GOOGLE IT SUPPORT SPECIALIZATION
COURSE 2: COMPUTER NETWORKING
02 THE NETWORK LAYER

Table des matières

1 Introduction to networking

2 The network layer

2.1 Introduction to the network layer

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 course, we learned about how computers communicate with each other over short distances or on a single network segment or LAN. In these next lessons, we'll focus on the technologies that allow data to cross many networks facilitating communications over great distances.

By the end of this module, you'll be able to describe the IP addressing scheme and how Subnetting works. This means, you'll learn how to perform basic math in binary in order to describe subnets.

You'll also be able to demonstrate how encapsulation works, and how protocols such as ARP allow different layers of the network to communicate.

And finally, you'll gain an understanding of the basics behind routing, routing protocols and how the internet works. For now, route yourself to the next video and we'll get started.

2.2 The network layer

2.2.1 The network layer

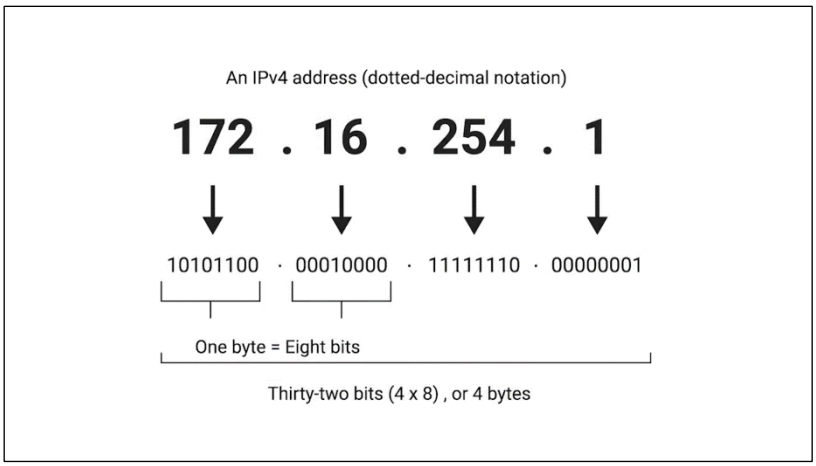
On a local area network or LAN, nodes can communicate with each other through their physical MAC addresses. This works well on small scale because switches can quickly learn the MAC addresses connected to each other ports to forward transmissions appropriately.

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.

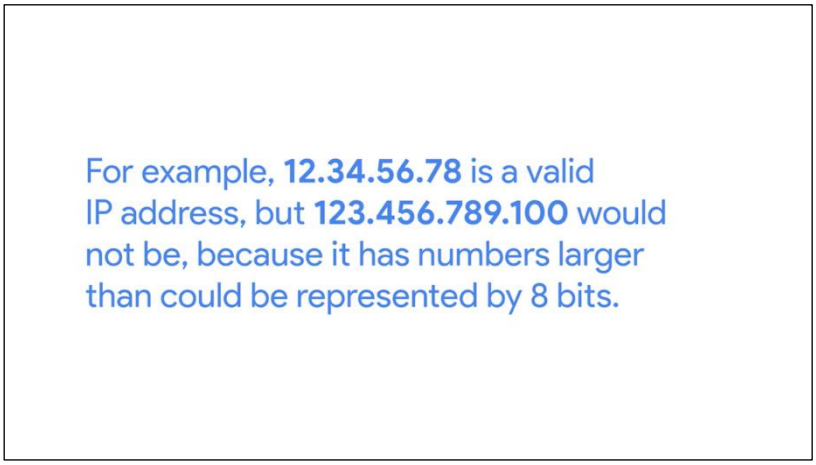
Later on in this lesson, when we introduce ARP or Address Resolution Protocol, you'll see that the way that nodes learn about each other's physical addressing isn't translatable to anything besides a single network signet anyway. Clearly we need another solution, and that is the network layer, and the 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.

2.2.2 IP adresses

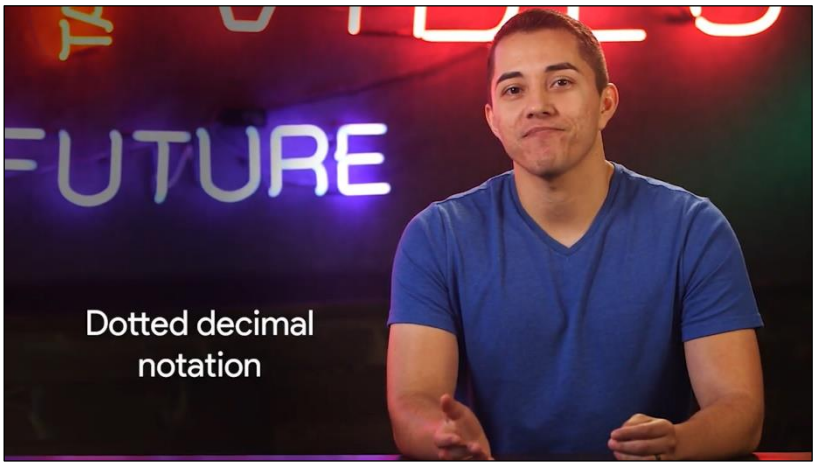
IP addresses are 32-bit long numbers made up of 4 octets, and each octet is normally described in decimal numbers. 8 bits of data, or a single octet, can represent all decimal numbers from 0 to 255.



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.



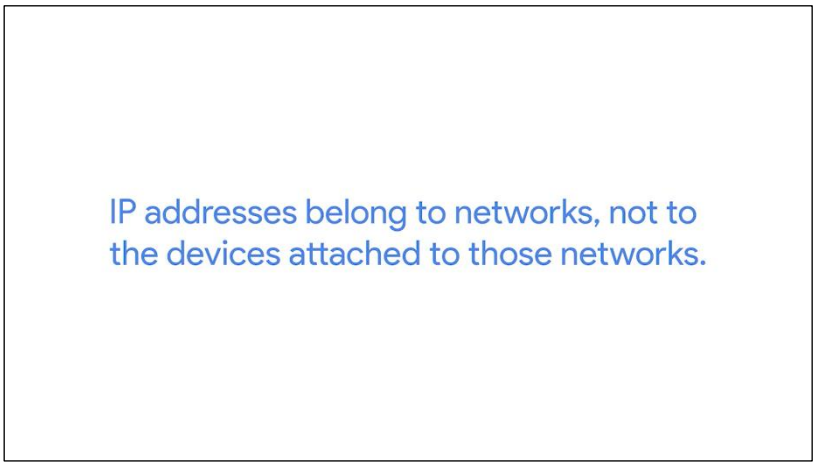
We'll deep dive into how some of this works in an upcoming lesson about subnetting.

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, 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.

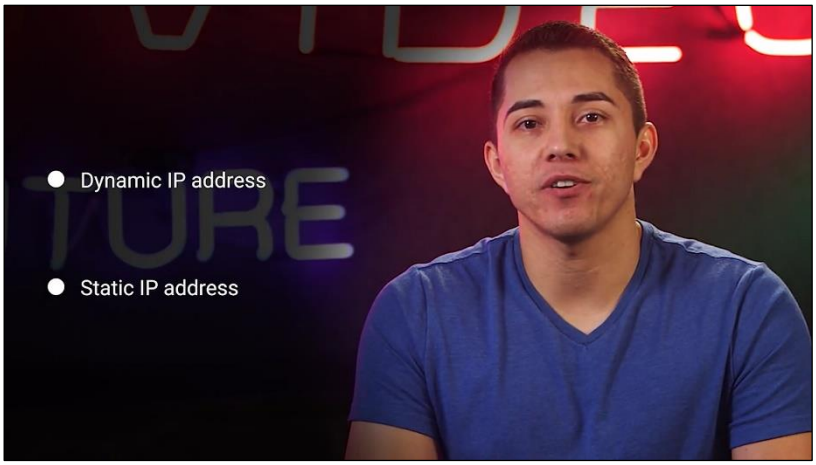
It's important to call out that IP addresses belong to the networks, not the devices attached to those networks.



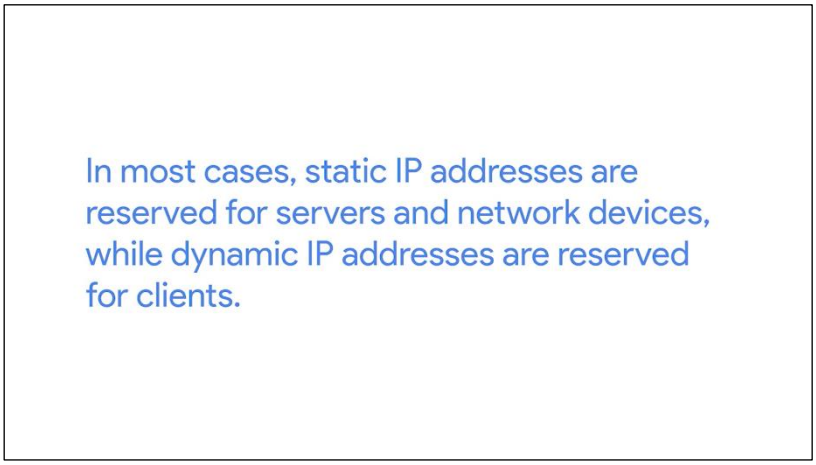
So your laptop will always have the same MAC address, no matter where you use it. But it'll have a different IP address assigned to it at an Internet cafe than it would when you're at home. The LAN at the Internet cafe or the LAN at your house would each be individually responsible for handing out an 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.

You'll learn more about some of the technologies at play in a later lesson. 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.



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.

2.2.3 IP datagrams and encapsulation

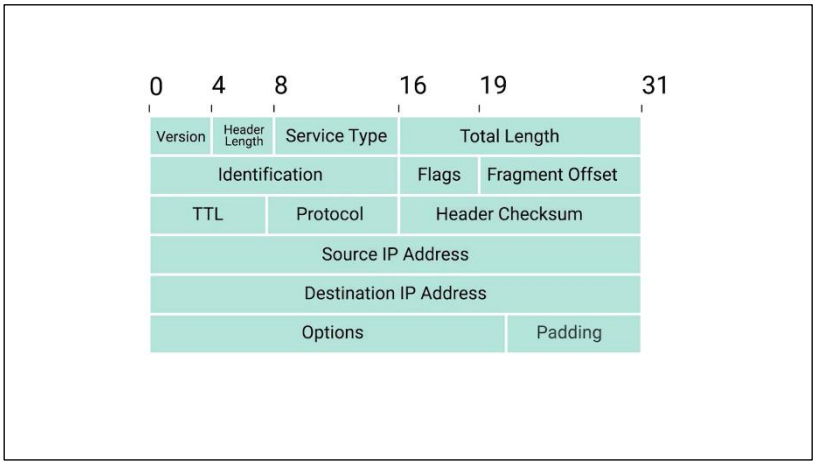
Just like all the data packets at the Ethernet layer have a specific name, Ethernet frames, so do packets at the network layer. Under the IP protocol, a packet is usually referred to as an IP datagram.

Just like any Ethernet frame, an IP datagram is a highly structured series of fields that are strictly defined.

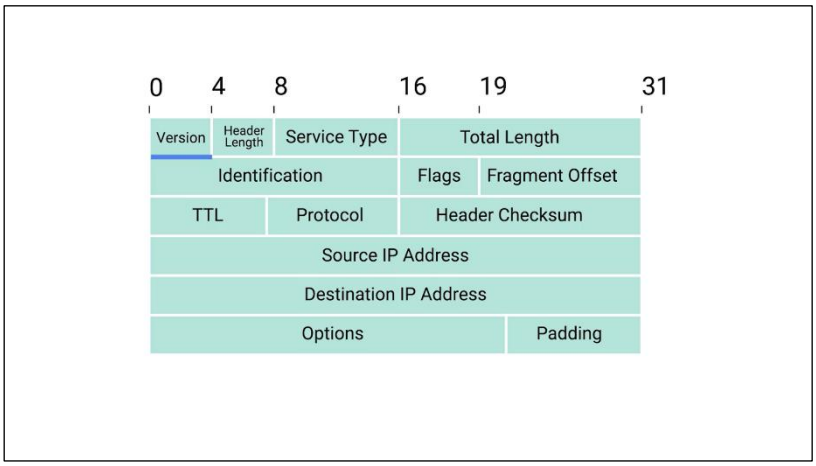
IP datagram

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.



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

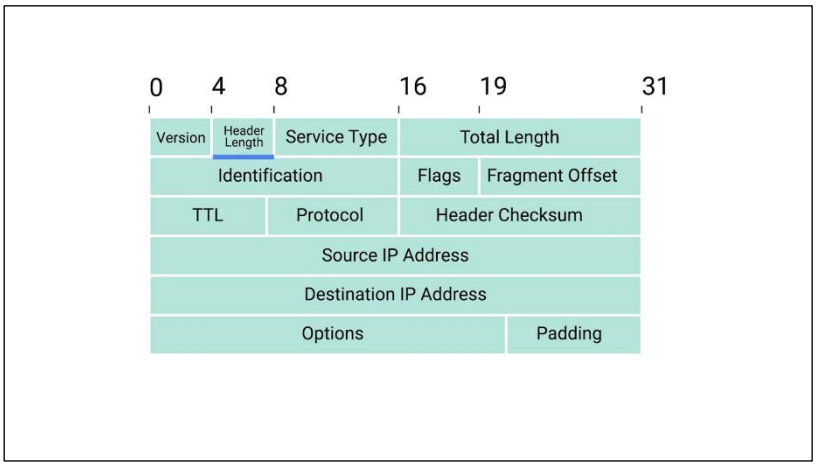


The most common version of IP is version four or IPv4.

The most common version of **IP** is version 4, or **IPv4**.

Version six or IPv6, is rapidly seeing more widespread adoption, but we'll cover that in a later module.

After the version field, we have the Header Length field.



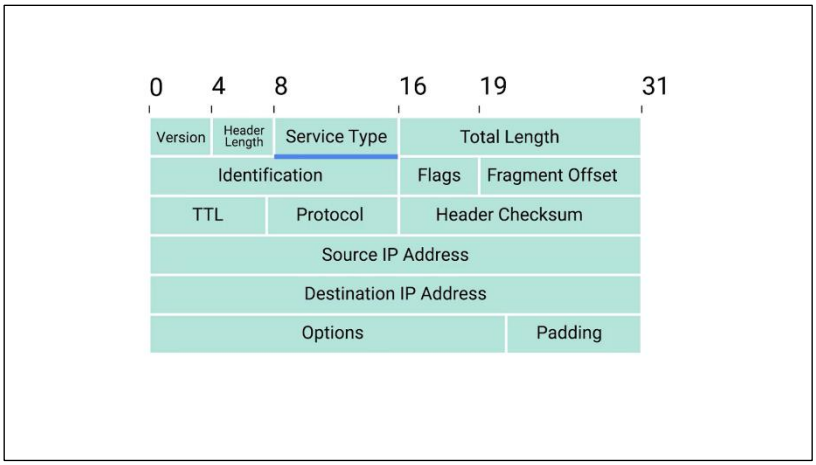
This is also a four bit field that declares how long the entire header is. This is almost always 20 bytes in length when dealing with IPv4.

Header Length field

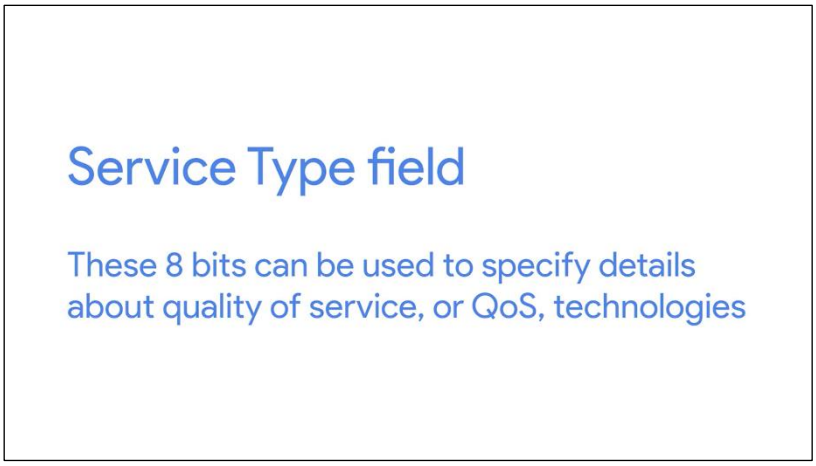
Almost always 20 bytes in length when dealing with IPv4

In fact, 20 bytes is the minimum length of an IP header. You couldn't fit all the data you need for a properly formatted IP header in any less space.

Next, we have the Service Type field.

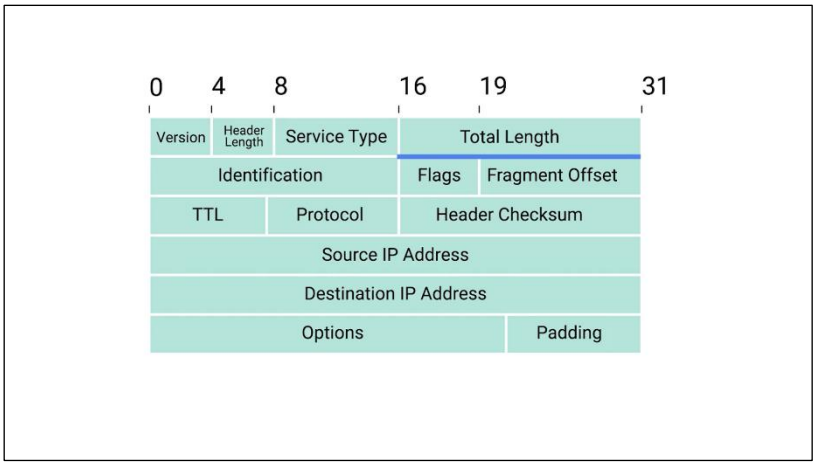


These eight bits can be used to specify details about quality of service or QoS technologies.



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.

The next field is a 16 bit field, known as the Total Length field.

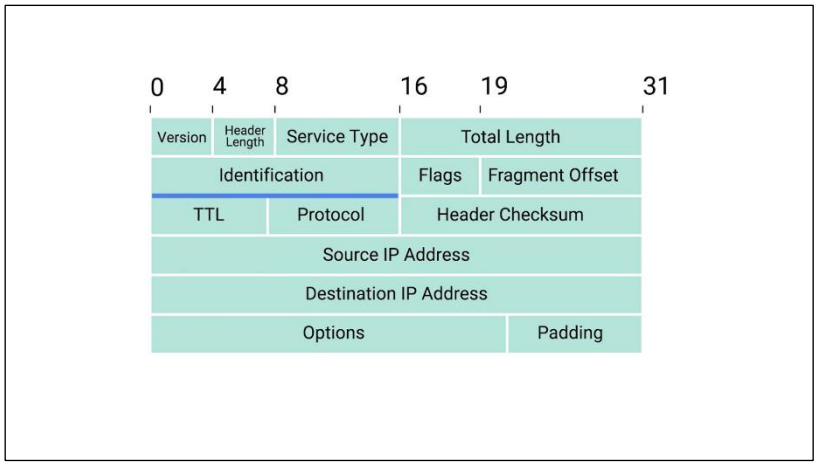


It's used for exactly what it sounds like; to indicate the total length of the IP datagram it's attached to.

Total Length field

Indicates the total length of the IP datagram it's attached to

The identification field, is a 16-bit number that's used to group messages together.



Identification field

A 16-bit number that's used to group messages together

IP datagrams have a maximum size and you might already be able to figure out what that is. 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.

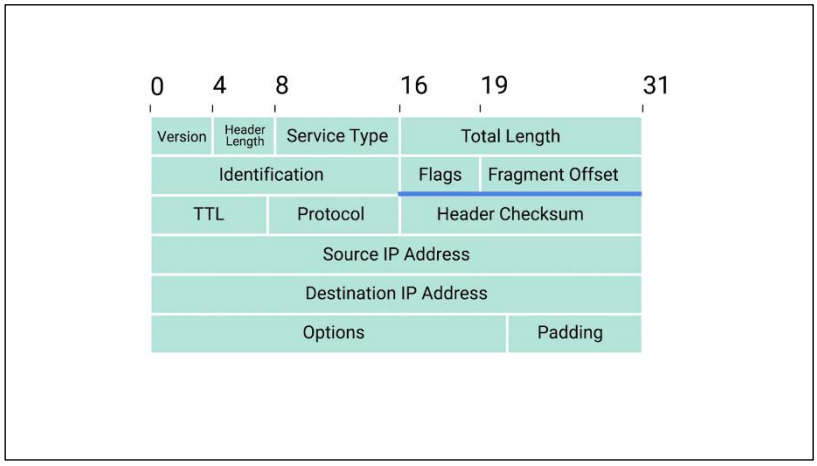
The maximum size of a single datagram is the largest number you can represent with 16 bits: 65,535.

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.

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.

Next up, we have two closely related fields. 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.

Flag field

Used to indicate if a datagram is allowed to be fragmented, or to indicate that the datagram has already been fragmented

Fragmentation is the process of taking a single IP datagram and splitting it up into several smaller datagrams.

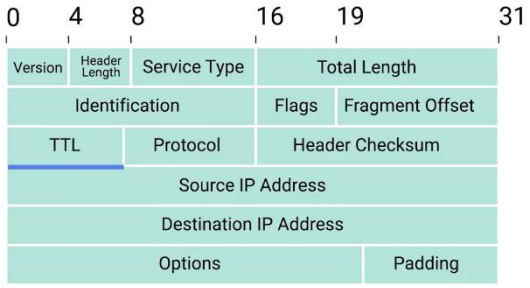
Fragmentation

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.

Let's move along to The Time to Live or TTL field.



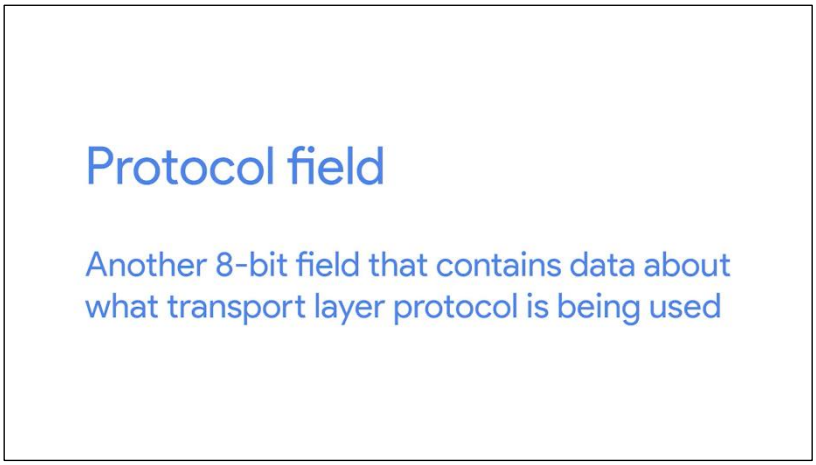
This field is an 8-bit field that indicates how many router hops a datagram can traverse before it's thrown away.

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.

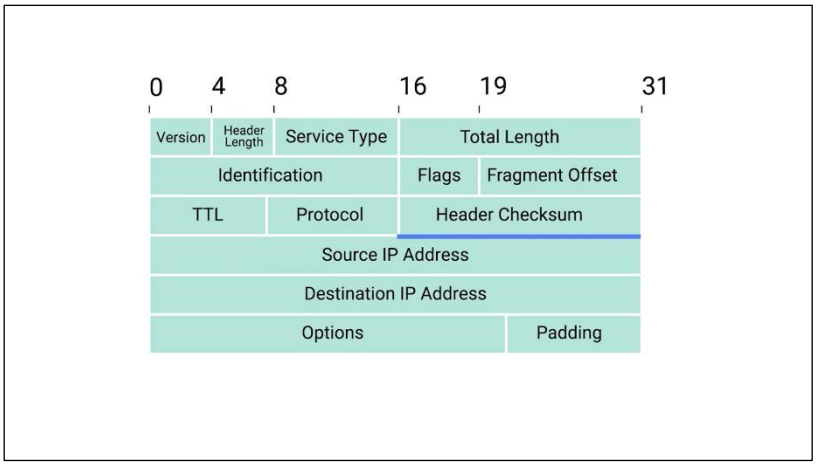
Spoiler alert. In an upcoming module, you'll learn that the TTL field has valuable troubleshooting qualities, but secrets like these are only released to those who keep going.

After the TTL field, you'll find the Protocol field.

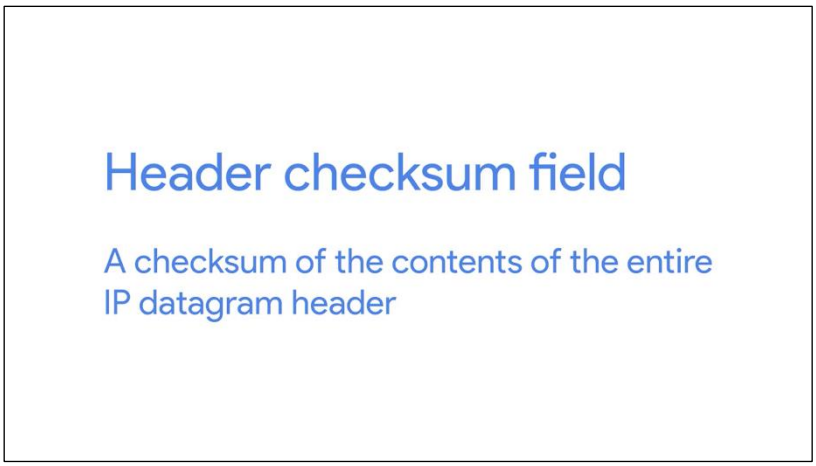


This is another 8-bit field that contains data about what transport layer protocol is being used. The most common transport layer protocols are TCP and UDP, and we'll cover both of those in detail in the next few lessons.

So next, we find the header checksum field.

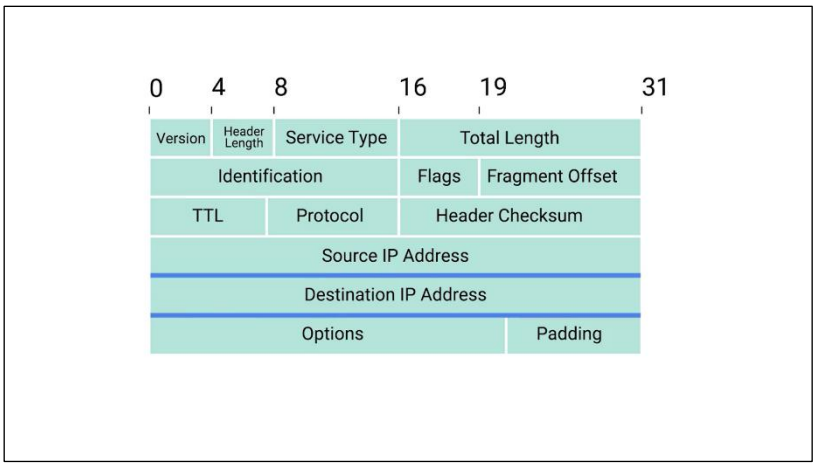


This field is a checksum of the contents of the entire IP datagram header.



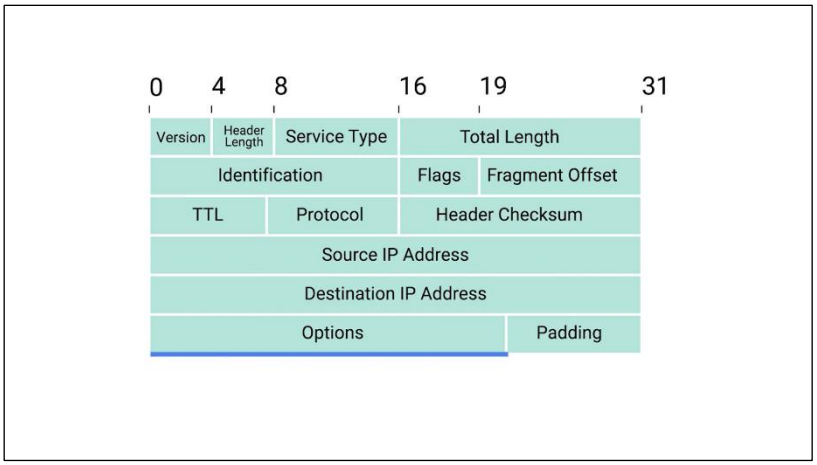
It functions very much like the Ethernet checksum field we discussed in the last module. Since the TTL field has to be recomputed at every router that a datagram touches, the checksum field necessarily changes, too.

After all of that, we finally get to two very important fields, the source and destination IP address fields.

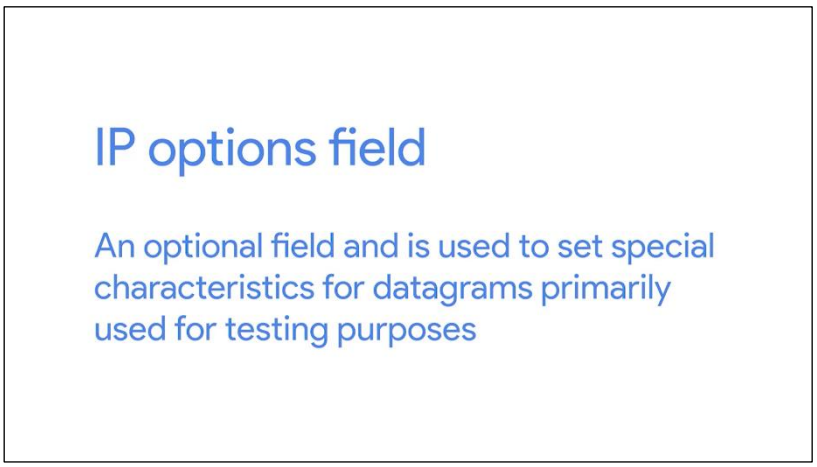


Remember that an IP address is a 32 bit number so, it should come as no surprise that these fields are each 32 bits long.

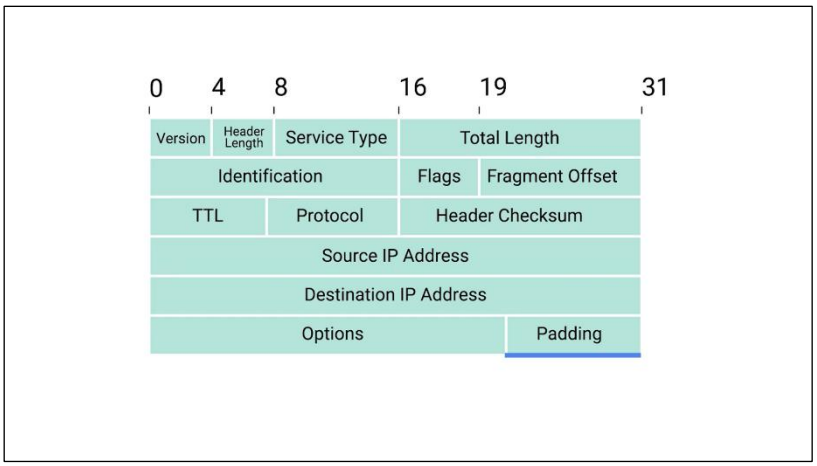
Up next, we have the IP options field.



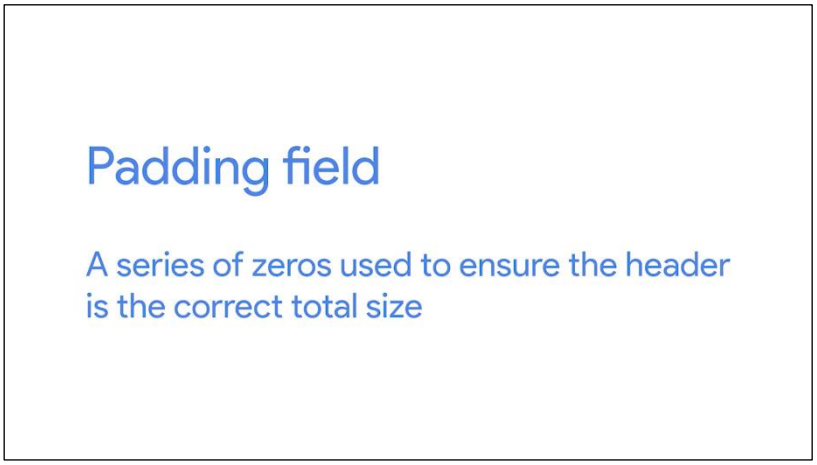
This is an optional field and is used to set special characteristics for datagrams primarily used for testing purposes.



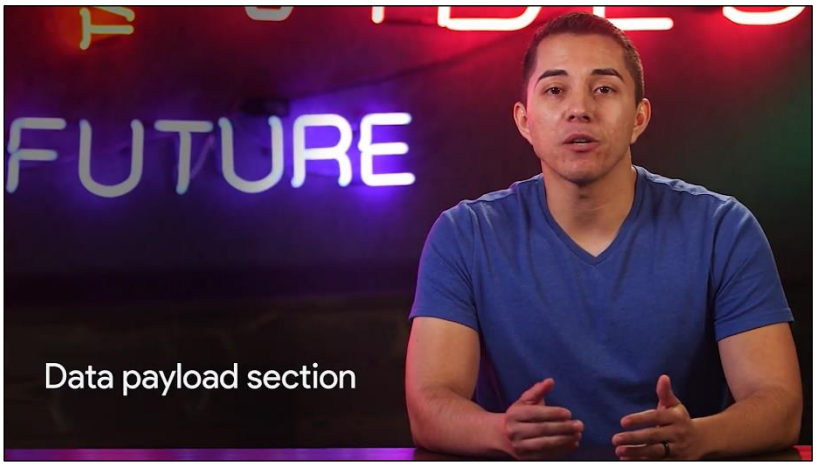
The IP options field is usually followed by a padding field.



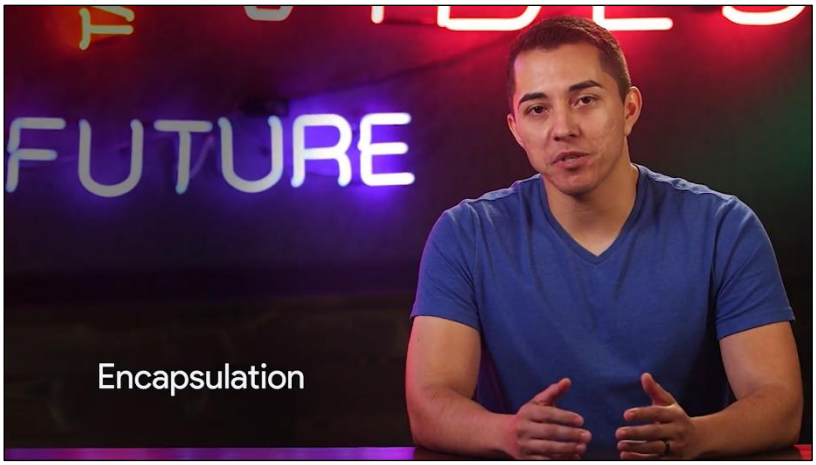
Since the IP options field is both optional and variable in length, the padding field is just a series of zeros used to ensure the header is the correct total size.



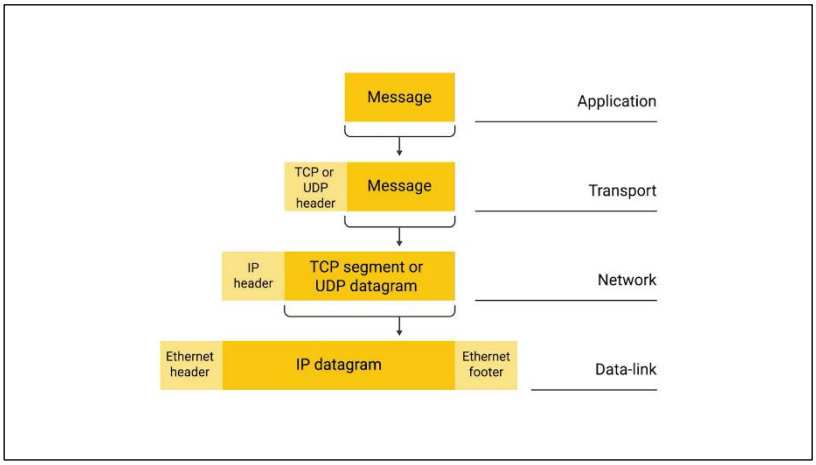
Now that you know about all of the parts of an IP datagram, you might wonder how this relates to what we've learned so far. You might remember that in our breakdown of an Ethernet frame, we mentioned a section we described as the data payload section.



This is exactly what the IP datagram is, and this process is known as encapsulation.



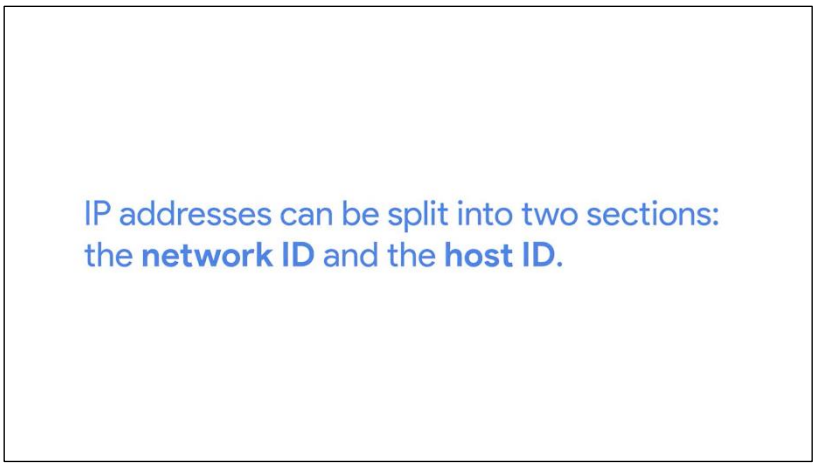
The entire contents of an IP datagram are encapsulated as the payload of an Ethernet frame. You might have picked up on the fact that our IP datagram also has a payload section.



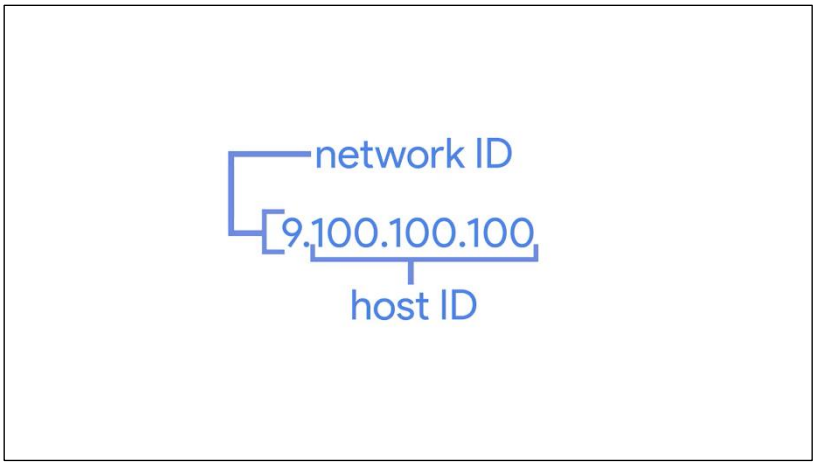
The contents of this payload are the entirety of a TCP or UDP packet which we'll cover later. Hopefully, this helps you better understand why we talk about networking in terms of layers. Each layer is needed for the one above it.

2.2.4 IP address classes

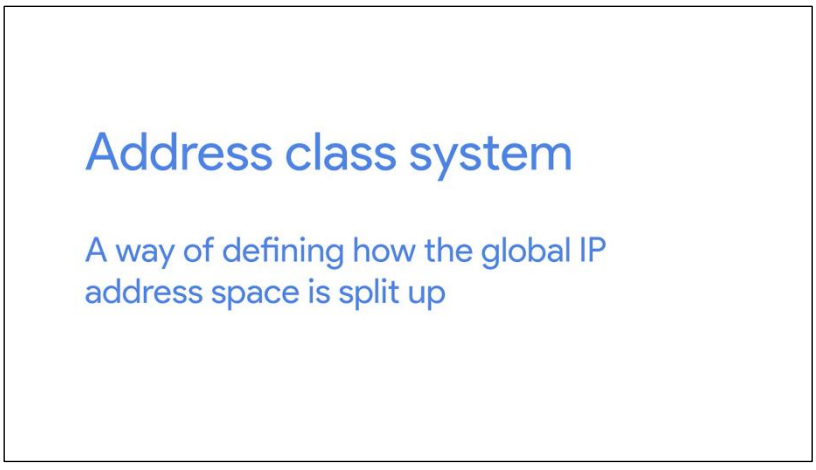
IP addresses can be split into two sections, the network ID, and the host ID.



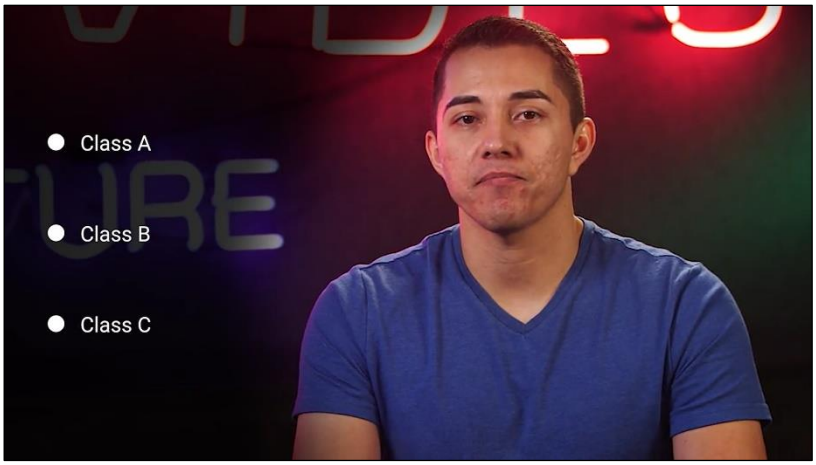
Earlier, we mentioned that IBM owns all IP addresses that having a nine as the value of the first octet in an IP address. If we take an example IP address of 9.100.100.100, the network ID would be the first octet, and the host ID, would be the second, third and fourth octets.



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, as you might have guessed, are those where the first three octets are used for the network ID, and only the final octet is used for the host ID.

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-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.

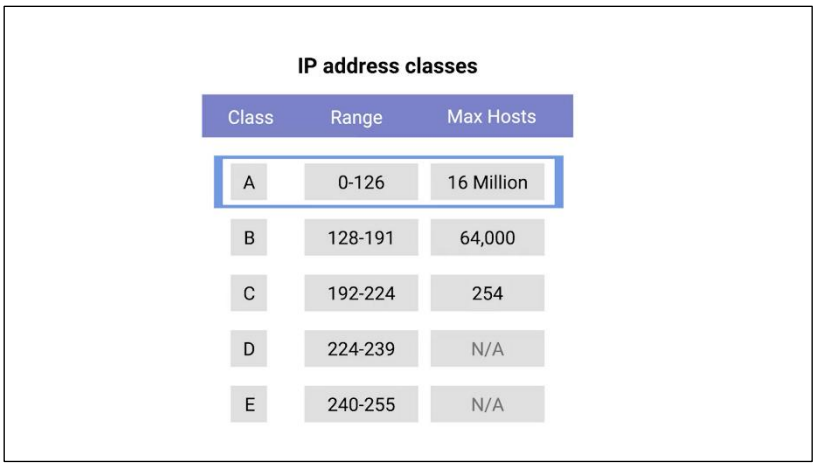
IP address classes

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

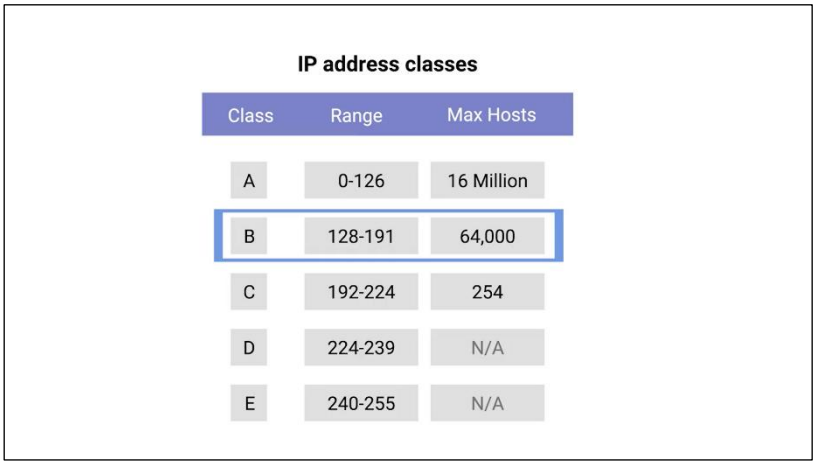
You can also tell exactly what address class an IP address belongs to just by looking at it. If the very first bit of an IP address is a zero, it belongs to a class A network, if the first bits are one, zero, it belongs to a class B network. Finally, if the first bits are 110, it belongs to a class C network.

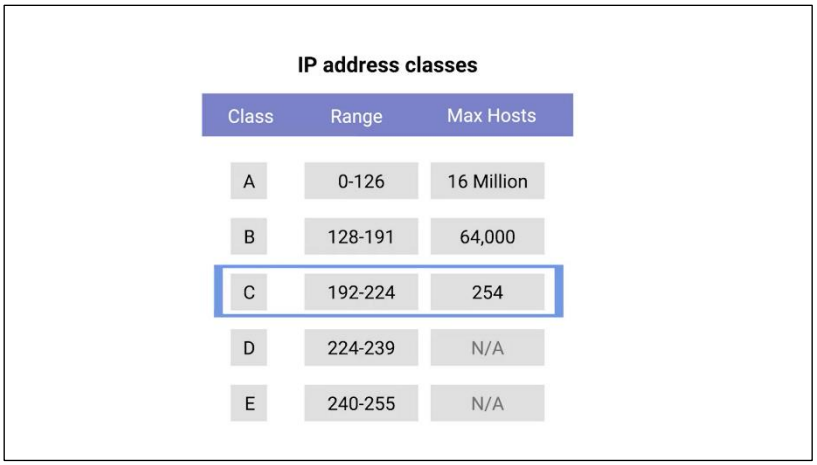
Since humans aren't great at thinking in binary, it's good to know that this also translates nicely to how these addresses are represented in dotted decimal notation. You might remember that each octet in an IP address is eight bits, which means each octet can take a value between zero and 255.

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.



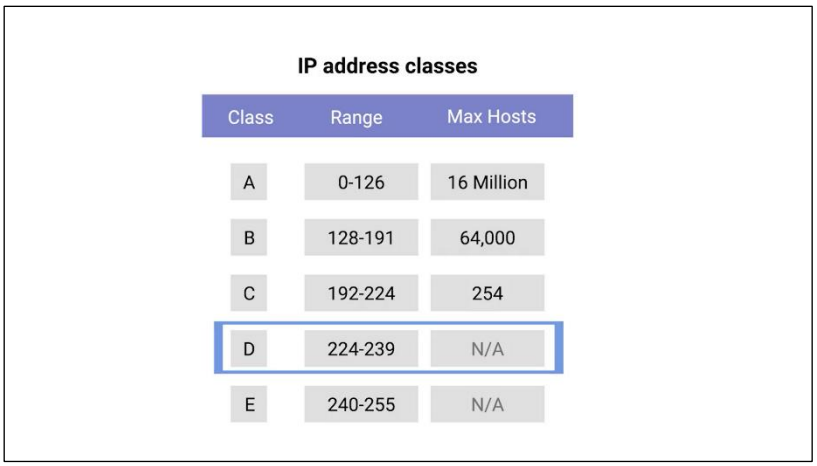
Similarly, class B addresses are restricted to those that begin with the first octet value of 128 through 191, and class C addresses begin with the first octet value of 192 through to 223.



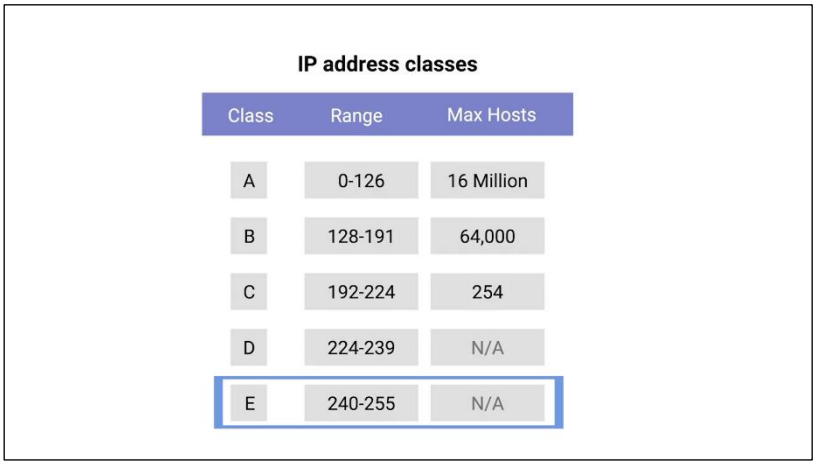


You might notice that this doesn't cover every possible IP address. That's because there are two other IP address classes, but they're not quite as important to understand.

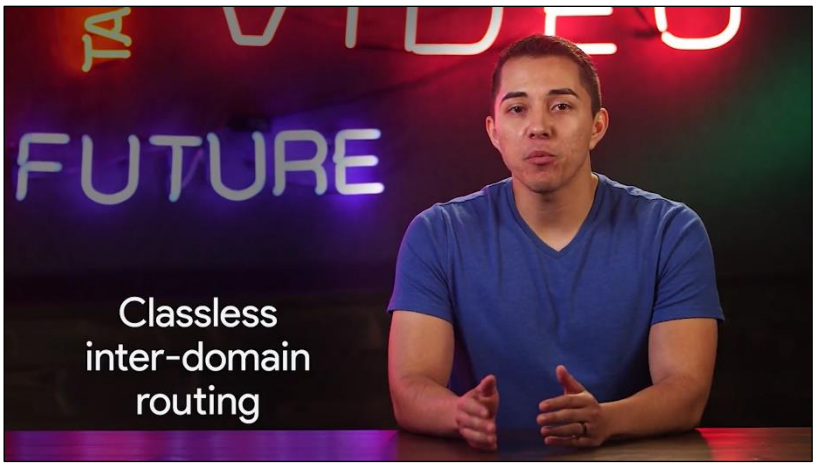
Class D addresses always begin with the bits 1110, and are used for multicasting, which is how a single IP datagram can be sent to an entire network at once. These addresses begin with decimal values between 224 and 239.



Lastly, class E 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.

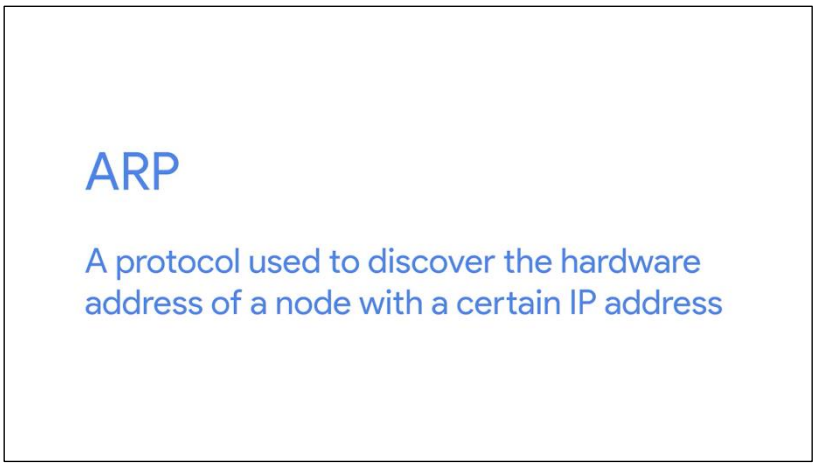


But the address class system is still in place in many ways, and is important to understand for anyone looking for a well rounded networking education, and you know we're all about that. So don't worry, we'll be covering CIDR in a future lesson.

2.2.5 Address resolution protocol

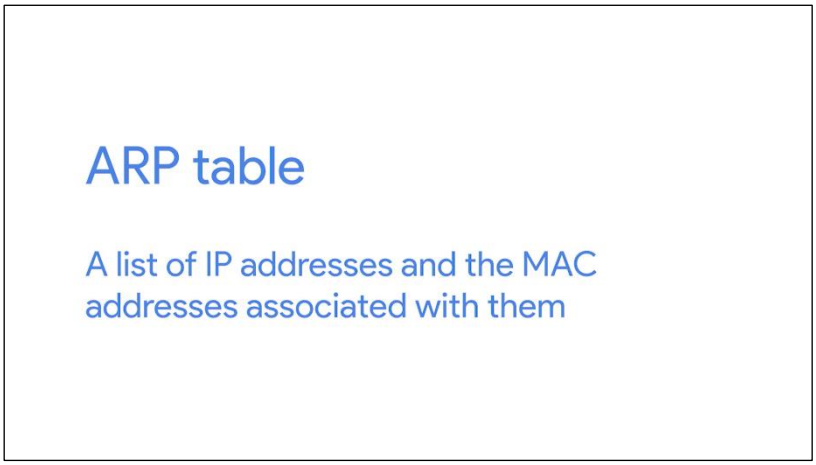
Congrats! You now understand how both MAC addresses are used at the Data Link Layer and how IP addresses are used at the network layer. Now we need to discuss how these two separate addresses types relate to each other.

This is where Address Resolution Protocol or ARP comes into play. ARP is a protocol used to discover the hardware address of a node with a certain IP address.

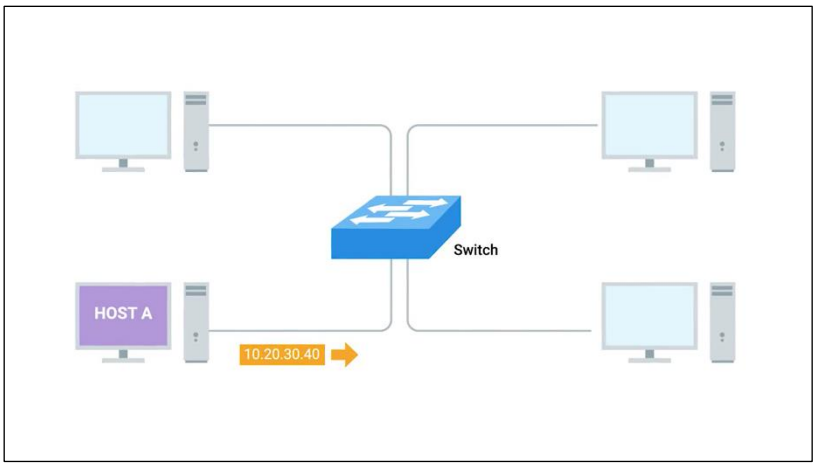


Once an IP datagram has been fully formed, it needs to be encapsulated inside an Ethernet frame. This means, that the transmitting device needs a destination MAC address to complete the Ethernet frame header.

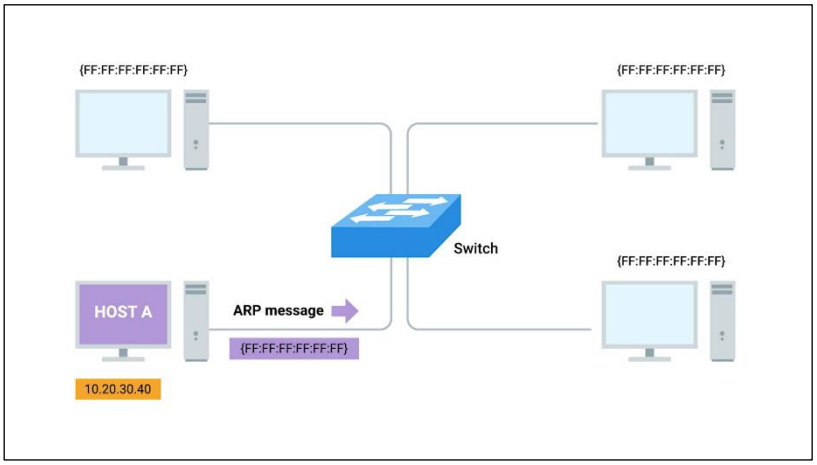
Almost all network connected devices while retaining local ARP table. In ARP table is just a list of IP addresses and the MAC addresses associated with them.



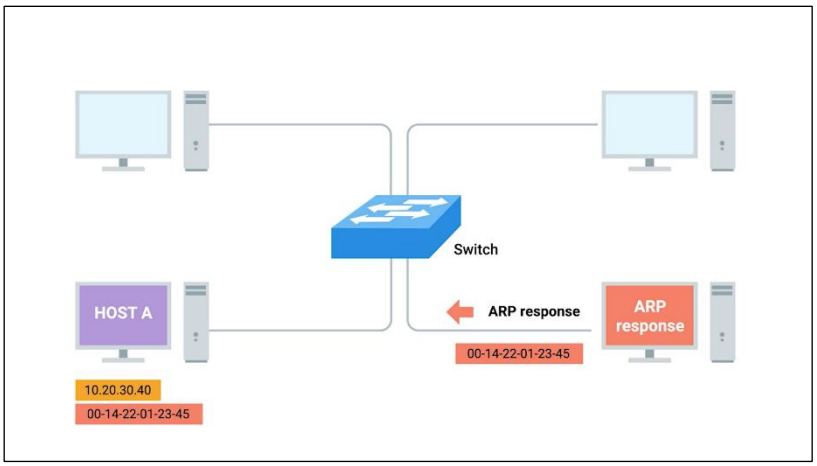
Let's say we want to send some data to the IP address 10.20.30.40. It might be the case that this destination doesn't have an entry in the ARP table.



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

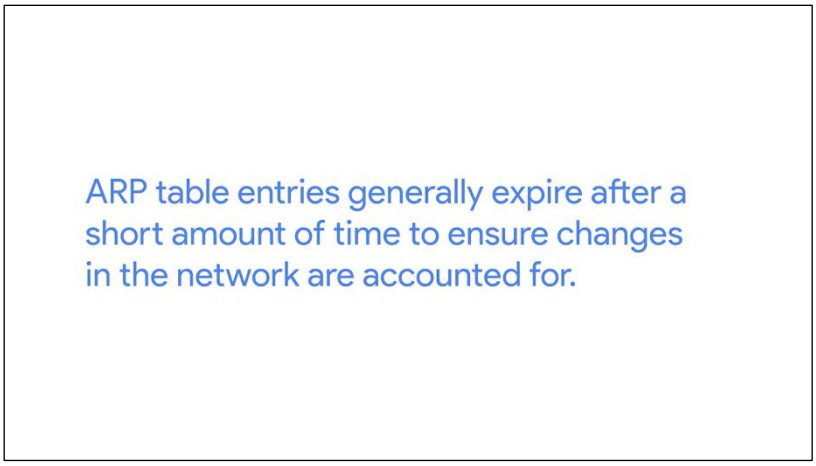


These kinds of broadcast ARP messages are delivered to all computers on the local network. When the network interface that's been assigned an IP of 10.20.30.40 receives this ARP broadcast, it sends back what's known as an ARP response. This response message will contain the MAC address for the network 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.

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

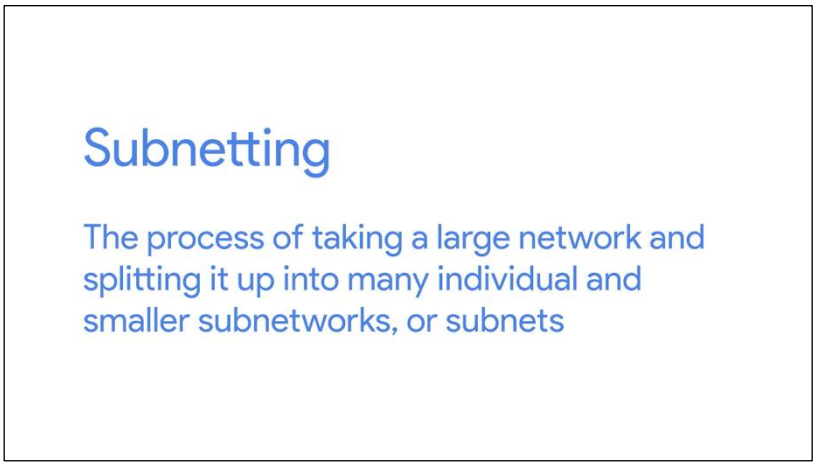


So don't expect it to stick around, the way I expect you to stick around for the next ungraded quiz.

2.3 Subnetting

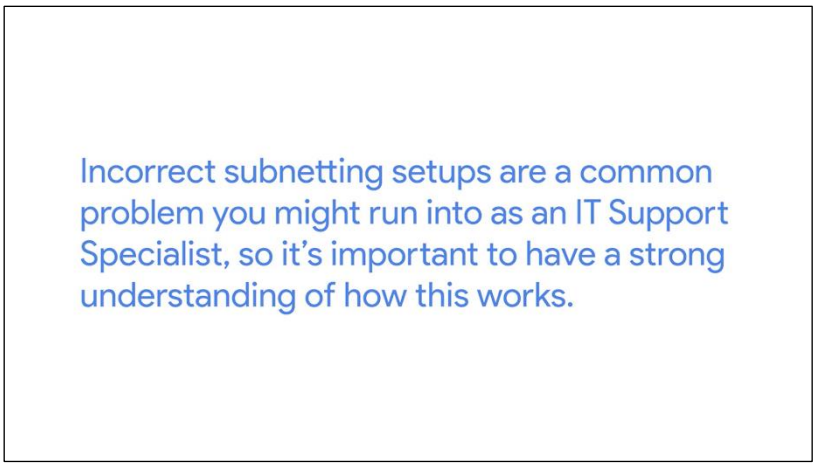
2.3.1 Subnetting

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.



By the end of this lesson, 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. You'll also be able to discuss how a technique known as CIDR allows for even more flexibility than plain subnetting. Lastly, you'll be able to apply some basic binary math techniques to better understand how all of this works.

Incorrect subnetting setups are a common problem you might run into as an IT support specialist, so it's important to have a strong understanding of how this works.

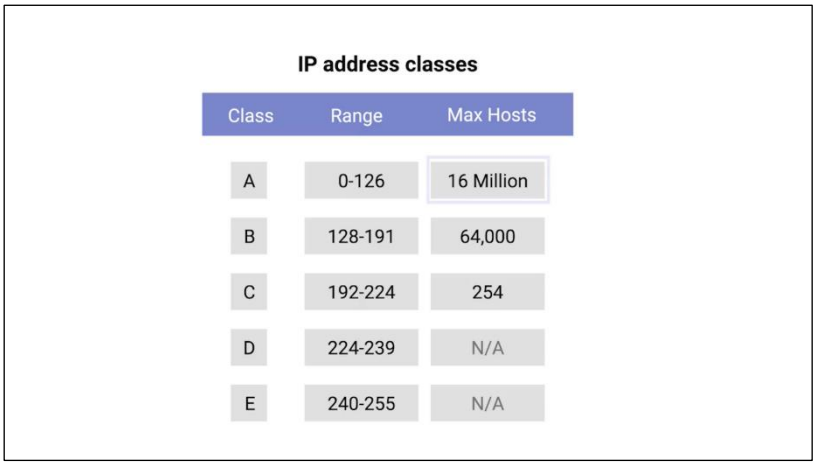


That's a lot, so let's dive in. As you might remember, from the last lesson, address classes give us a way to break the total global IP space into discrete networks. If you want to communicate with the IP address 9.100.100.100, core routers on the Internet know that this IP belongs to the 9.0.0.0 class 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.

2.3.2 Subnet masks

So far, we've learned about network IDs, which are used to identify networks, and host IDs, which are used to identify individual hosts. If we want to split things up even further, and we do, we'll need to introduce a third concept, the subnet ID.

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. 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. Warning: dense material ahead. We're about to get into some tough material, but it's super important to properly understand how subnet masks work because they're so frequently misunderstood. Subnet masks are often glossed over as magic numbers. People just memorize some of the common ones without fully understanding what's going on behind the scenes. In this course, we're really trying to ensure that you lead with a well-rounded networking education. So, even though subnet masks can seem tricky at first, stick with it, and you'll get the hang of it in no time. Just know that in the next video, we'll be covering some additional basics of binary math. Feel free to watch this video a second or third time after reviewing the material. Go at your own pace, and you'll get there in the perfect amount of time. 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. 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. 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. That was a lot. Make sure to go back and watch this video again if you need a refresher, or if you're a total wiz, you can move on to the next video on basic binary math. I'll see you there or maybe here.

2.3.3 Basic binary maths

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. 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. 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. 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. 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. 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. Now, you might be wondering why we've covered all of this. No, it's not to confuse you. It's all really to help explain subnet masks a bit more. 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 favorite IP address 9.100.100.100 and our favorite 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. 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. I bet you never thought you'd have a favorite IP address or subnet but that's what happens in the wonderful world of basic binary math.

2.3.4 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. 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. 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. CIDR is where we get this shorthand slash notation that we discussed in the earlier video on subnetting. 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. Take a second and lock that into your memory. Then when you're ready, we have a short ungraded quiz for you before we move on to routing in the next lesson.