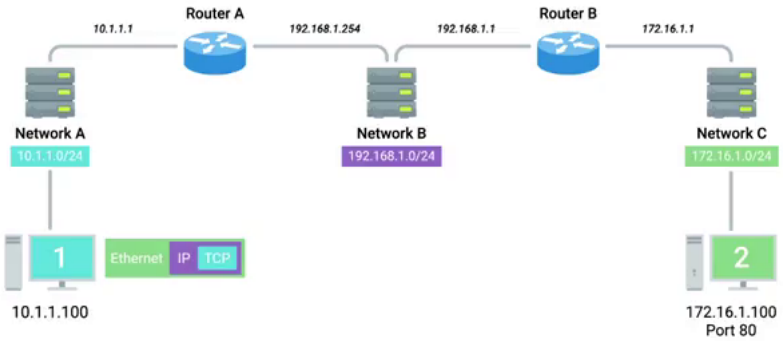
So an Ethernet Datagram is constructed. All the relevant fields are filled in with the appropriate data, most notably, the source and destination MAC addresses.



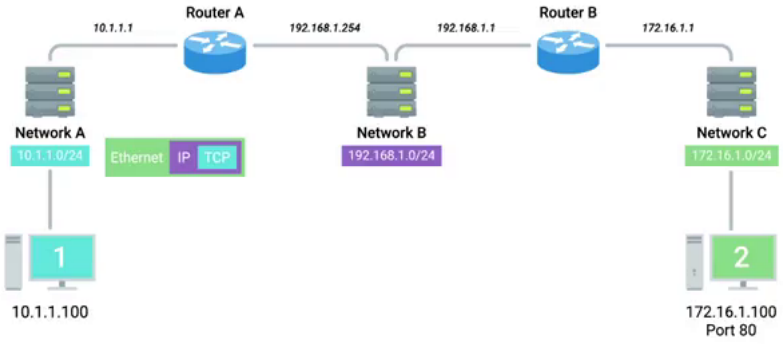
Finally, the IP datagram is inserted as the data payload of the Ethernet frame, and another checksum is calculated.



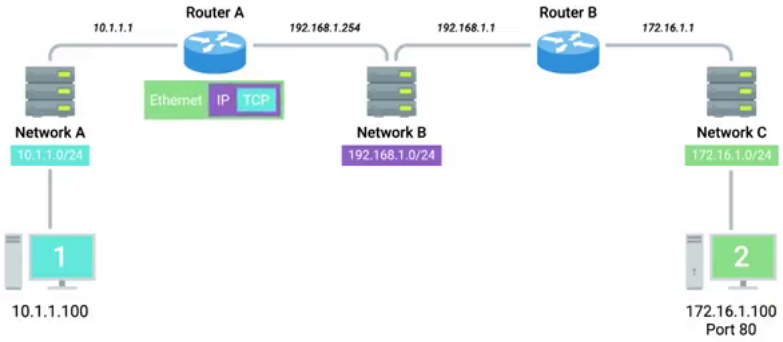
Now we have an entire Ethernet frame ready to be sent across the physical layer.



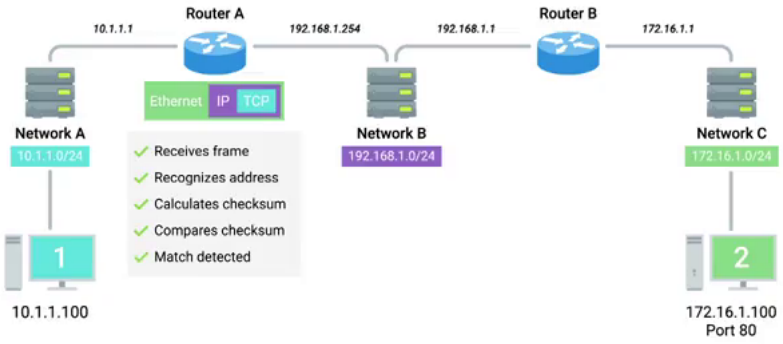
The network interface connected to computer 1 sends this binary data as modulations of the voltage of an electrical current running across a CAT6 cable that's connected between it and a network switch. This switch receives the frame and inspects the destination MAC address.



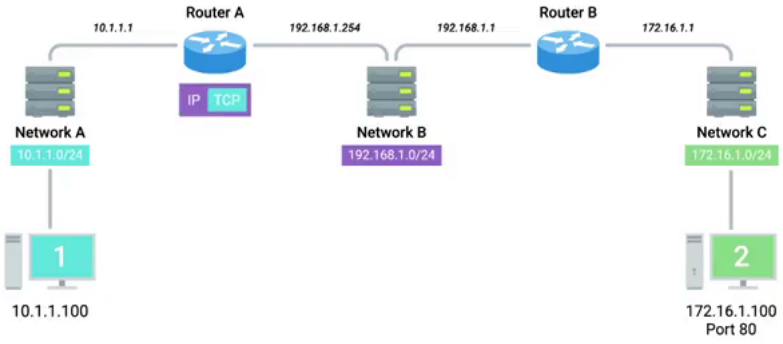
The switch knows which of its interfaces this MAC address is attached to, and forwards the frame across only the cable connected to this interface.



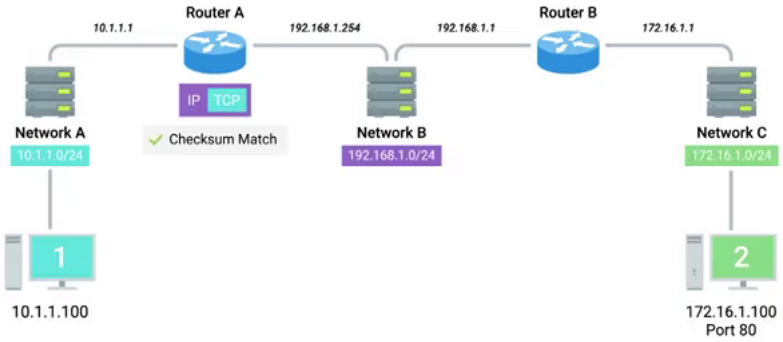
At the other end of this link is router A, which receives the frame and recognizes its own hardware address as the destination. Router A knows that this frame is intended for itself. So it now takes the entirety of the frame and calculates a checksum against it. Router A compares this checksum with the one in the Ethernet frame header and sees that they match. Meaning that all of the data has made it in one piece.



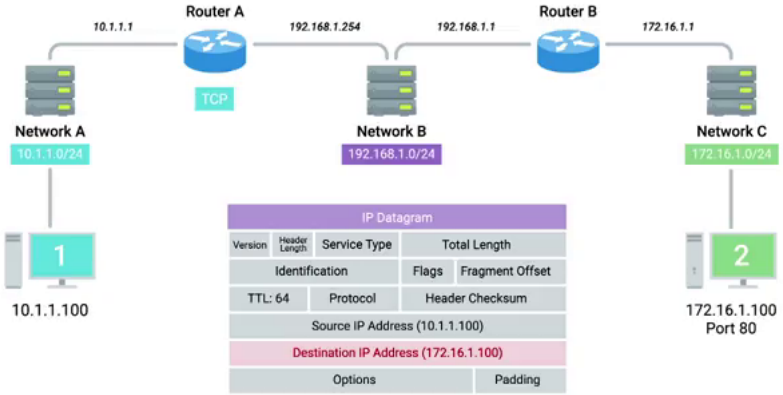
Next, Router A strips away the Ethernet frame, leaving it with just the IP datagram.



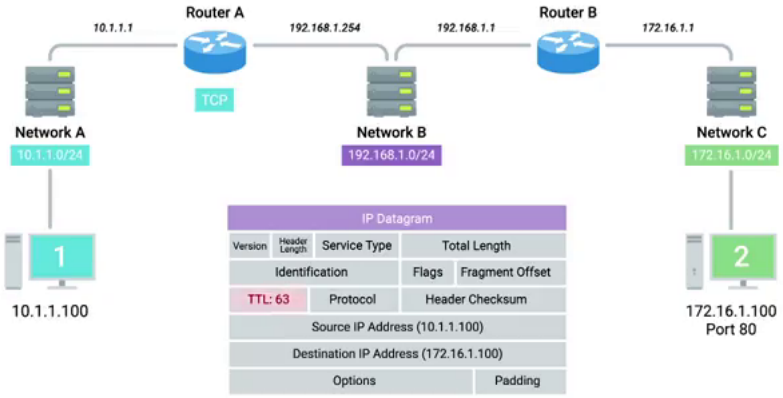
Again, it performs a checksum calculation against the entire datagram. And again, it finds that it matches, meaning all the data is correct.



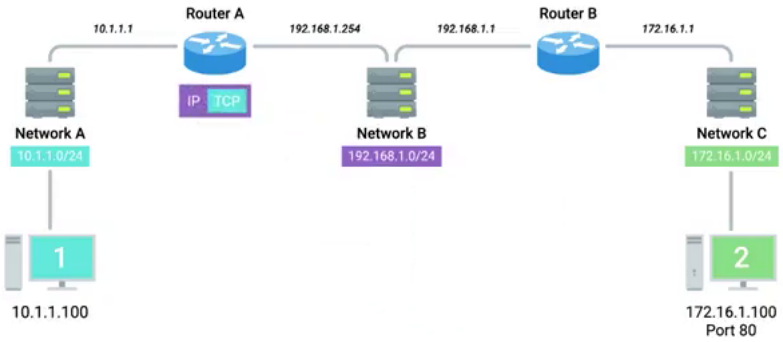
It inspects the destination IP address and performs a lookup of this destination in its routing table.



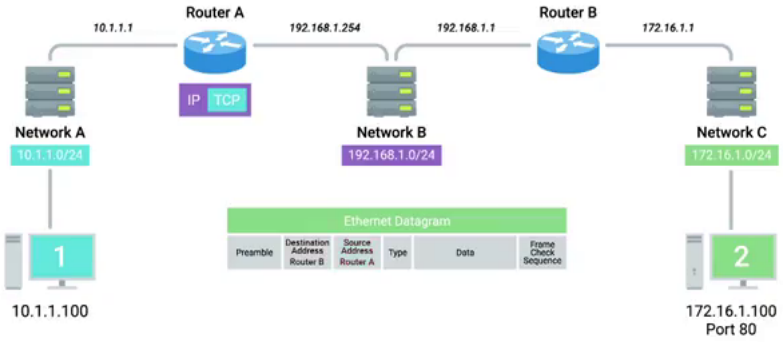
Router A sees that in order to get data to the 172.16.1.0/24 network, the quickest path is one hop away via Router B, which has an IP of 192.168.1.1. Router A looks at all the data in the IP datagram, decrements the TTL by 1, calculates a new checksum reflecting that new TTL value,



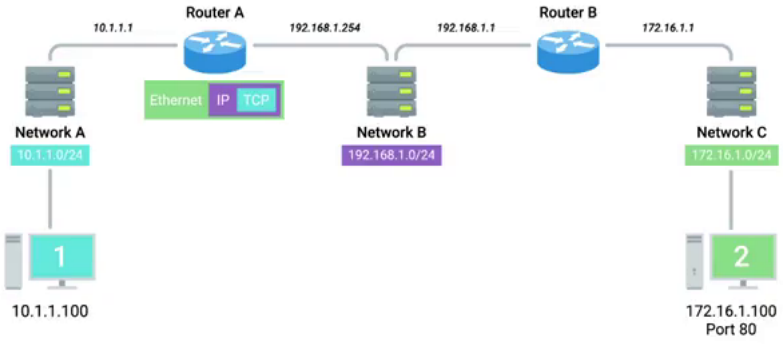
Then, It makes a new IP datagram with this data.



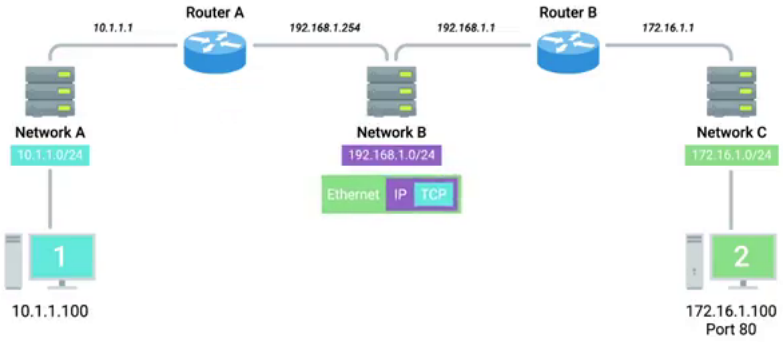
Router B knows that it needs to get this datagram to router B, which has an IP address of 192.168.1.1. It looks at its ARP table, and sees that it has an entry for 192.168.1.1. Now router A can begin to construct an Ethernet frame with the MAC address of its interface on network B as the source . And the MAC address on router B's interface on network B as the destination.



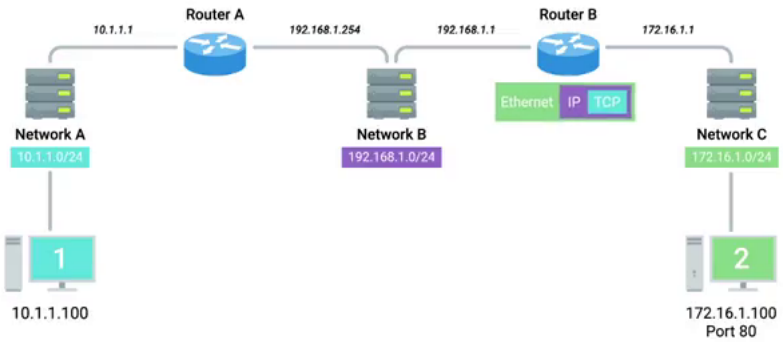
Once the values for all fields in this frame have been filled out, router A places the newly constructed IP datagram into the data payload field. Calculates a checksum, and places this checksum into place.



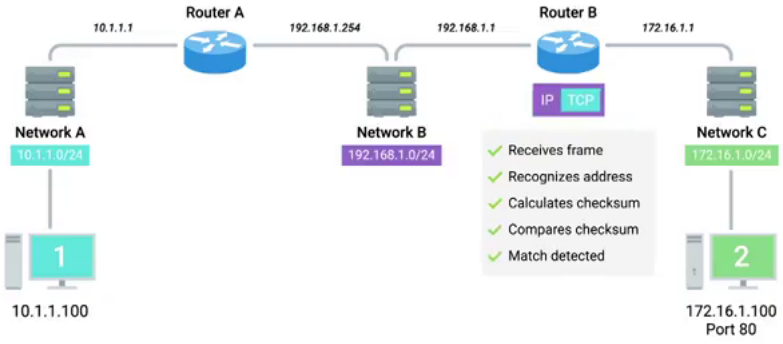
And sends the frame out to network B.



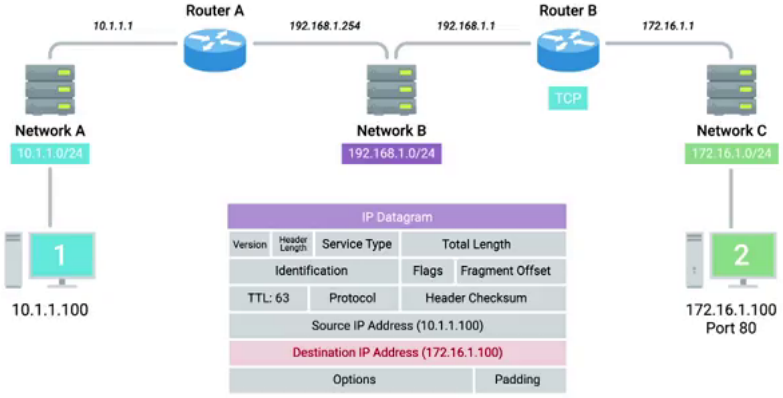
Just like before, this frame makes it across network B, and is received by router B.



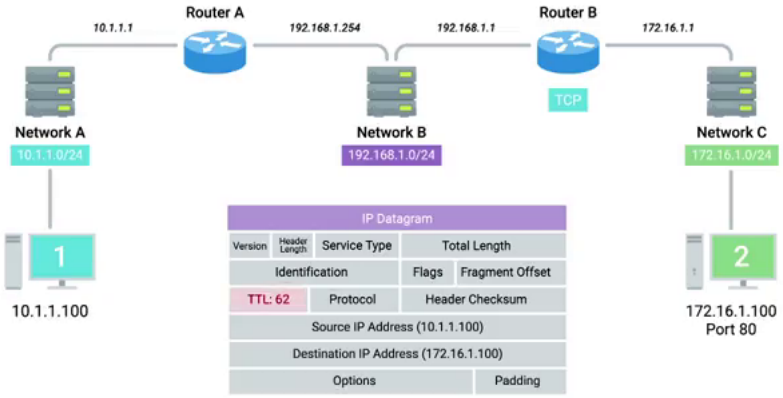
Router B performs all the same checks, removes the the Ethernet frame encapsulation, and performs a checksum against the IP datagram.



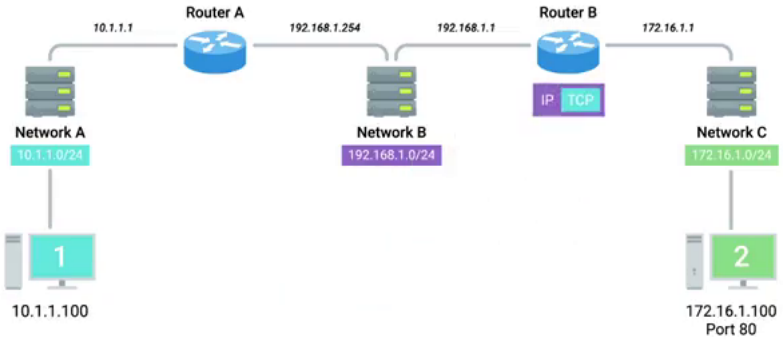
It then examines the destination IP address. Looking at its routing table, router B sees that the destination address of computer 2, or 172.16.1.100, is on a locally connected network.



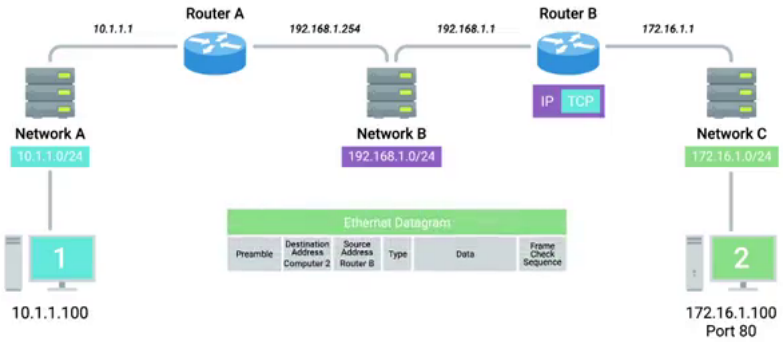
So it decrements the TTL by 1 again and calculates a new checksum.



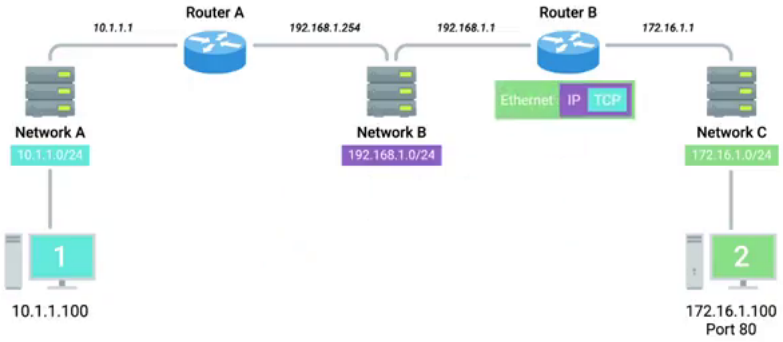
Creates a new IP datagram.



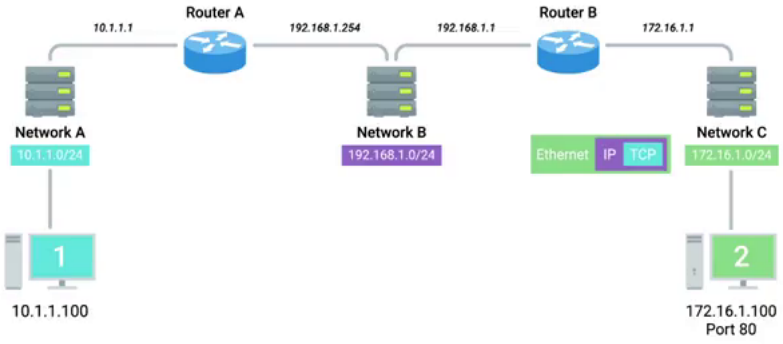
This new IP datagram is again encapsulated by a new Ethernet frame. This one with the source and destination MAC address of router B and computer 2.



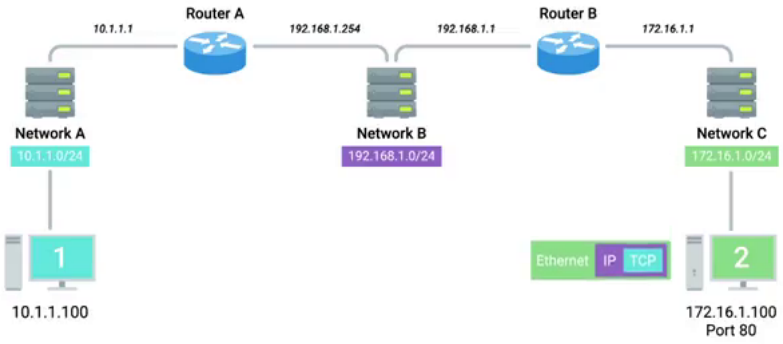
And the whole process is repeated one last time.



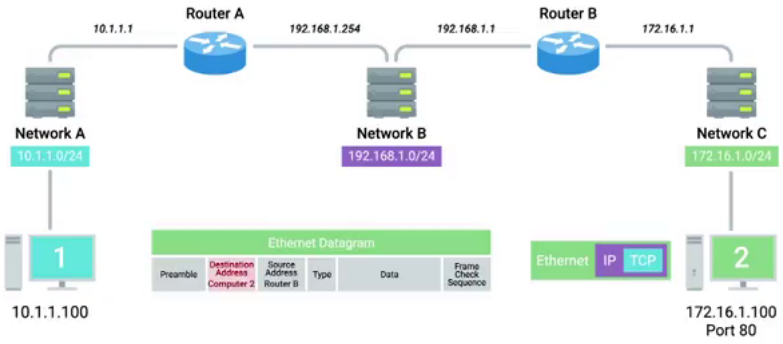
The frame is sent out onto network C, a switch ensures it gets sent out of the interface that computer 2 is connected to.



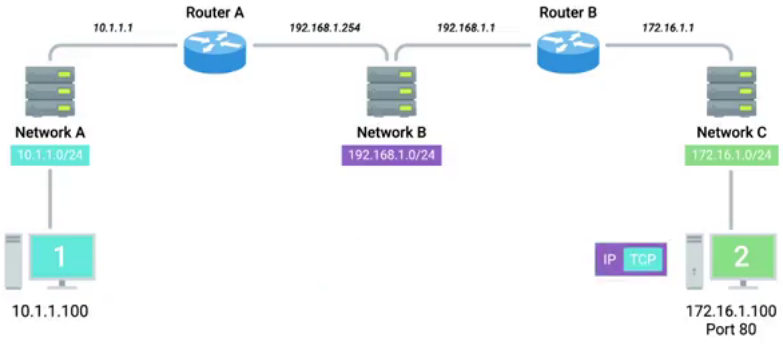
Computer 2 receives the frame,



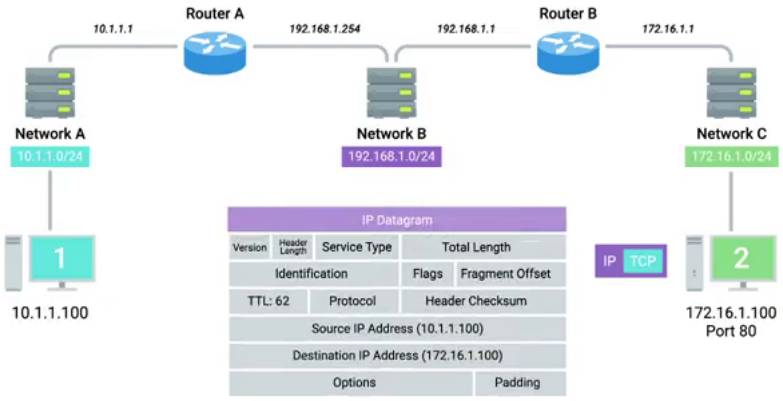
It then identifies its own MAC address as the destination, and knows that it's intended for itself from Ethernet Frame.



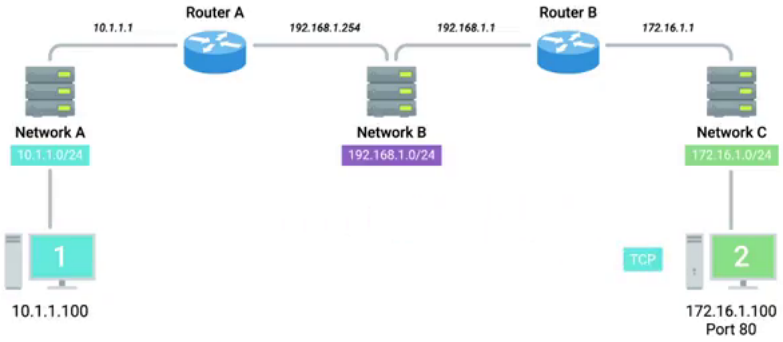
Computer 2 then strips away the Ethernet frame, leaving it with the IP datagram.



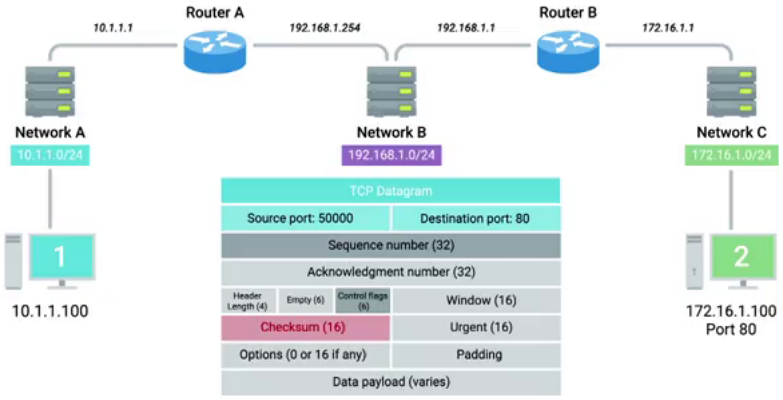
It performs a CRC and recognizes that the data has been delivered intact. It then examines the destination IP address and recognizes that as its own.



Next, computer 2 strips away the IP datagram, leaving it with just the TCP segment.



Again, the checksum for this layer is examined, and everything checks out.



Next, computer 2 examines the destination port, which is 80. The networking stack on computer 2 checks to ensure that there's an open socket on port 80, which there is. It's in the listen state, and held open by a running Apache web server.

Computer 2 then sees that this packet has the SYN flag set. So it examines the sequence number and stores that, since it'll need to put that sequence number in the acknowledgement field once it crafts the response. After all of that, all we've done is get a single TCP segment containing a SYN flag from one computer to a second one. Everything would have to happen all over again for computer 2 to send a SYN-ACK response to computer 1. Then everything would have to happen all over again for computer 1 to send an ACK back to computer 2, and so on and so on. Looking at all of this end to end hopefully helps show how all the different layers of our networking model have to work together to get the job done. it also gives you some perspective in understanding how remarkable computer networking truly is.

Chapter 4

Networking Services

Outcome

1. You'll be able to describe why name resolution is important, identify the many steps involved with the DNS lookup, and understand the most common DNS record types.
2. You'll also be able to explain DHCP makes network administration a simpler task.
3. You'll be able to demonstrate how NAT technologies help keep networks secure and help preserve precious IP address space.
4. You'll be able to describe how VPNs and proxies help users get connected and stay secure.

4.1 Intro to Networking Services

Computer networking is a complicated business that involves many technologies, layers, and protocols. The main purpose of computer networking is so network services can be available to answer requests for the data from clients. The sheer number and variety of things that might comprise a network service makes it impossible to cover all of them. But there are a lot of network services and technologies that are used to help make computer networking more user friendly and secure. These network services and technologies, are ones that directly relate to the business of networking itself, and it's important to understand how those work.
If something on the network isn't working as expected, the first place you should look at are the services.

4.2 Name Resolution

4.2.1 Why do we need DNS

Computers speak to each other in numbers. At the very lowest levels, all computers really understand are 1 and 0. Reading binary numbers isn't the easiest for humans, so most binary numbers are represented in lots of different forms. This is especially true in the realm of networking. Remember that an IP address is really just a 32-bit binary number, but it's normally written out as 4 octets in decimal form since that's easier for humans to read.

You might also remember that MAC addresses are just 48-bit binary numbers that are normally written out in 6 groupings of 2 hexadecimal digits each. While remembering 192.168.1.100 might be easier than remembering a long string of 1s and 0s, It still doesn't do a very good job when you have to remember more than just a few addresses. Imagine having to remember the four octets of an IP address for every website you visit. It's just not a thing that the human brain is normally good at. Humans are much better at remembering words. That's where DNS, or domain name system, comes into play.

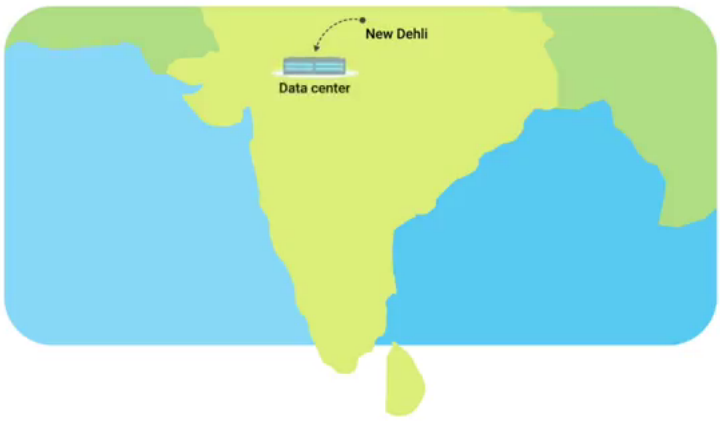
DNS
DNS is a global and highly distributed network service that resolves strings of letters into IP address for you.

Let's say you wanted to check a weather website to see what the temperature is going to be like. It's much easier to type www.weather.com into a web browser than it is to remember that one of the IP adresses for this site is 184.29.131.121.
The IP address for a domain name can also change all the time for a lot of different reasons. A domain name is just the term we use for something that can be resolved by DNS. In the example we just used, www.weather.com would be the domain name, and the IP it resolves to could change, depending on a variety of factors. Let's say that weather.com was moving their web server to a new data center. By using DNS, an organization can just change what IP a domain name resolves to, and the end user would never even know. So, not only does DNS make it easier for humans to remember how to get to a website, It also lets administrative changes happen behind the scenes without an end user having to change their behavior.

Try to imagine a world where you'd have to remember every IP for every website you visit, while also having to memorize new ones if something changed. We'd spend our whole day memorizing numbers. The importance of DNS for how the Internet operates, today, can't be overstated.

IP addresses might resolve to different things depending on where in the world you are. While most Internet communications travel at the speed of light, the further you have to route data, the slower things will become. In almost all situations, it's going to be quicker to transmit a certain amount of data between places that are geographically close to each other. If you're a global web company, you'd want people from all over the world to have a great experience accessing your website. So instead of keeping all of your web servers in one place, you could distribute them across data centers across the globe.
This way, someone in New York, visiting a website, might get served by a web server close to New York,

While someone in New Delhi might get served by a web server closer to New Delhi.



Again, DNS helps provide this functionality. Because of its global structure, DNS lets organizations decide, if you're in the region, resolve the domain name to this IP. If you're in this other region, resolve this domain to this other IP.

4.2.2 The Many Steps of Name Resolution

At its most basic, DNS is a system that converts domain names into IP addresses. It's the way humans are likely to remember and categorize things resolved into the way computers prefer to think of things. This process of using DNS to turn a domain name into an IP address is known as name resolution.
Let's take a closer look at exactly how this works. The first thing that's important to know is that DNS servers, are one of the things that need to be specifically configured at a node on a network. For a computer to operate on a modern network, they need to have certain number of things configured. Remember, that MAC addresses are hard coded and tied to specific pieces of hardware.
But we've also covered that the **IP address, subnet mask, and gateway** for a host must be specifically configured, a **DNS server**, is the fourth and final part of the standard modern network configuration. These are almost always the four things that must be configured for a host to operate on a network in an expected way. It should call out, that a computer can operate just fine without DNS or without a DNS server being configured, this makes things difficult for any human that might be using that computer.

There are five primary types of DNS servers:

1. Caching name servers
2. Recursive name servers

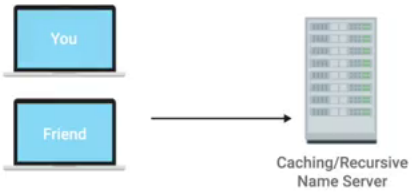
3. Root name servers
4. TLD name servers
5. Authoritative name servers.

As we dive deeper into these, it's important to note that any given DNS server can fulfill many of these roles at once.

Caching and recursive name servers are generally provided by an ISP or your local network. Their purpose is to store domain name lookups for a certain amount of time.

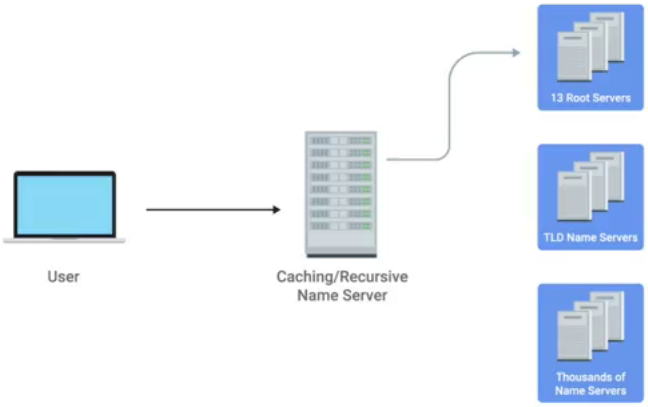
There are lots of steps in order to perform a fully qualified resolution of a domain name. In order to prevent this from happening every single time a new TCP connection is established, your ISP or local network will generally have a caching name server available. Most caching name servers are also recursive name servers. Recursive name servers are ones that perform full DNS resolution requests. In most cases, your local name server will perform the duties of both, but it's definitely possible for a name server to be either just caching or just recursive.

Let's introduce an example to better explain how this works. You and your friend are both connected to the same network and you both want to check out Facebook.com, your friend enters www.facebook.com into a web browser, which means that their computer now needs to know the IP of www.facebook.com in order to establish a connection. Both of your computers are on the same network which usually means, that they both been configured with the same name server. So your friends computer ask the name server for the IP of www.facebook.com which it doesn't know, this name server now performs a fully recursive resolution to discover the correct IP for www.facebook.com.



This IP is then both delivered to your friend's computer and stored locally in a cache. A few minutes later you enter www.facebook.com into a web browser. Again, your computer needs to know the IP for this domain, so your computer asks the local name server it's been configured with, which is the same one your friend's computer was just talking to. Since the domain name www.Facebook.com had just been looked up, the local name server still has the IP that it resolved to stored and is able to deliver that back to your computer without having to perform a full lookup. This is how the same servers act as a caching server.

All domain names in the global DNS system have a TTL or time to live. This is a value in seconds, that can be configured by the owner of a domain name for how long a name server is allowed to cache in entry before it should discard it and perform a full resolution again. Several years ago, it was normal for these TTL's to be really long, sometimes a full day or more. This is because the general bandwidth available on the Internet was just much less, so network administrators didn't want to waste what bandwidth was available to them by constantly performing full DNS lookups. As the Internet has grown and gone faster, these TTL's for most domains have dropped to anywhere from a few minutes to a few hours. But it's important to know that sometimes you still run into a domain names with very lengthy TTL's, it means that it can take up to the length of a total TTL for a change in DNS record to be known to the entire Internet.



Now, let's look at what happens when your local recursive server needs to perform a full recursive resolution. The first step is always to contact a root named server, there are 13 total root name servers and they're responsible for directing queries toward the appropriate TLD name server. In the past, these 13 root servers were distributed to very specific geographic regions, but today, they're mostly distributed across the globe via anycast. Anycast is a technique that's used to route traffic to different destinations depending on factors like location, congestion, or link health. Using anycast, a computer can send a data gram to a specific IP but could see it routed to one of many different actual destinations depending on a few factors.
This should also make it clear that there aren't really only 13 physical route name servers anymore. It's better to think of them as 13 authorities that provide route name lookups as a service. The root servers will respond to a DNS lookup with the TLD name server that should be queried. TLD stands for top level domain and represents the top of the hierarchical DNS name resolution system. A TLD is the last part of any domain name, using www.facebook.com as an example again, the dot com portion should be thought of as the TLD. For each TLD in existence, there is a TLD name server, but just like with root servers, this doesn't mean there's only physically one server in question, it's most likely a global distribution of any cast accessible servers responsible for each TLD.

The TLD name servers will respond again with a redirect, this time informing the computer performing the name lookup with what authoritative name server to contact. Authoritative name servers are responsible for the last two parts of any domain name which is the resolution at which a single organization may be responsible for DNS lookups. Using www.weather.com as an example, the TLD name server would point a lookup at the authoritative server for Weather.com, which would likely be controlled by the Weather Channel, the organization itself that runs the site. Finally, the DNS lookup could be redirected at the authoritative server for weather.com which would finally provide the actual IP of the server in question. This strict hierarchy is very important to the stability of the internet, making sure that all full DNS resolutions go through a strictly regulated and controlled series of lookups to get the correct responses, is the best way to protect against malicious parties redirecting traffic. Your computer will blindly send traffic to whatever IP it's told to. So by using a hierarchical system controlled by trusted entities in the way DNS does, we can better ensure that the responses to DNS lookups are accurate. Now that you see how many steps are involved, it should make sense why we trust our local name servers to cache DNS lookups, its so that full lookup path doesn't have to happen for every single TCP connection. In fact, your local computer from your phone to a desktop will generally have its own temporary DNS cache as well, that way, it doesn't have to bother its local name server for every TCP connection either.

4.3 DHCP

Managing hosts on a network can be a daunting and time consuming task. Every single computer on a modern TCP/IP based network needs to have at least four things to specifically configured.

1. IP Address
2. Subnet mask for the local network,
3. Primary gateway,
4. Nname server.

These four things don't seem like much, but when you have to configure them on hundreds of machines, it becomes super tedious. Out of these four things, three are likely the same on just about every node on the network, the subnet mask, the primary gateway, and DNS server. But the last item, an IP address, needs to be different on every single node on the network. That could require a lot of tricky configuration work, and this is where DHCP, or Dynamic Host Configuration Protocol, comes into play.

DHCP is an application layer protocol that automates the configuration process of hosts on a network. With DHCP, a machine can query a DHCP server when the computer connects to the network and receive all the network configuration in one go. Not only does DHCP reduce the administrative overhead of having to configure lots of network devices on a single network, it also helps address the problem of having to choose what IP to assign to what machine.

Every computer on a network requires an IP for communications, but very few of them require an IP that would be commonly known. For servers or network equipment on your network, like your gateway router is static and known IP address, it's pretty important. For example, the devices on a network need to know the IP of their gateway at all times. If the local DNS server was malfunctioning, network administrators would still need a way to connect to some of these devices through their IP. Without a static IP configured for a DNS server, it would be hard to connect to it to diagnose any problems if it was malfunctioning. But for a bunch of client devices like desktops, or laptops, or even mobile phones, it's really only important that they have an IP on the right network. It's much less important exactly which IP that is. Using DHCP, you can configure a range of IP addresses that's set aside for these client devices. This ensures that any of these devices can obtain an IP address when they need one, but solves the problem of having to maintain a list of every node on the network and its corresponding IP.

There are a few standard ways that DHCP can operate. DHCP dynamic allocation is the most common, A range of IP addresses is set aside for client devices and one of these IPs is issued to these devices when they request one. Under a dynamic allocation, the IP of a computer could be different, almost every time it connects to the network.

Automatic allocation is very similar to dynamic allocation, in that a range of IP addresses is set aside for assignment purposes. The main difference here is that the DHCP server is asked to keep track of which IPs it's assigned to certain devices in the past. Using this information, the DHCP server will assign the same IP to the same machine each time, if possible.

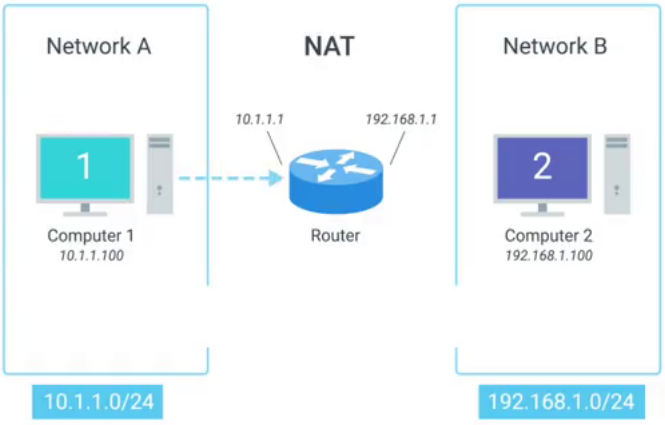
Finally, there's whats known as fixed allocation. Fixed allocation requires a manually specified list of MAC address and the corresponding IPs. When a computer requests an IP, the DHCP server looks for its MAC address in a table, and assigns the IP that corresponds to that MAC address. If the MAC address isn't found, the DHCP server might fall back to automatic or dynamic allocation, or it might refuse to assign an IP altogether. This can be used as a security measure to ensure that only devices that have had their MAC address specifically configured at the DHCP server will ever be able to obtain an IP and communicate on the network. Along with things like IP address and primary gateway, you can also use DHCP to assign things like NTP servers. NTP stands for Network Time Protocol, and is used to keep all computers on a network synchronized in time.

4.4 Network Address Translation

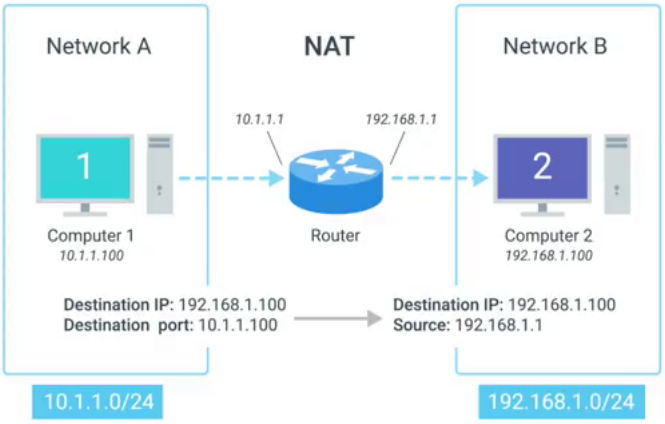
4.4.1 Basics of NAT

So unlike protocols like DNS and DHCP, network address translation or NAT is a technique instead of a defined standard. . Different operating systems and different network hardware vendors have implemented the details of NAT in different ways. But the concepts of what it accomplishes are constant. Network address translation takes one IP address and translates into another. There are lots of reasons why you would want to do this. They range

from security safeguards, to preserving the limited amounts of available IPV4 space.
At its most basic level NAT, is a technology that allows a gateway usually a router or a firewall to rewrite the source IP of an outgoing IP datagram, while retaining the original IP in order to rewrite it into the response, to explain this better, let's look at a simple NAT example.



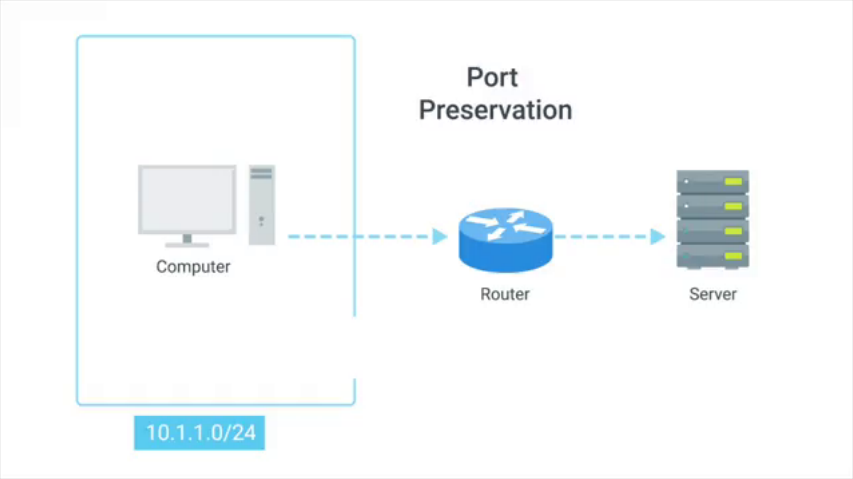
Let's say we have two networks, network A consists of the 10.1 1.0/24 address space, and network B consists of the 192. 1 6 8.1.0/24 address space. Sitting between these networks is a router that has an interface on network A with an IP of 10.1.1.1 and an interface on Network B of 192.168. 1.1 . The two computers on these networks. Computer one is on network A, and has an IP of 10.1.1.100 and computer two is on network B and has an IP of 192.168.1.100. Computer one wants to communicate with a web server on computer two. So it crafts the appropriate packet at all layers and sends this to its primary gateway.



The router sitting between the two networks, so far this is a lot like many of our earlier examples, but in this instance the router is configured to perform NAT for any outbound packets. Normally a router will inspect the contents of an IP datagram, decrement the TTL by one, recalculate the checksum and forward the rest of the data at the network layer without touching it. But with NAT the router will also rewrite the source IP address, which in this instance becomes the routers IP on Network B or 192. 168. 1.1. When the datagram gets to computer two, it will look like it originated from the router not from computer one. Now computer two crafts its response and sends it back to the router.
The router knowing that this traffic is actually intended for computer one, rewrites the destination IP field before forwarding it along. What NAT is doing in this example is hiding the IP of computer one from computer two. This is known as IP masquerading. IP masquerading is an important security concept. The most basic concept at play here, is that no one can establish a connection to your computer if they don't know what IP address it has. By using NAT in the way we've just described, we could actually have hundreds of computers on network A, all of their IPs being translated by the router to its own. To the outside world, the entire address space of network A is protected and invisible. This is known as one to many NAT.

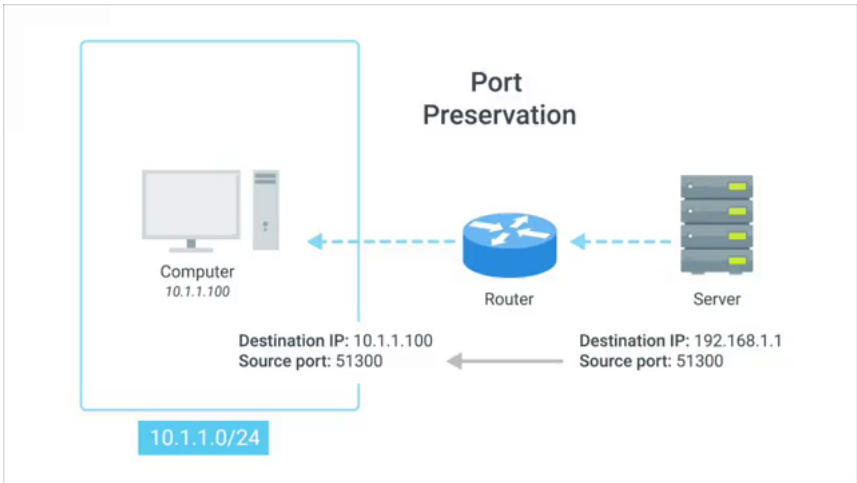
4.4.2 NAT and the Transport Layer

NAT at the network layer is pretty easy to follow. One IP address is translated to another by a device usually a router. But at the transport layer things get a little bit more complicated and several additional techniques come into play to make sure everything works properly. With one to many NAT, we've talked about how hundreds even thousands of computers can all have their outbound traffic translated via NAT to a single IP.
This is pretty easy to understand when the traffic is outbound, but a little more complicated once return traffic is involved. We now have potentially hundreds of responses all directed at the same IP and the router at this IP needs to figure out which responses go to which computer. The simplest way to do this, is through port preservation.

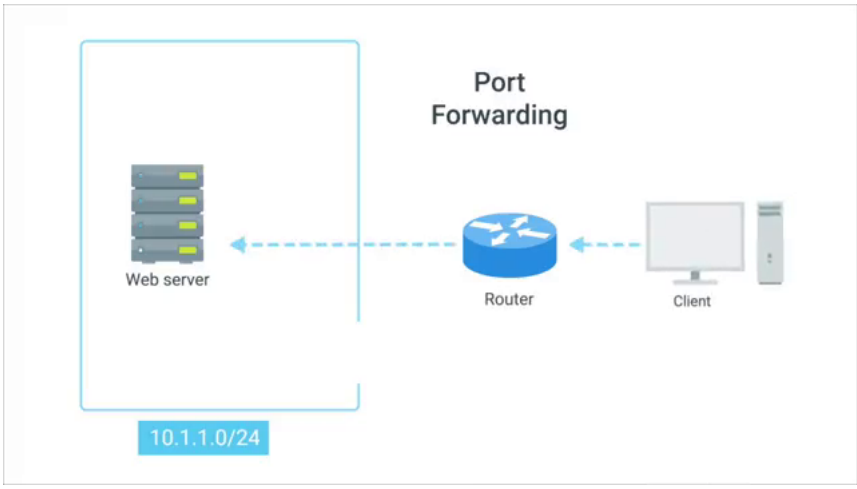


Port preservation is a technique where the source port chosen by a client, is the same port used by the router. Remember that outbound connections choose a source port at random, from the ephemeral ports or the ports in the range 49,152 through 65, 535. In the simplest setup, a router setup to NAT outbound traffic, will just keep track of what this source port is, and use that to direct traffic back to the right computer.

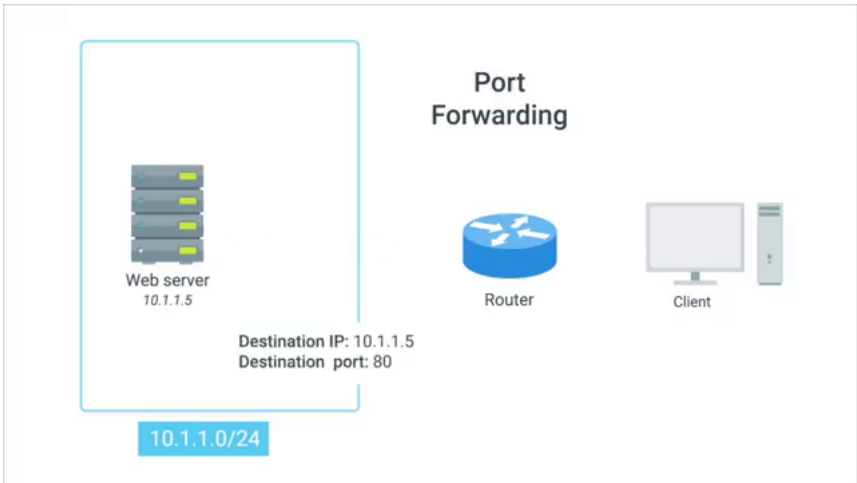
Let's imagine a device with an IP of 10.1.1.100. It wants to establish an outbound connection and the networking stack of the operating system chooses port 51,300 for this connection. Once this outbound connection gets to the router, it performs network address translation and places its own IP in the source address field of the IP datagram, but it leaves the source port in the TCP datagram the same and stores this data internally in a table.



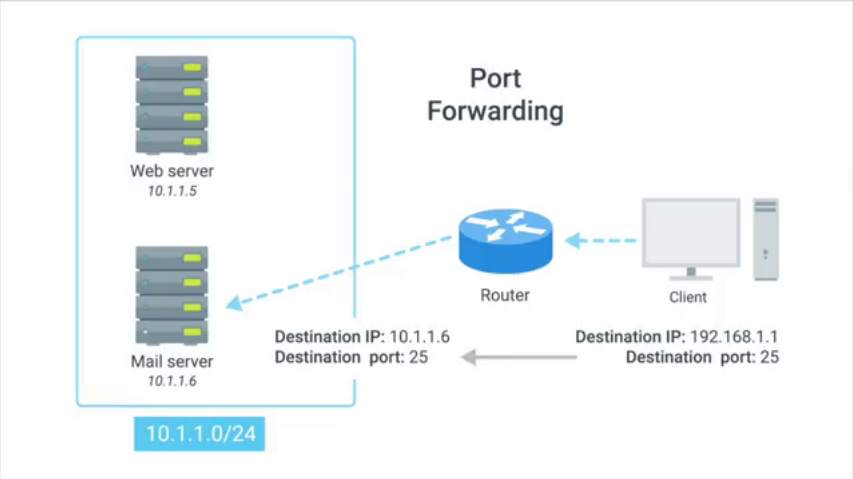
Now, when traffic returns to the router and port 51,300, it knows that this traffic needs to be forwarded back to the IP 10.1.1.100. Even with how large the set of ephemeral ports is, it's still possible for two different computers on a network to both choose the same source port around the same time. When this happens, the router normally selects an unused port at random to use instead.



Another important concept about NAT and the transport layer, is port forwarding. Port forwarding is a technique where a specific destination ports can be configured to always be delivered to specific nodes. This technique allows for complete IP masquerading, while still having services that can respond to incoming traffic.



Let's use our network 10.1.1.0 /24 again to demonstrate this. Let's say there's a web server configured with an IP of 10.1.1.5. With port forwarding, no one would even have to know this IP. Prospective web clients would only have to know about the external IP of the router. Let's say it's 192.168.1.1. Any traffic directed at port 80 on 192.168.1.1, would get automatically forwarded to 10.1.1.5. Response traffic would have the source IP rewritten to look like the external IP of the router.



This technique not only allows for IP masquerading, it also simplifies how external users might interact with lots of services all run by the same organization. Let's imagine a company with both a web server and mail server, both need to be accessible to the outside world but they run on different servers with different IPs. Again, let's say the web server has an IP of 10.1.1.5, and the mail server has an IP of 10.1.1.6. With port forwarding, traffic for either of these services could be aimed at the same external IP and therefore the same DNS name, but it would get delivered to entirely different internal servers due to their different destination ports.

4.4.3 NAT, Non-Routable Address Space and the Limits of IPv4

The IANA has been in charge of distributing IP addresses since 1988. Since that time, the internet has expanded at an incredible rate. The 4.2 billion possible IPv4 addresses have been predicted to run out for a long time and they almost have. For some time now, the IANA has primarily been responsible with assigning address blocks to the five regional internet registries or RIRs. The five RIRs are AFRINIC, which serves the continent of Africa, ARIN which serves the United States, Canada and parts of the Caribbean. APNIC, which is responsible for most of Asia, Australia and New Zealand and Pacific Island nations. LACNIC which covers Central and South America and any parts of the Caribbean not covered by ARIN. And finally RIPE, which serves Europe, Russia and the Middle East and portions of Central Asia. These five RIRs have been responsible for assigning IP address blocks to organizations within their geographic areas and most have already run out. The IANA assigned the last unallocated slash eight network blocks to various RIRs on February 3rd, 2011. Then in April 2011, APNIC ran out of addresses. RIPE was next, in September of 2012. LACNIC ran out of addresses to assign in June 2014. And ARIN did the same in September 2015. Only AFNIC has some IPs left, but those are predicted to be depleted by 2018. This is of course, a major crisis for the internet. IPv6 will eventually resolve these problems and we'll covered in more detail later in this course. But implementing IPv6 worldwide is going to take some time. For now, we wanted to continue to grow and we want more people and devices to connect to it but without IP addresses to assign, a workaround is needed.

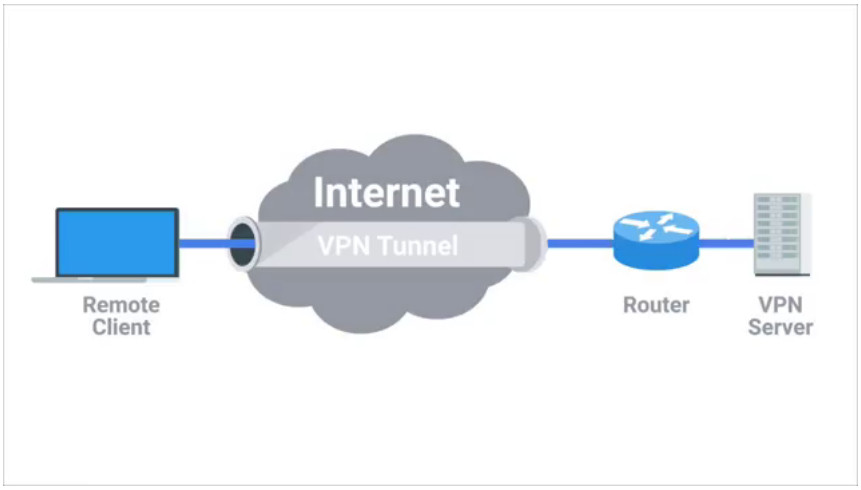
The major components of this workaround. NAT and non-routable address space. Remember that non-routable address space was defined in RFC1918 and consists of several different IP ranges that anyone can use. And unlimited number of networks can use non-routable address space internally because internet routers won't forward traffic to it. This means there's never any global collision of IP addresses when people use those address spaces. Non-routable address space is largely usable today because of technologies like NAT. With NAT, you can have hundreds even thousands of machines using non-routable address space. Yet, with just a single public IP, all those computers can still send traffic to and receive traffic from the internet. All you need is one single IPv4 address and via NAT, a router

with that IP can represent lots and lots of computers behind it. It's not a perfect solution, but until IPv6 becomes more globally available, non-routable address space and NAT will have to do.

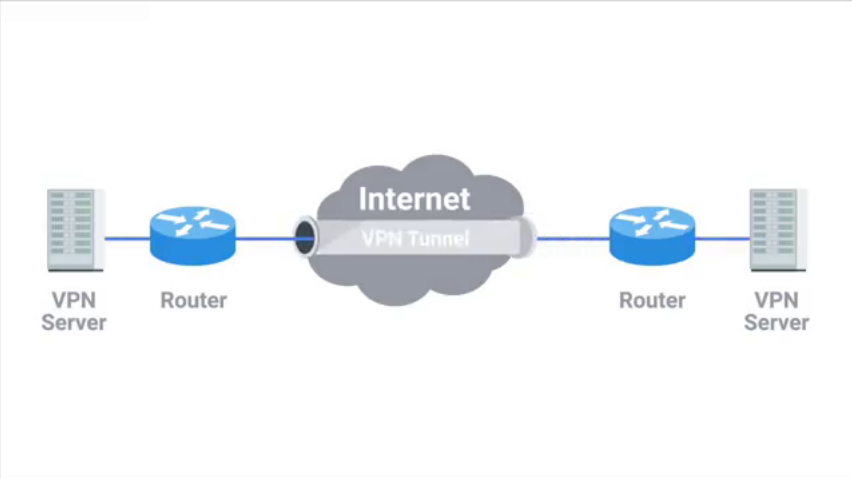
4.5 VPNs and Proxies

4.5.1 Virtual Private Networks

Businesses have lots of reasons to want to keep their network secure. And they do this by using some of the technologies we've already discussed. Firewalls, NAT, the use of non-routable address space, things like that. Organizations often have proprietary information that needs to remain secure. Network services that are only intended for employees to access, and other things. One of the easiest ways to keep networks secure is to use various securing technologies, so only devices physically connected to their local area network, can access these resources. But, employees aren't always in the office. They might be working from home, or on a business trip, and they might still need access to these resources in order to get their work done. That's where VPNs come in.
Virtual Private Networks or VPNs, are a technology that allows for the extension of a private or local network, to a host that might not work on that same local network. VPNs come in many flavors and accomplish lots of different things. But the most common example of how VPNs are used, is for employees to access their businesses network when they're not in the office. VPNs are a tunneling protocol. Which means, they provision access to something not locally available. When establishing a VPN connection, you might also say that a VPN tunnel has been established. Let's go back to the example of an employee who needs to access company resources while not in the office. The employee could use a VPN client to establish a VPN tunnel to their company network.



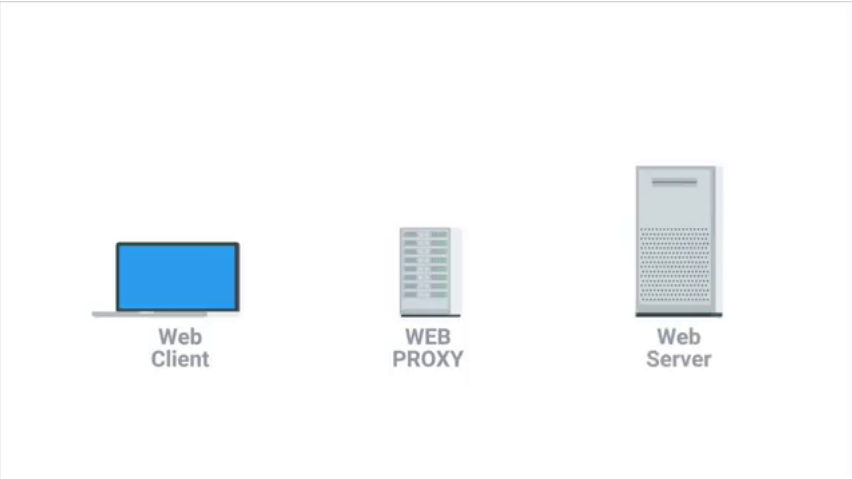
This would provision their computer with what's known as a virtual interface, with an IP that matches the address space of the network that established a VPN connection to. By sending data out of this virtual interface, the computer could access internal resources just like if it was physically connected to the private network. Most VPNs work by using the payload section of the transport layer to carry an encrypted payload that actually contains an entire second set of packets. The network, the transport, and the application layers of a packet intended to traverse the remote network. Basically, this payload is carried to the VPNs endpoint, where all the other layers are stripped away and discarded. Then, the payload is unencrypted, leaving the VPN server with the top three layers of a new packet. This gets encapsulated with the proper data link layer information, and sent out across the network. This process is completed in the inverse, in the opposite direction. VPNs, usually requires strict authentication procedures in order to ensure that they can only be connected to by computers and users authorized to do so. In fact, VPNs were one of the first technologies where two-factor authentication became common. Two-factor authentication is a technique where more than just a username and password are required to authenticate. Usually, a short-lived numerical token is generated by the user through a specialized piece of hardware or software.



VPNs can also be used to establish site-to-site connectivity. Conceptually, there isn't much difference between how this works compared to a remote employee situation. It's just that the router, or sometimes a specialized VPN device on one network, establishes the VPN tunnel to the router or VPN device on another network. This way, two physically separated offices might be able to act as one network and access network resources across the tunnel. It's important to call out that just like NAT, VPNs a general technology concept, not a strictly defined protocol. There are lots of unique implementations of VPNs. And the details of how they all work can differ a ton. The most important takeaway is that VPNs are a technology that use encrypted tunnels to allow for a remote computer or network, to act as if it's connected to a network that it's not actually physically connected to.

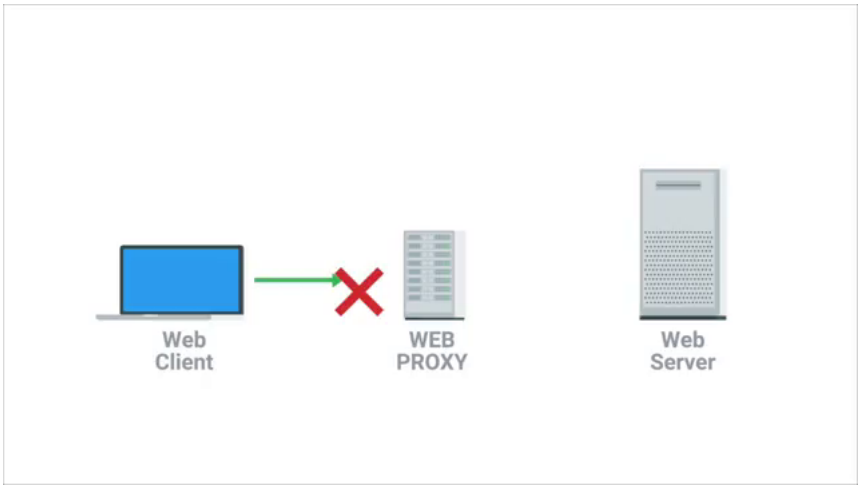
4.5.2 Proxy Services

A proxy service is a server that acts on behalf of a client in order to access another service. Proxies sit between clients and other servers, providing some additional benefit such as anonymity, security, content filtering, increased performance, a couple other things. The concept of a proxy is just that, a concept or an abstraction. It doesn't refer to any specific implementation. Proxies exist at almost every layer of our networking model.

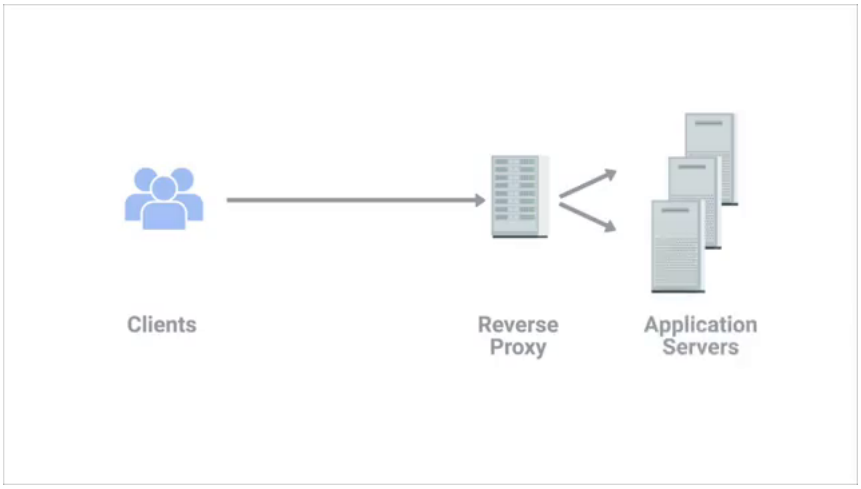


Most often, you'll hear the term, proxy, used to refer to web proxies. As you might guess, these are proxies specifically built for web traffic. A web proxy can serve lots of purposes. Many years ago, when most Internet connections were much slower than they are today, lots of organizations used web proxies for increased performance. Using a web proxy, an organization would direct all web traffic through it, allowing the proxy server itself to actually retrieve the webpage data from the Internet.
It would then cache this data. This way, if someone else requested the same webpage, it could just return the cached data instead of having to retrieve the fresh copy every time. This kind of proxy is pretty old and you wont

often find them in use today, why? Well, for one thing, most organizations now have connections fast enough that caching individual webpages doesn't provide much benefit. Also, the Web has become much more dynamic. Going to www.twitter.com is going to look different to every person with their own Twitter account, so caching this data wouldn't do much good.

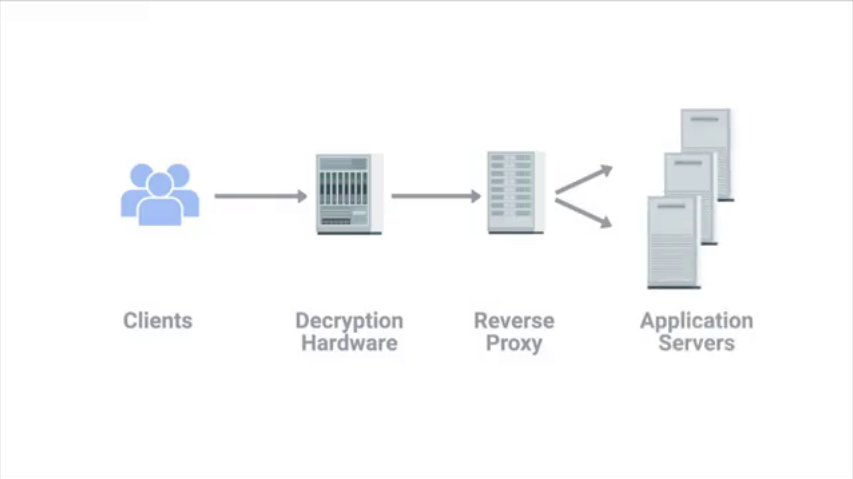


A more common use of a web proxy today might be to prevent someone from accessing sites, like Twitter, entirely. A company might decide that accessing Twitter during work hours reduces productivity. By using a web proxy, they can direct all web traffic to it, allow the proxy to inspect what data is being requested, and then allow or deny this request, depending on what site is being accessed.



Another example of a proxy is a reverse proxy. A reverse proxy is a service that might appear to be a single server to external clients, but actually represents many servers living behind it. A good example of this is how lots of popular websites are architected today. Very popular websites, like Twitter, receive so much traffic that there's no way a single web server could possibly handle all of it. A website that popular might need many, many web servers in order to keep up with processing all incoming requests.

A reverse proxy, in this situation, could act as a single front-end for many web servers living behind it. From the clients' perspective, it looks like they're all connected to the same server. But behind the scenes, this reverse proxy server is actually distributing these incoming requests to lots of different physical servers. Much like the concept of DNS Round Robin, this is a form of load balancing.



Another way that reverse proxies are commonly used by popular websites is to deal with decryption. More than half of all traffic on the Web is now encrypted, and encrypting and decrypting data is a process that can take a lot of processing power. Reverse proxies are now implemented in order to use hardware built specifically for cryptography, to perform the enryption and decryption work. So that the web servers are free to just serve content.
Proxies come in many other flavors, way too many for us to cover them all here. But the most important takeaway is that proxies are any server that act as a intermediary between a client and another server.

Chapter 5

Network Troubleshooting

Outcome

1. You'll learn about the most common techniques and tools you use as an IT support specialist when troubleshooting network issues.
2. You'll be able to use a number of important troubleshooting tools to help resolve network issues
3. , You'll be able to detect and fix a lot of the common network connectivity problems, by using tools available on the three most common operating systems: Microsoft Windows, Mac OS and Linux.
4. You'll be learn the concepts that are important to the future of networking: the cloud and IPv6.

5.1 Introduction to Troubleshooting

Computer networking can be an incredibly complicated business. There are so many layers, protocols and devices at play. And sometimes this means that things just don't work properly. Many of the protocols and devices we've covered have built-in functionalities to help protect against some of these issues. These functionalities are known as error detection and error recovery.

Error-detection
The ability for a protocol or program to determine that something went wrong

Error-recovery
The ability for a protocol or program to attempt to fix it

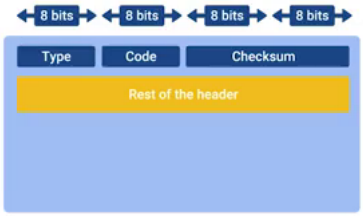
For example, cyclical redundancy checks are used by multiple layers to make sure that the correct data was received by the receiving end. If a CRC value doesn't match the data payload, the data is discarded. At that point, the transport layer will decide if the data needs to be reset. But, even with all of these safeguards in place, errors still pop up. Misconfigurations occur, hardware breaks down and system incompatibilities come to light.

5.2 Verifying Connectivity

5.2.1 Ping: Internet Control Message Protocol

When network problems come up, the most common issue run into is the inability to establish a connection to something. It could be a server can't reach at all or a website that isn't loading. Maybe you can only reach your resource on your LAN and can't connect to anything on the internet. Whatever the problem is, being able to diagnose connectivity issues is an important part of network troubleshooting.

When a network error occurs, the device that detects it needs some way to communicate this to the source of the problematic traffic. It could be that a router doesn't know how to route to a destination or that a certain port isn't reachable. It could even be that the TTL of an IP datagram expired and no further router hops will be attempted. For all of these situations and more, ICMP or internet control message protocol is used to communicate these issues. ICMP is mainly used by router or remote hosts to communicate while transmission has failed back to the origin of the transmission. The makeup of an ICMP packet is simple.



It has a header with a few fields and a data section that's used by host to figure out which of their transmissions generated the error.

1. Type field, eight bits long which specifies what type of message is being delivered.

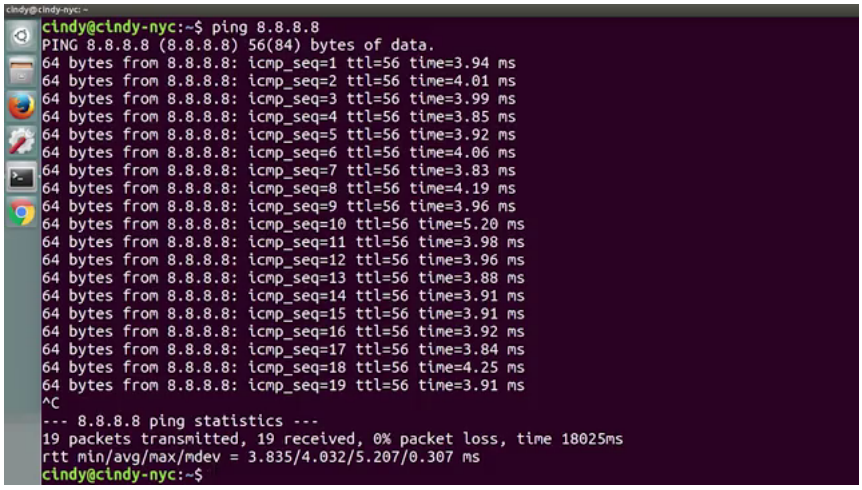
   Note

   Some examples for type field are destination unreachable or time exceeded. Immediately after this is the code field which indicates a more specific reason for the message. For example, of the destination unreachable type, there are individual codes for things like destination network unreachable and destination port unreachable.

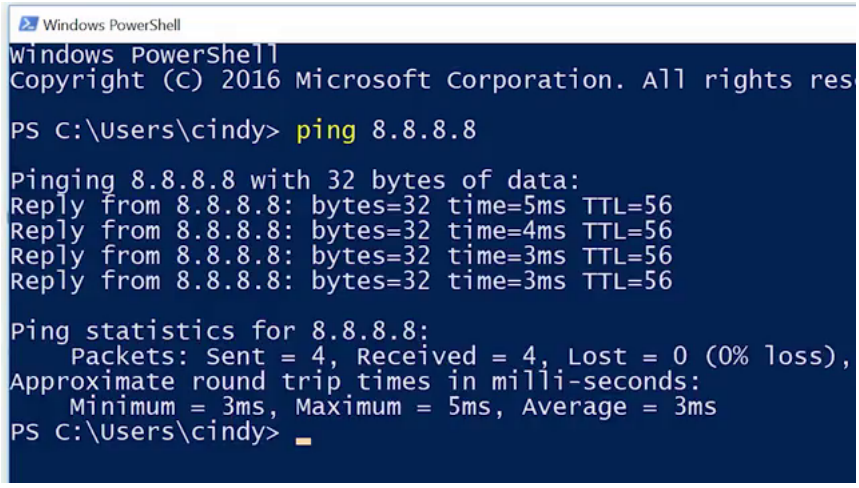
2. Code : This field is optionally used by some of the specific types and codes to send more data.
3. 16 bit checksum that works like every other checksum field .
4. Rest of header
5. After this is the data payload for an ICMP packet. The payload for an ICMP packet exists entirely so that the recipient of the message knows which of their transmissions caused the error being reported. It contains the entire IP header and the first eight bytes of the data payload section of the offending packet.

ICMP wasn't really developed for humans to interact with. The point is so that these sorts of error messages can be delivered between networked computers automatically. But, there's also a specific tool and two message types that are very useful to human operators. This tool is called ping.

Some version of it exist on just about every operating system, and has for a very long time. Ping is a super simple program and the basics are the same no matter which operating system you're using. Ping lets you send a special type of ICMP message called an Echo Request. An ICMP echo request essentially just ask the destination, "Hey, are you there?" If the destination is up and running and able to communicate on the network, it will send back an ICMP echo reply message type. You can invoke the ping command from the command line of any modern operating system. In its most basic use, you just type ping and a destination IP or a fully qualified domain name. Output of the ping command is very similar across each of the different operating systems. Every line of output will generally display the address sending the ICMP echo reply, and how long it took for the round trip communications.



It will also have the TTL remaining and how large the ICMP message is in bytes. Once the command ends, there will also be some statistics displayed like percentage of packets transmitted and received, the average round trip time, and a couple of other things like that. On Linux and Mac OS, the ping command will run until it's interrupted by an end user sending an interrupt event. They do this by pressing the control key and the C key at the same time.



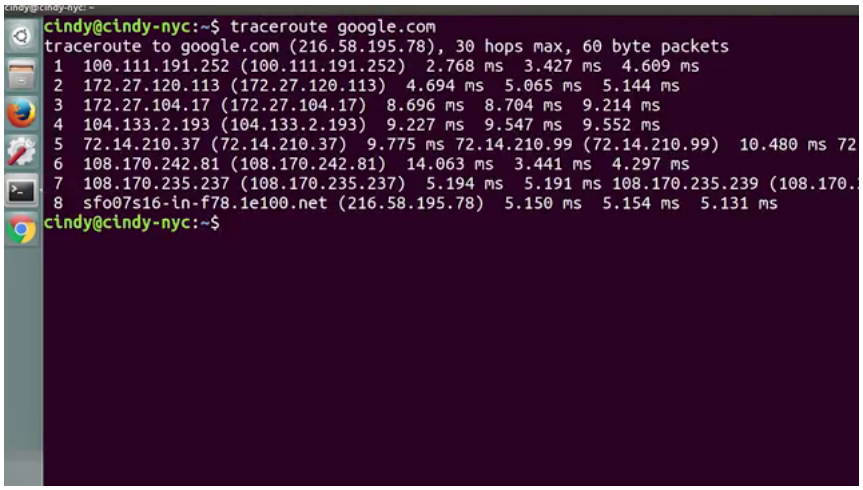
On Windows, ping defaults to only sending four echo requests. In all environments, ping supports a number of command line flags that let you change its behavior like the number of echo requests to send, how large they should be, and how quickly they should be sent.

5.2.2 Traceroute

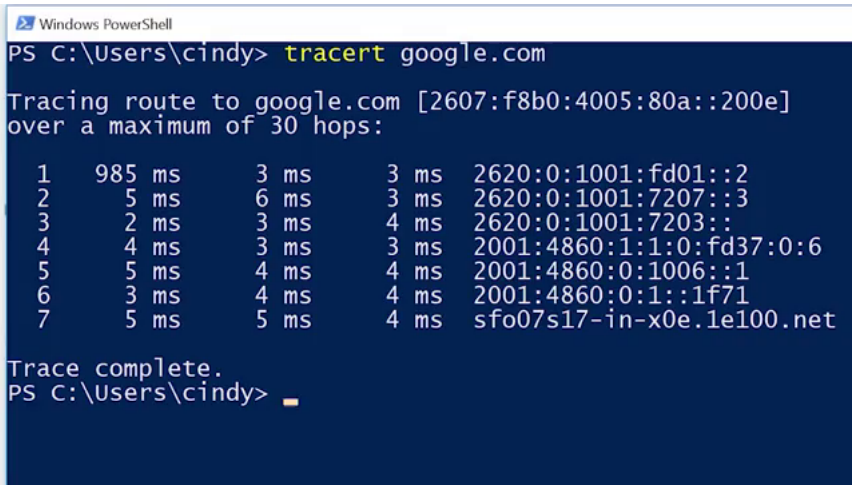
With ping, you now have a way to determine if you can reach a certain computer from another one. You can also understand the general quality of the connection. But communications across networks, especially across the Internet usually, cross lots of intermediary nodes. Sometimes, you need a way to determine where in the long chain of router hops the problems actually are.

Traceroute is an utility that lets you discover the paths between two nodes, and gives you information about each hop along the way. The way traceroute works, is through a clever manipulation technique of the TTL field at the IP level. We learned earlier that the TTL field is decremented by one, by every router that forwards the packet. When the TTL field reaches zero, the packet is discarded and an ICMP Time Exceeded message is sent back to the originating host.
Traceroute uses the TTL field by first setting it to one for the first packet, then two for the second, three for the third and so on. By doing this clever little action, traceroute makes sure that the very first packet sent will be discarded by

the first router hop. This results in an ICMP Time Exceeded message, the second packet will make it to the second router, the third will make it to the third, and so on. This continues until the packet finally makes it all the way to its destination. For each hop, traceroute will send three identical packets.



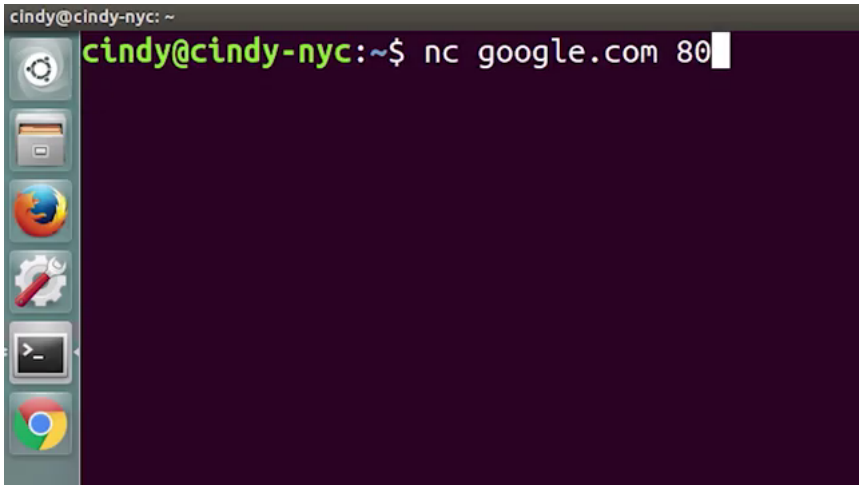
Just like with ping, the output of a traceroute command is simple. On each line, you'll see the number of the hop and the round trip time for all three packets. You will also see the IP of the device at each hop, and a host name if traceroute can resolve one. On Linux and MacOS, traceroute sends UDP packets to very high port numbers.



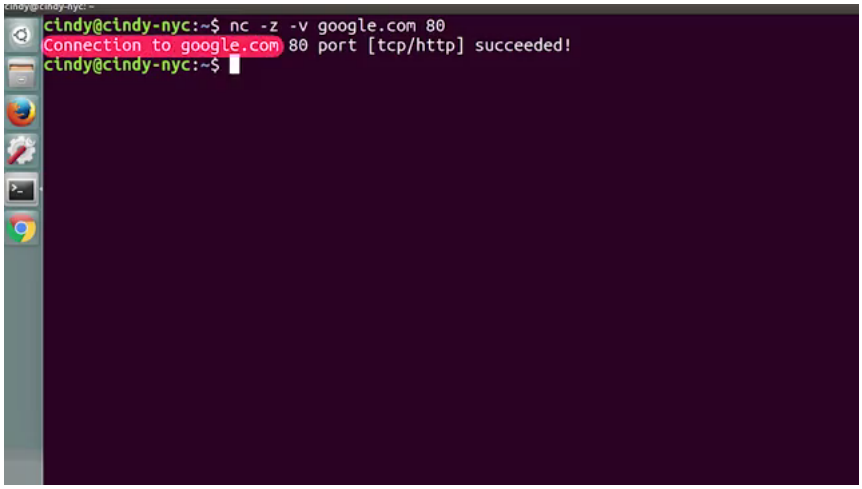
On Windows, the command has a shortened name tracert, and defaults to using ICMP echo request. On all platforms, traceroute has more options than can be specified using command line flags. Two more tools that are similar to traceroute are mtr on Linux and MacOS and pathping on Windows. These two tools act as long running traceroutes. So you can better see how things change over a period of time. Mtr works in real time and will continually update its output with all the current aggregate data about the traceroute. You can compare this with pathping, which runs for 50 seconds and then displays the final aggregate data all at once.

5.2.3 Testing Port Connectivity

Sometimes, you need to know if things are working at the transport layer. For this, there are two powerful tools at your disposal. Netcat on Linux and Mac OS and Test-NetConnection on Windows. The Netcat tool can be run through the command nc, and has two mandatory arguments, a host and a port.

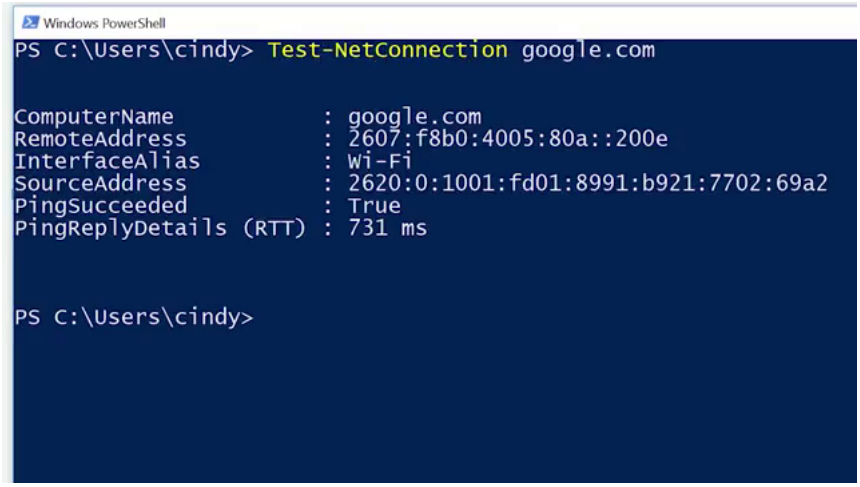


Running nc google.com 80 would try to establish a connection on port 80 to google.com. If the connection fails, the command will exit. If it succeeds, you'll see a blinking cursor, waiting for more input. This is a way for you to actually send application layered data to the listening service from your own keyboard. If you're really only curious about the status of a report, you can issue the command, with a -Z flag, which stands for Zero Input/Output Mode. A -V flag, which stands for Verbose, is also useful in this scenario. This makes the commands output useful to human eyes as opposed to non-verbose output, which is best for usage in scripts.



So by issuing the Netcat command with the -Z and -V flags, the command's output will simply tell you if a connection to the port in question is possible or not.

On Windows, Test-NetConnection is a command with some of the similar functionality. If you run Test-NetConnection with only a host specified, it will default to using an ICMP echo request, much like the program ping. But, it will display way more data, including the data link layer protocol being used. When you issue Test-NetConnection with the -port flag, you can ask it to test connectivity to a specific port.

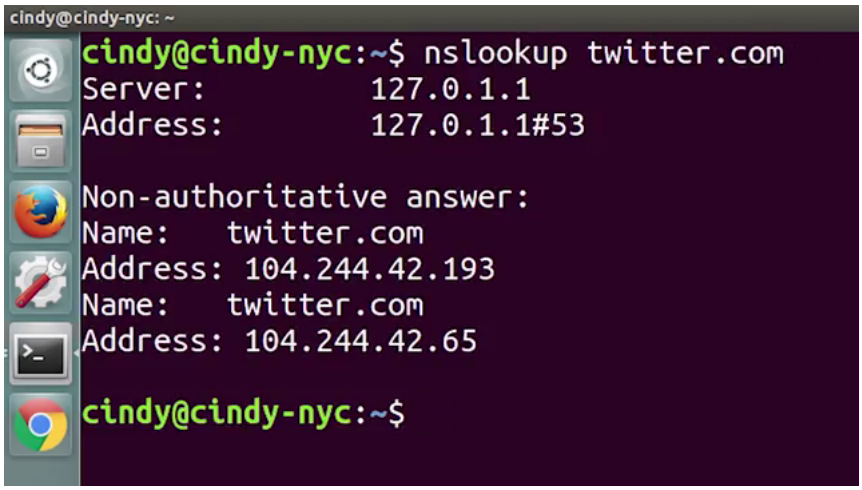


It's important to call out that both Netcat and Test-NetConnection are way more powerful than the brief port connectivity examples we've covered.

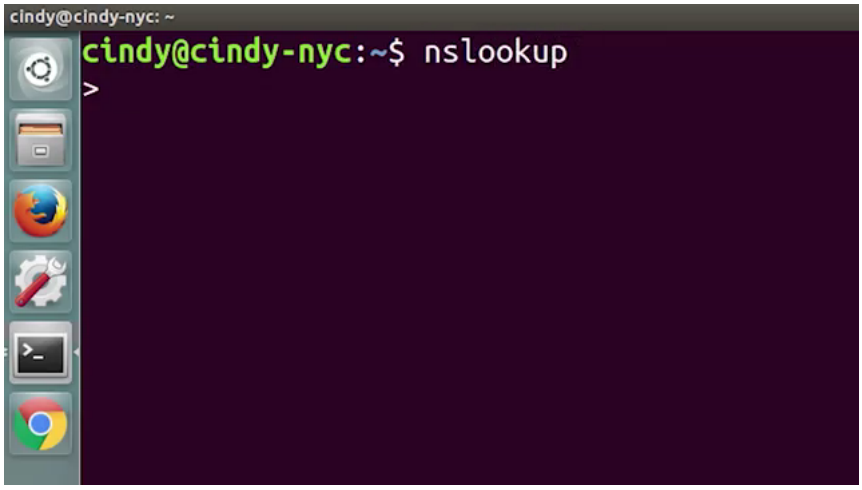
5.3 Digging into DNS

5.3.1 Name Resolution Tools

Name resolution is an important part of how the Internet works. Most of the time, your operating system handles all look ups for you. The most common tool is known as nslookup. And it's available on all three of the operating system, Linux, Mac, and Windows. A basic use of nslookup is simple. You execute the nslookup command with the host name following it. And the output displays what server was used to perform the request and the resolution result.



Let's say you needed to know the IP address for a twitter.com. You would just enter nslookup twitter.com and the A record would be returned. Nslookup is way more powerful than just that. It includes an interactive mode that lets you set additional options and run lots of queries in a row. To start an interactive nslookup session, you just enter nslookup, without any hostname following it. You should see an angle bracket acting as your prompt.



From interactive mode, you can make lots of requests in a row. You can also perform some extra configuration to help with more in-depth trouble shooting. While in interactive mode, if you type server, then an address, all the following name resolution queries will be attempted to be made using that server instead of the default name server.

5.3.2 Public DNS Servers

Having functional DNS is an important part of a functional network. An ISP almost always gives you access to a recursive name server as part of the service it provides. In most cases, these name servers are all you really need for your computer to communicate with other devices on the internet.
But, most businesses also run their own DNS servers. In the very least, this is needed to resolve names of internal hosts. Anything from naming a computer, Nayas- laptop, to being able to refer to a printer by name instead of an IP requires your own name server. A third option is to use a DNS as a service provider, and it's getting more and more popular.

Some internet organizations run what are called public DNS servers, which are name servers specifically set up so that anyone can use them for free. Using these public DNS servers is a handy technique for troubleshooting any kind of name resolution problems you might be experiencing.

The IPs for Google's public DNS. Google operates public name servers on the IPs 8.8.8.8 and 8.8.4.4, these are officially acknowledged and documented by Google to be used for free by anyone. Most public DNS servers are available globally through anycast. Lots of other organizations also provide public DNS servers, but few are as easy to remember as those two options. Always do your research before configuring any of your devices to use that type of name server. Hijacking outbound DNS requests with faulty responses is an easy way to redirect your users to malicious sites. Always make sure the name server is run by a reputable company, and try to use the name servers provided by your ISP outside of troubleshooting scenarios. Most public DNS servers also respond to ICMP echo requests, so they're a great choice for testing general internet connectivity using ping.

5.3.3 DNS Registration and Expiration

DNS is a global system managed in a tiered hierarchy with ICANN at the top level. Domain names need to be globally unique for a global system like this to work. You can't just have anyone decide to use any domain name. It would be chaos. Enter the idea of a registrar, an organization responsible for assigning individual domain names to other organizations or individuals. Originally, there were only a few registrars. The most notable was a company named Network Solutions Inc. It was responsible for the registration of almost all domains that weren't country specific. As the popularity of the Internet grew, there was eventually enough market demand for competition in this space. Finally, the United States government and Network Solutions Inc. came to an agreement to let other companies also sell domain names.

Today, there are hundreds of companies like this all over the world. Registering a domain name for use is pretty simple. Basically, you create an account with the registrar, use their web UI to search for a domain name to determine if it's still available, then you agree upon a price to pay and the length of your registration. Once you own the domain name, you can either have the registrar's name servers act as the authoritative name servers for the domain, or you can configure your own servers to be authoritative. Domain names can also be transferred by one party to another and from one registrar to another.

The way this usually works is that the recipient registrar will generate a unique string of characters to prove that you own the domain and that you're allowed to transfer it to someone else. You configure your DNS settings to contain the string in a specific record, usually a TXT record. Once this information has propagated, it can be confirmed that you both own the domain and approve its transfer. After that, ownership would move to the new owner or registrar. An important part of domain name registration is that these registrations only exist for a fixed amount of time. You typically pay to register domain names for a certain number of years. It's important to keep on top of when your domain names might expire because once they do, they're up for grabs and anyone else could register them.

5.4 IPv6

5.4.1 IPv6 Addressing and Subnetting

The IANA is out of IP addresses. When IPv4 was first developed, a 32-bit number was chosen to represent the address for a node on a network. The Internet was in its infancy, and no one really expected it to explode in popularity the way it has. 32 bits were chosen, but it's just not enough space for the number of Internet-connected devices we have in the world. IPv6 was developed exactly because of this issue. By the mid 1990s, it was more and more obvious that we were going to run out of IPv4 address space at some point. So a new Internet protocol was developed, Internet Protocol version 6, or IPv6. You might wonder what happened to version 5 or IPv5. IPv5 was an experimental protocol that introduced the concept of connections. It never really saw wide adoption, and connection state was handled better later on by the transport layer and TCP. Even though IPv5 is mostly a relic of history, when development of IPv6 started, the consensus was to not reuse the IPv5 name.
The biggest difference between IPv4 and IPv6 is the number of bits reserved for an address. While IPv4 addresses are 32 bits, meaning there can be around 4.2 billion individual addresses, IPv6 addresses are 128 bits in size. This size difference is staggering once you do the math. 2 to the power of 128 would produce a 39-digit-long number. That number range has a name you've probably never even heard of, an undecillion. An undecillion isn't a number you hear a lot because it's ginormous. There really aren't things that exist at that scale. Some guesses on the total number of atoms that make up the entire planet Earth and every single thing on it get into that number range. That should tell you we're talking about a very, very large number. If we can give every atom on Earth its own IP address, we're probably be okay when it comes to network devices for a very long time.
Just like how an IPv4 address is really just a 32-bit binary number, IPv6 addresses are really just 128-bit binary numbers. IPv4 addresses are written out in four octets of decimal numbers just to make them a little more readable for humans. But trying to do the same for an IPv6 address just wouldn't work. Instead, IPv6 addresses are usually written out as 8 groups of 16 bits each.

Each one of these groups is further made up of four hexadecimal numbers. so IPv6 has a notation method that lets us break that down even more. A way to show how many IPv6 addresses there are is by looking at our example IP. Every single IPv6 address that begins with 2001:0db8 has been reserved for documentation and education, or for books. That's over 18 quintillion addresses, much larger than the entire IPv4 address space, reserved just for this purpose. There are two rules when it comes to shortening an IPv6 address. The first is that you can remove any leading zeros from a group. The second is that any number of consecutive groups composed of just zeros can be replaced with two colons.

I should call out that this can only happen once for any specific address. Otherwise, you couldn't know exactly how many zeros were replaced by the double colons. For this IP, we could apply the first rule and remove all leading zeros from each group. Once we apply the second rule, which is to replace consecutive sections containing just zeros with two colons. This still isn't as readable as an IPv4 address, but it's a good system that helps reduce the length a little bit. We can see this approach taken to the extreme with IPv6 loopback address. You might remember that with IPv4, this address is 127.0.0.1. With IPv6, the loopback address is 31 0s with a 1 at the end, which can be condensed all the way down to just ::1.

The IPv6 address space has several other reserved address ranges besides just the one reserved for documentation purposes or the loopback address. For example, any address that begins with FF00:: is used for multicast, which is a way of addressing groups of hosts all at once. It's also good to know that addresses beginning with FE80:: are used for link-local unicast. Link-local unicast addresses allow for local network segment communications and are configured based upon a host's MAC address. The link-local address are used by an IPv6 host to receive their network configuration, which is a lot like how DHCP works. The host's MAC address is run through an algorithm to turn it from a 48-bit number into a unique 64-bit number. It's then inserted into the address's host ID. The IPv6 address space is so huge, there was never any need to think about splitting it up into address classes like we use to do with IPv4. From the very beginning, an IPv6 address had a very simple line between network ID and host ID. The first 64 bits of any IPv6 address is the network ID, and the second 64 bits of any IPv6 address is the host ID. This means that any given IPv6 network has space for over 9 quintillion hosts. Still, sometimes network engineers might want to split up their network for administrative purposes. IPv6 subnetting uses the same CIDR notation that you're already familiar with. This is used to define a subnet mask against the network ID portion of an IPv6 address.