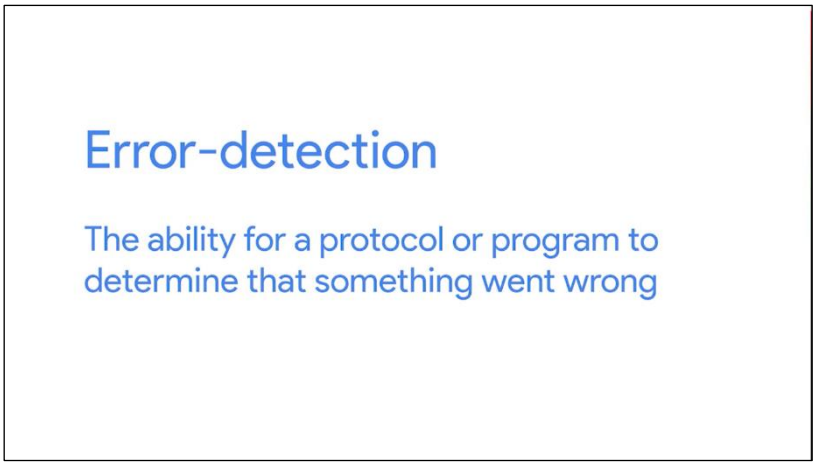
GOOGLE IT SUPPORT SPECIALIZATION
COURSE 2: COMPUTER NETWORKING
05 CONNECTING TO THE INTERNET

Table des matières

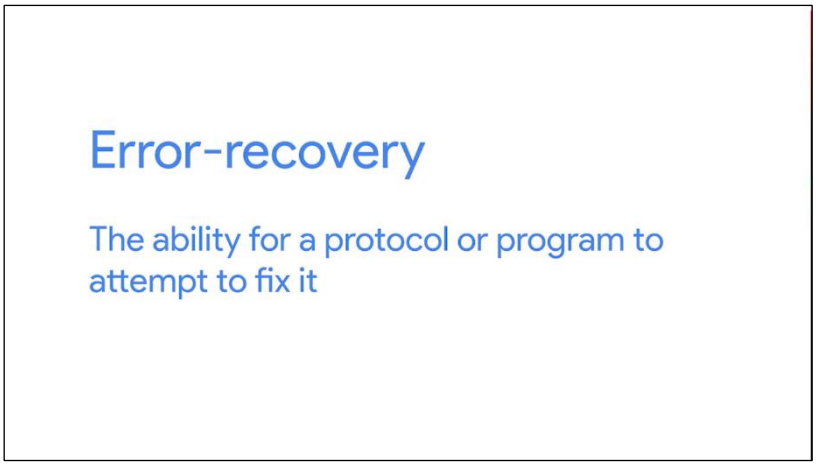
1 Introduction to networking
2 The network layer
3 Transport layer and application layer
4 Networking services
5 Connecting to the internet
6 Troubleshooting and the future of networking
6.1 Introduction to troubleshooting and the future of networking
6.1.1 Introduction to troubleshooting and the future of networking

Welcome back. As you've seen, 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. No surprise there. 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, is the ability for a protocol or program to determine that something went wrong.



Error-recovery is the ability for a protocol or program to attempt to fix it.



For example, you might remember that 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.

In this module, you'll learn about the most common techniques and tools you use as an IT support specialist when troubleshooting network issues. By the end of this module, 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.

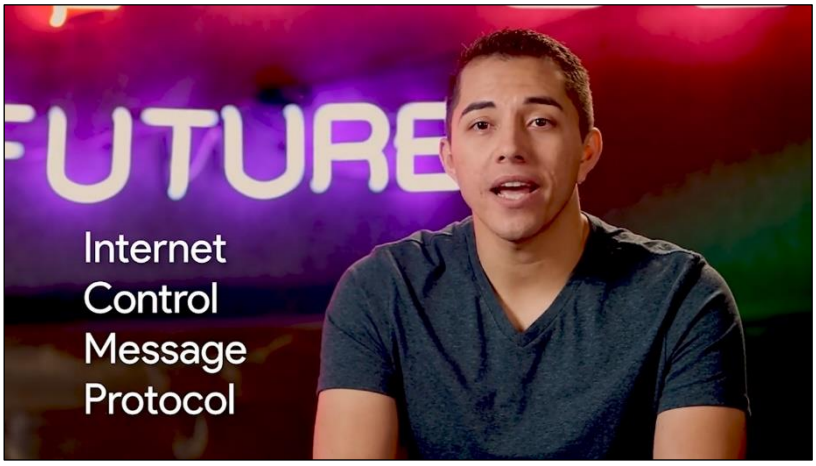
And finally, at the end of this module, we'll cover some concepts that are super important to the future of networking: the cloud and IPv 6.

6.2 Verifying connectivity

6.2.1 Ping internet control message protocol

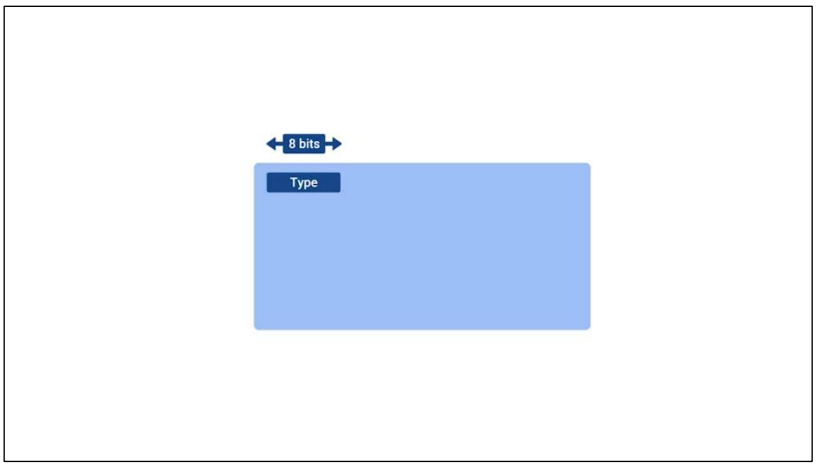
When network problems come up, the most common issue you'll run into is the inability to establish a connection to something. It could be a server you can't reach at all or a website that isn't loading. Maybe you can only reach your resource on your LAN and can't connect to anything on the internet.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

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.

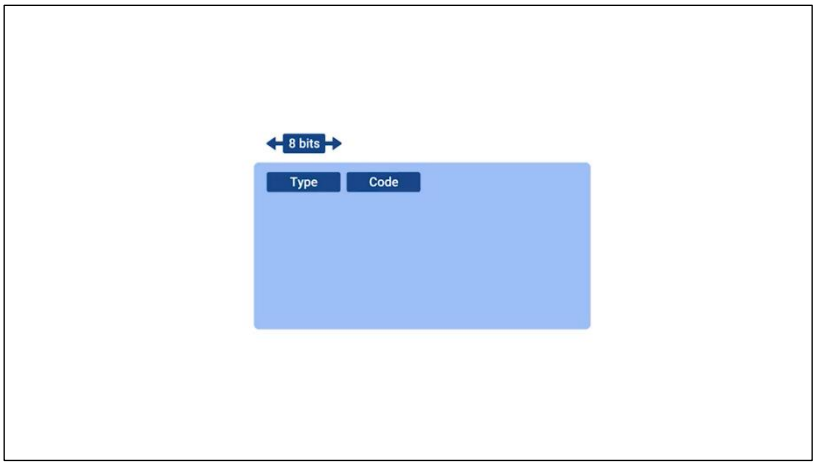


ICMP is mainly used by router or remote hosts to communicate while transmission has failed back to the origin of the transmission. The makeup of an ICMP packet is pretty simple. It has a header with a few fields and a data section that's used by host to figure out which of their transmissions generated the error.

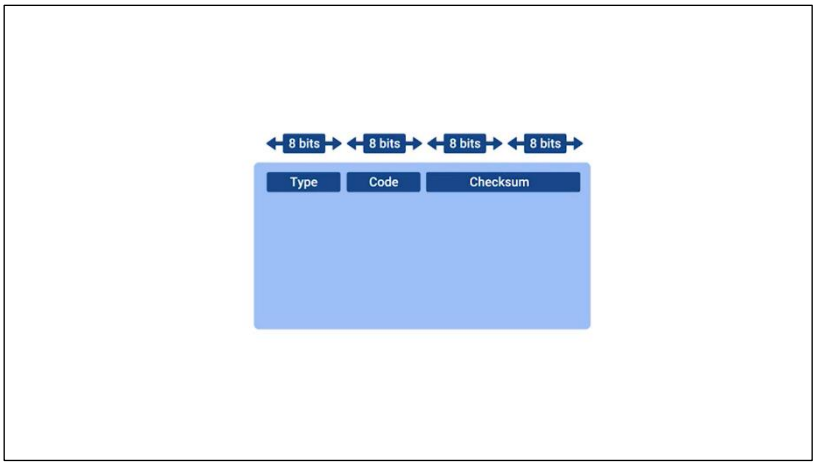
The first field is the type field, eight bits long which specifies what type of message is being delivered. Some examples are destination unreachable or time exceeded.



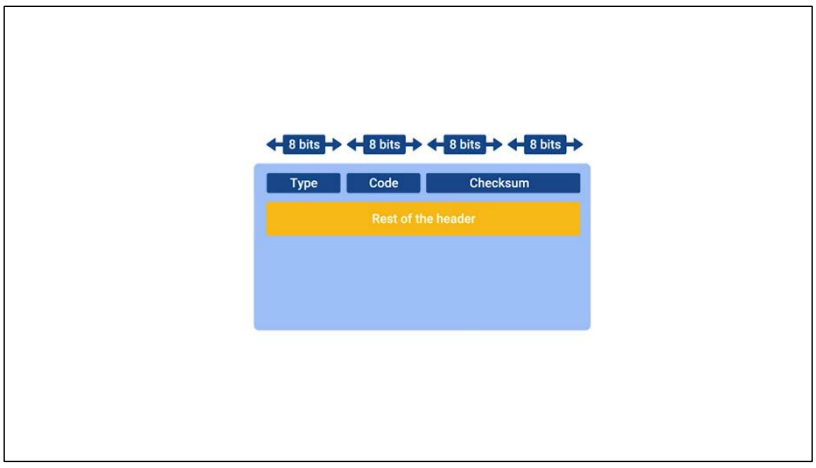
Immediately after this is the code field which indicates a more specific reason for the message than just the type. For example, of the destination unreachable type, there are individual codes for things like destination network unreachable and destination port unreachable.



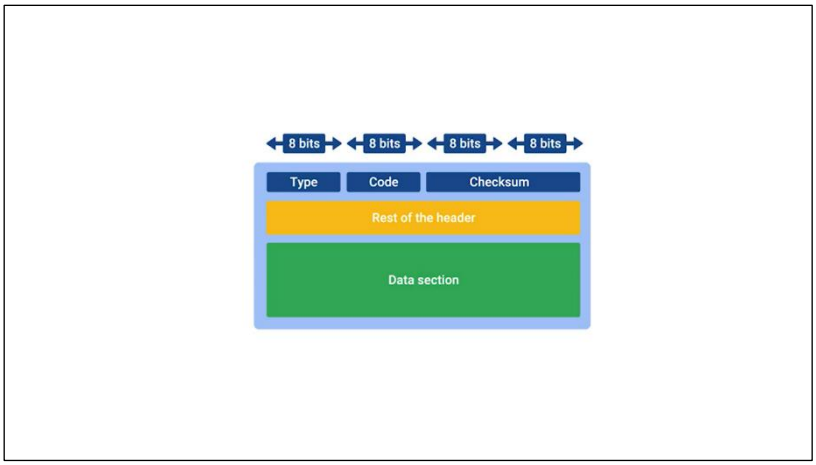
After this is a 16 bit checksum that works like every other checksum field we've covered so far.



Next up is a 32 bit field with an uninspired name, Rest of header. You think they could come up with something a bit more interesting but I can't really think of anything good. So, who am I to judge? Anyway, this field is optionally used by some of the specific types and codes to send more data.

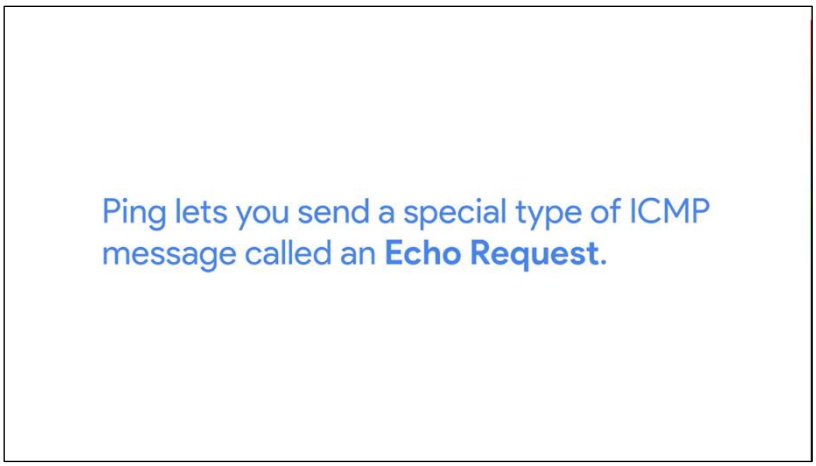


After this is the data payload for an ICMP packet. The payload for an ICMP packet exists entirely so that the recipient of the message knows which of their transmissions caused the error being reported. It contains the entire IP header and the first eight bytes of the data payload section of the offending packet.

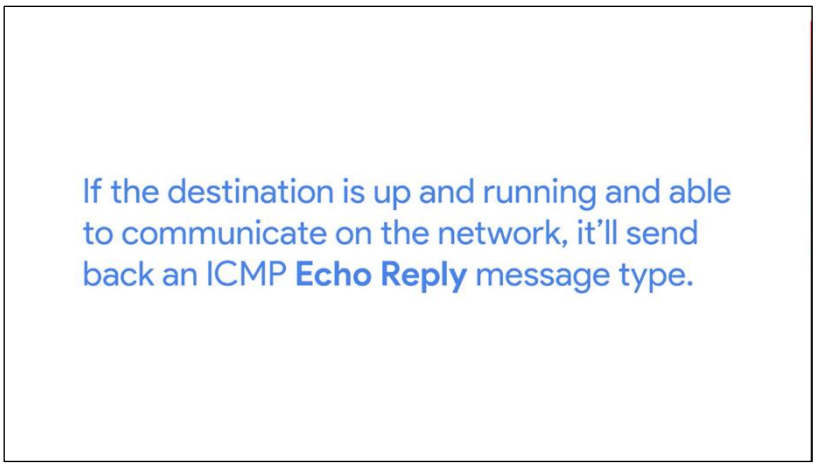


ICMP wasn't really developed for humans to interact with. The point is so that these sorts of error messages can be delivered between networked computers automatically. But, there's also a specific tool and two message types that are very useful to human operators.

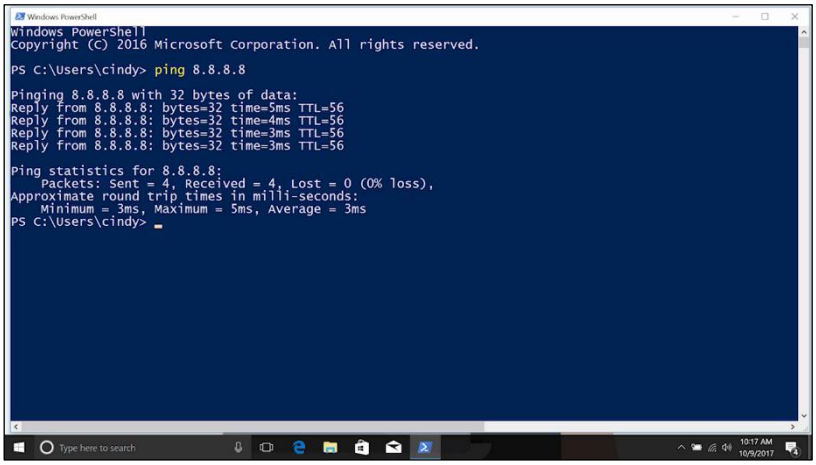
This tool is called ping. Some version of it exist on just about every operating system, and has for a very long time. Ping is a super simple program and the basics are the same no matter which operating system you're using. Ping lets you send a special type of ICMP message called an Echo Request.



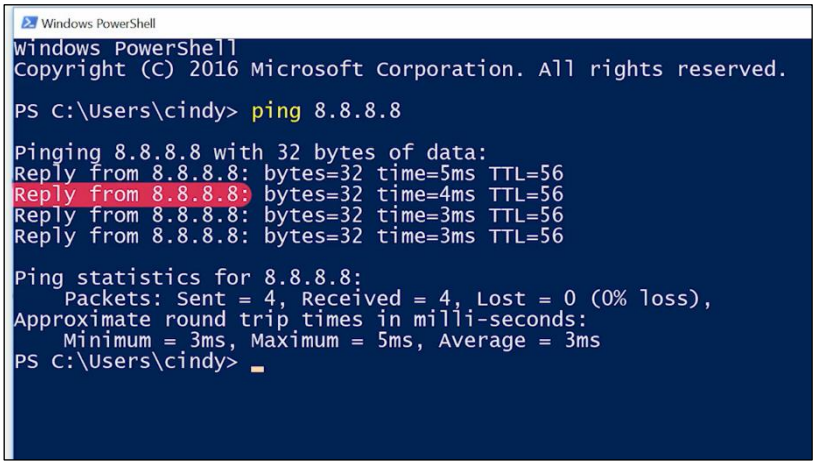
An ICMP echo request essentially just ask the destination, "Hey, are you there?" If the destination is up and running and able to communicate on the network, it will send back an ICMP echo reply message type.

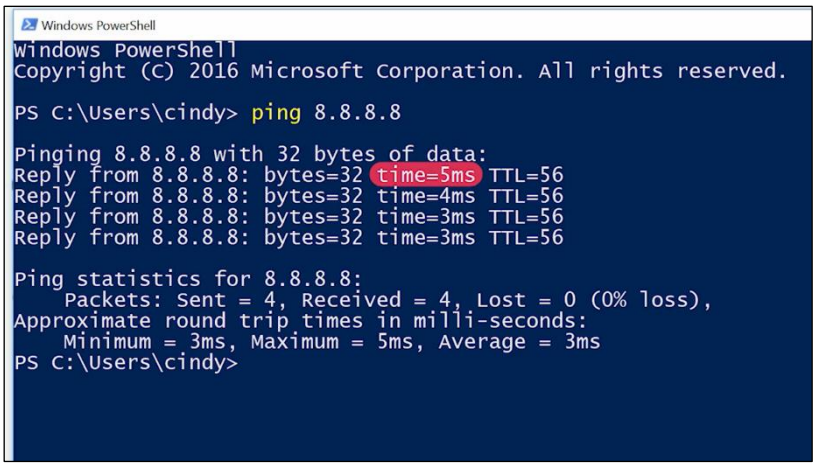


You can invoke the ping command from the command line of any modern operating system. In its most basic use, you just type ping and a destination IP or a fully qualified domain name. If you don't know how to use a command line in an operating system, don't worry, you will soon. We'll cover that in another course.

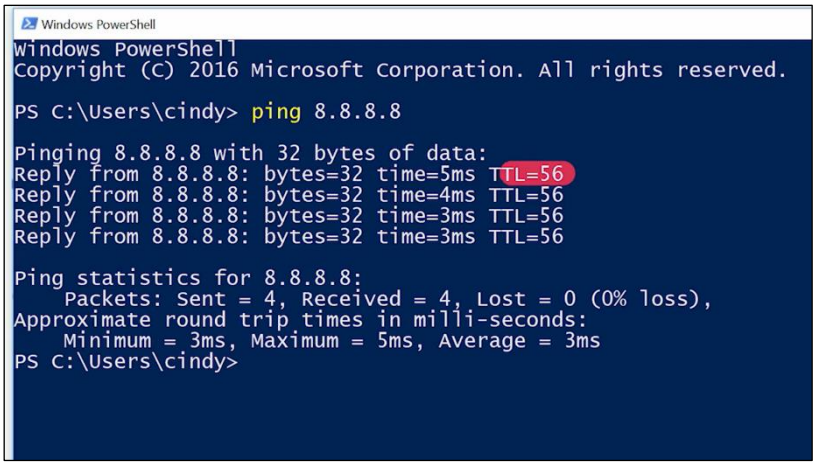


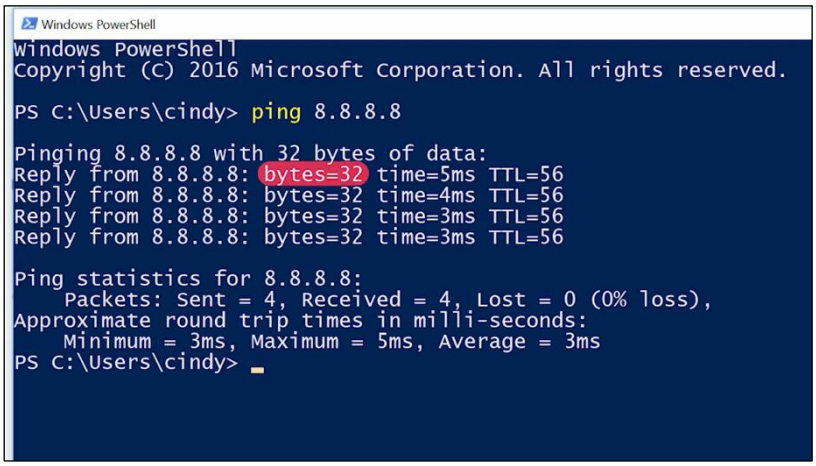
Output of the ping command is very similar across each of the different operating systems. Every line of output will generally display the address sending the ICMP echo reply, and how long it took for the round trip communications.



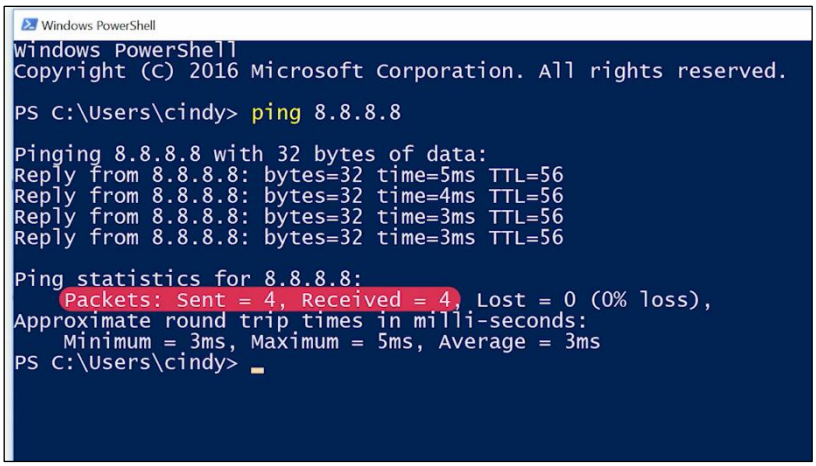


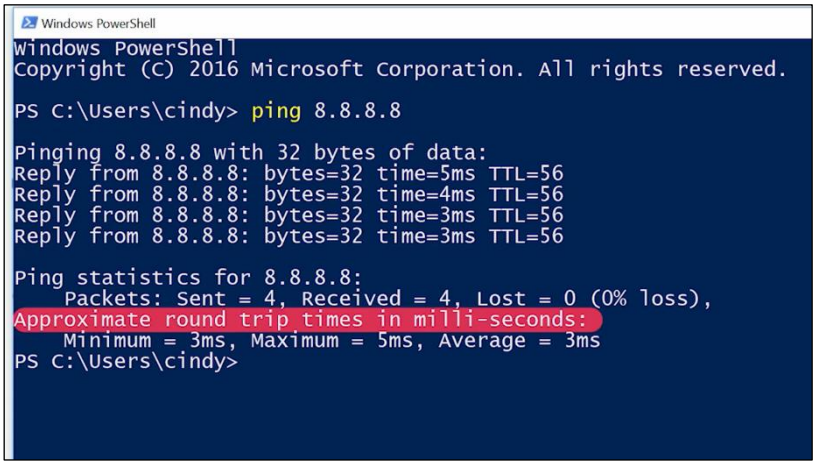
It will also have the TTL remaining and how large the ICMP message is in bytes.



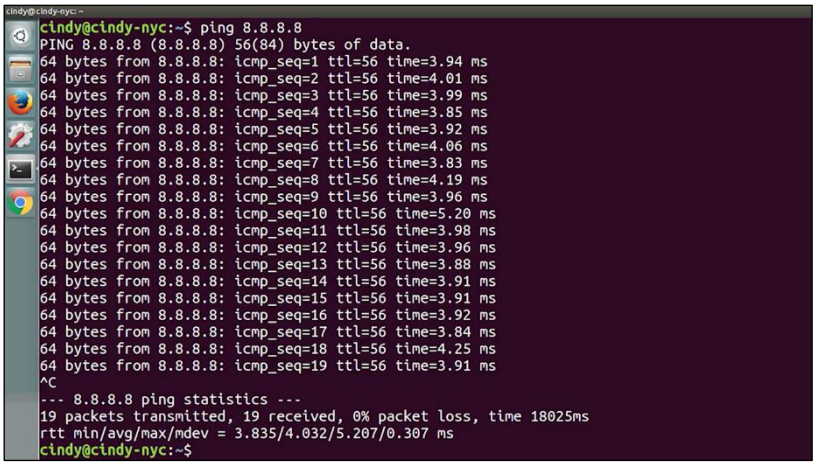


Once the command ends, there will also be some statistics displayed like percentage of packets transmitted and received, the average round trip time, and a couple of other things like that.

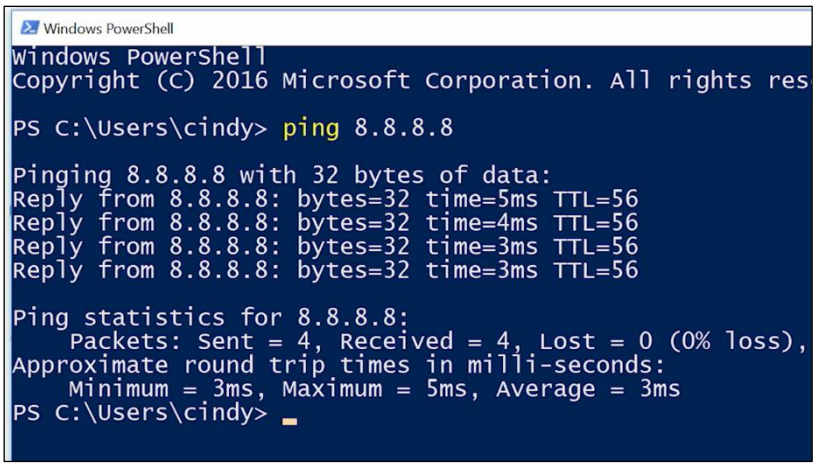




On Linux and Mac OS, the ping command will run until it's interrupted by an end user sending an interrupt event. They do this by pressing the control key and the C key at the same time.



On Windows, ping defaults to only sending four echo requests.

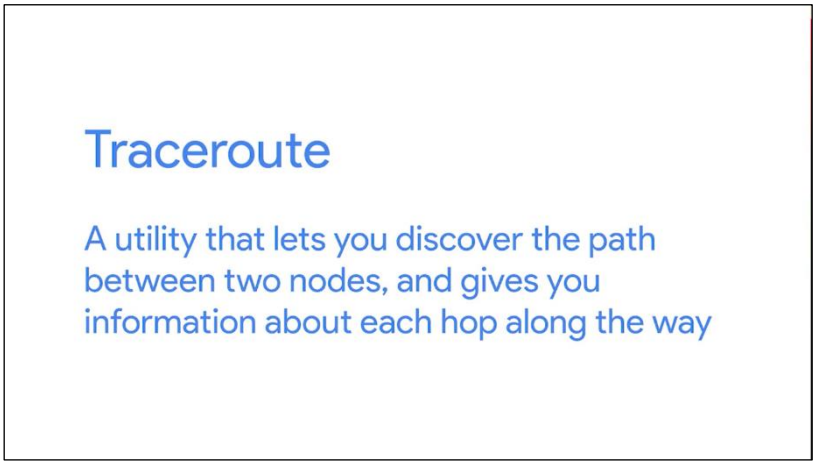


In all environments, ping supports a number of command line flags that let you change its behavior like the number of echo requests to send, how large they should be, and how quickly they should be sent. Check out the documentation for your operating system to learn a little bit more.

6.2.2 Traceroute

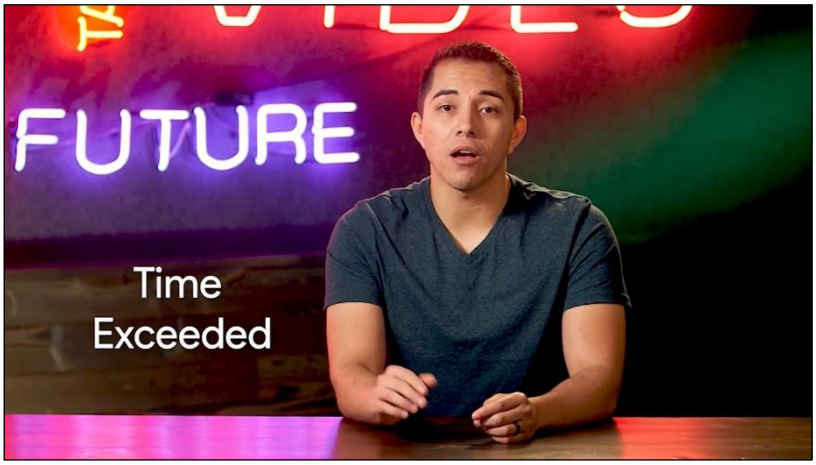
With ping, you now have a way to determine if you can reach a certain computer from another one. You can also understand the general quality of the connection. But communications across networks, especially across the Internet usually, cross lots of intermediary nodes. Sometimes, you need a way to determine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

Traceroute to the rescue. Traceroute is an awesome utility that lets you discover the paths between two nodes, and gives you information about each hop along the way.



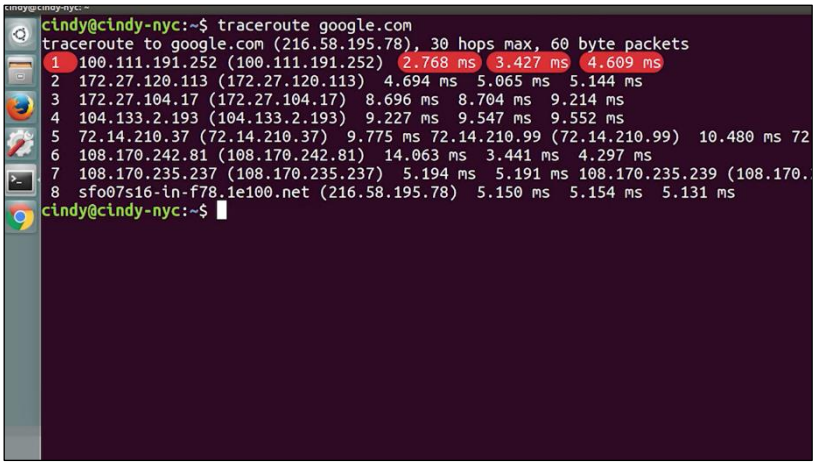
The way traceroute works, is through a clever manipulation technique of the TTL field at the IP level. We learned earlier that the TTL field is decremented by one, by every router that forwards the packet.

When the TTL field reaches zero, the packet is discarded and an ICMP Time Exceeded message is sent back to the originating host.

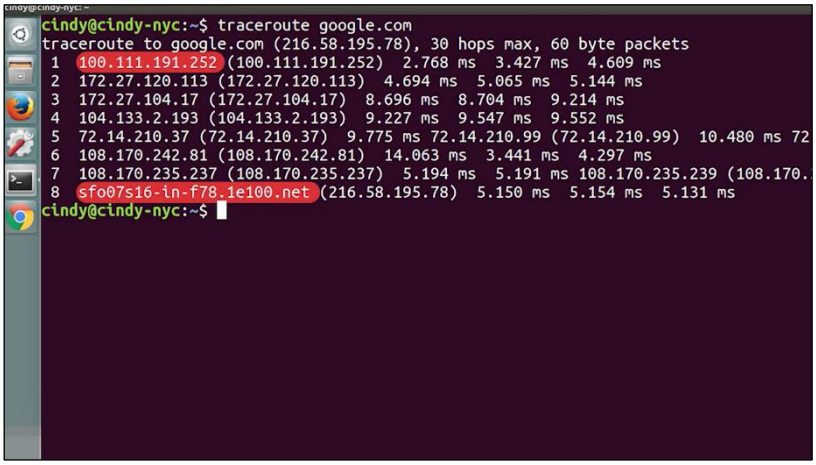


Traceroute uses the TTL field by first setting it to one for the first packet, then two for the second, three for the third and so on. By doing this clever little action, traceroute makes sure that the very first packet sent will be discarded by the first router hop. This results in an ICMP Time Exceeded 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 pretty simple. On each line, you'll see the number of the hop and the round trip time for all three packets.

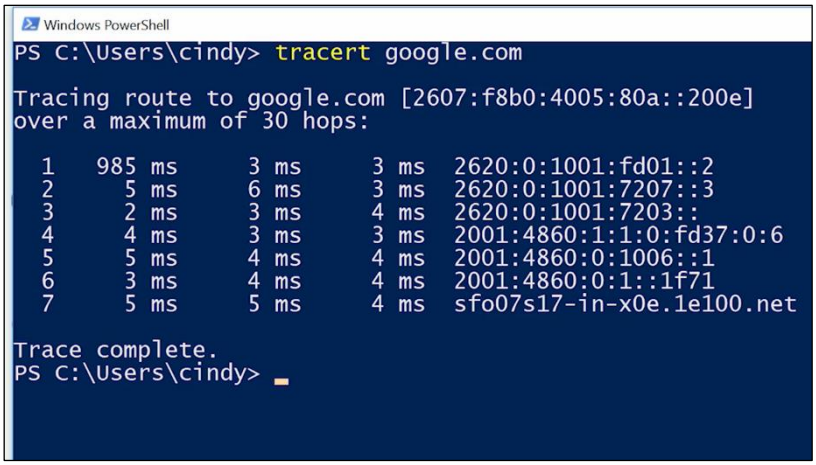


You will also see the IP of the device at each hop, and a host name if traceroute can resolve one.



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

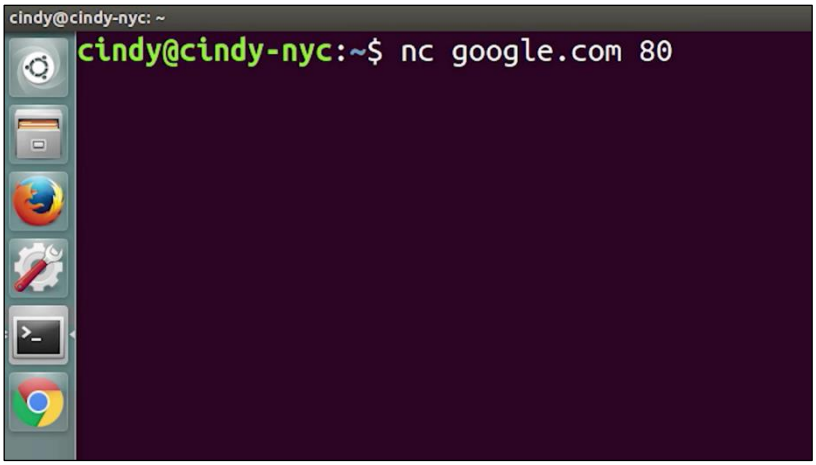
aggregate data about the traceroute. You can compare this with pathping, which runs for 50 seconds and then displays the final aggregate data all at once.

6.2.3 Testing port connectivity

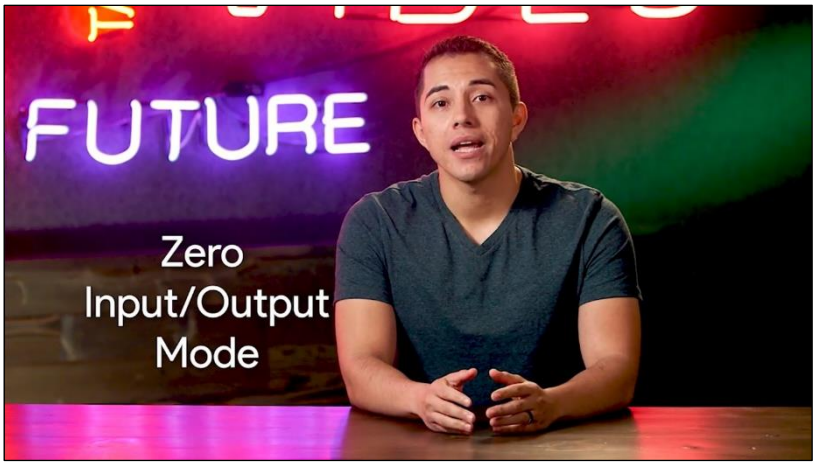
We've covered a bunch of ways to test connectivity between machines at the network layer. But sometimes, you need to know if things are working at the transport layer. For this, there are two super powerful tools at your disposal. Netcat on Linux and Mac OS and Test-NetConnection on Windows.

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.

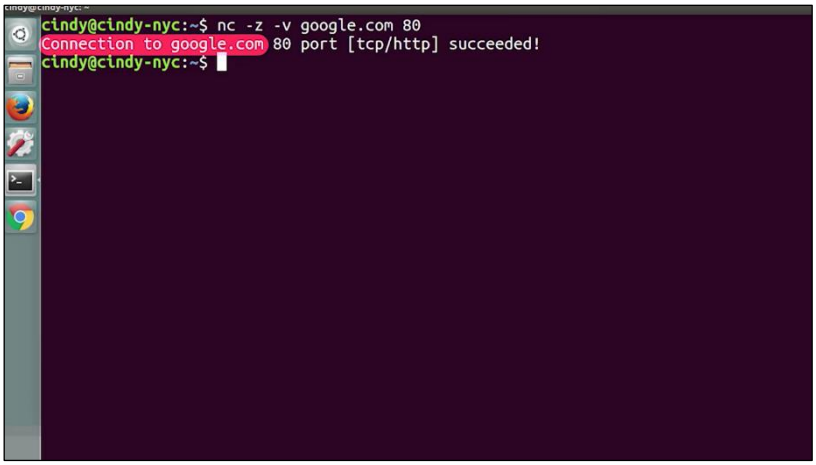


If you're really only curious about the status of a report, you can issue the command, with a -Z flag, which stands for Zero Input/Output Mode.

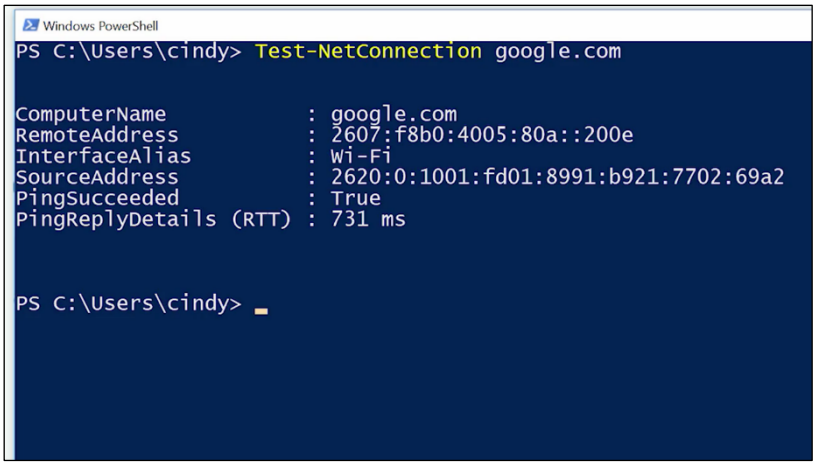


A -V flag, which stands for Verbose, is also useful in this scenario. This makes the commands output useful to human eyes as opposed to non-verbose output, which is best for usage in scripts.

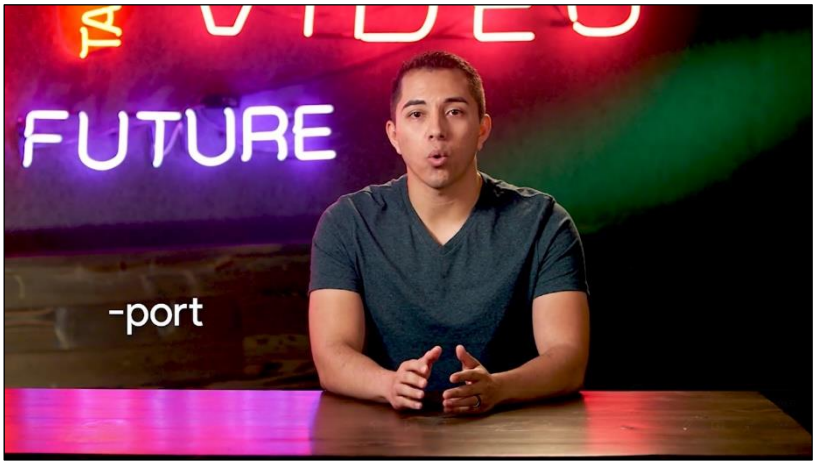
Side note, verbose basically means talking too much. So, while I bet you want to throw up a flag on me and my jabbering, we still have lots to get through. Okay, so by issuing the Netcat command with the -Z and -V flags, the command's output will simply tell you if a connection to the port in question 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.

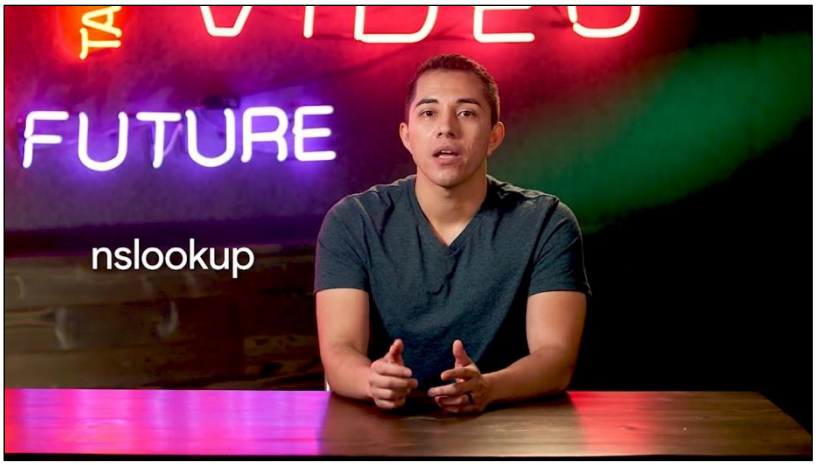


It's important to call out that both Netcat and Test-NetConnection are way more powerful than the brief port connectivity examples we've covered here. In fact, they're such complex tools that covering all of their functionality would be too much for one video. You should read up about all of the other things these super powerful tools can do. We've provided a few in the supplementary readings. And after you've had a chance to read through the material, we'll give you a short quiz.

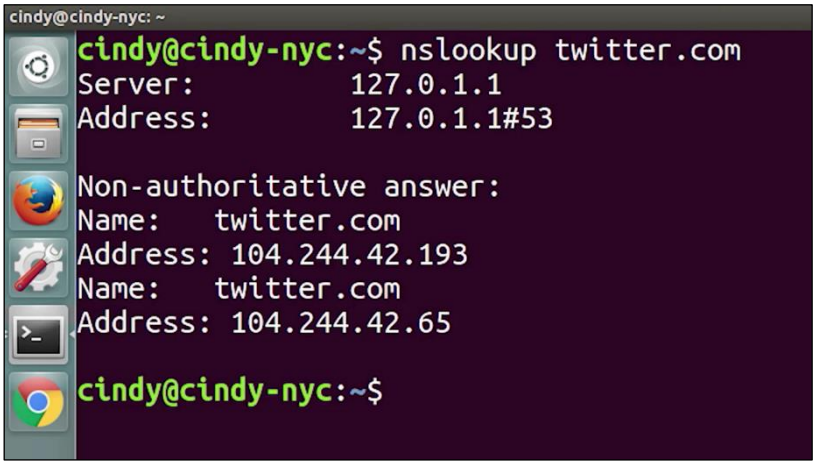
6.3 Digging into DNS

6.3.1 Name resolution tools

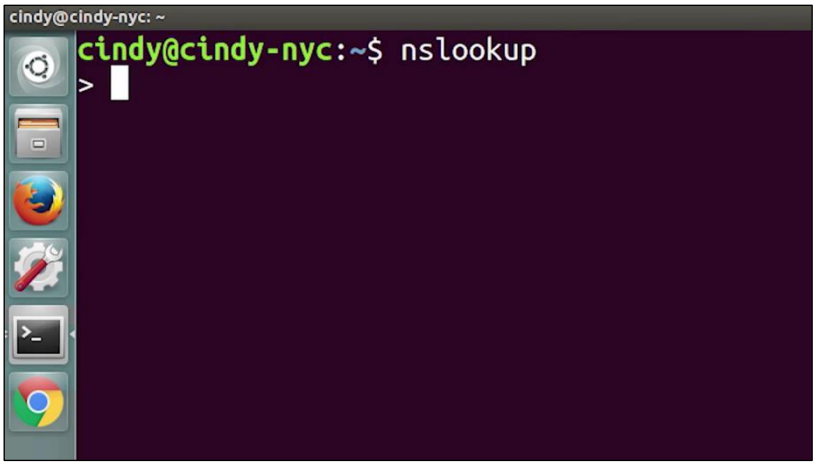
Name resolution is an important part of how the Internet works. Most of the time, your operating system handles all look ups for you. But as an IT support specialist, sometimes it can be useful to run these queries yourself, so you can see exactly what's happening behind the scenes. Luckily, there are lots of different command line tools out there to help you with this. The most common tool is known as nslookup. And it's available on all three of the operating systems we've been discussing, Linux, Mac, and Windows.



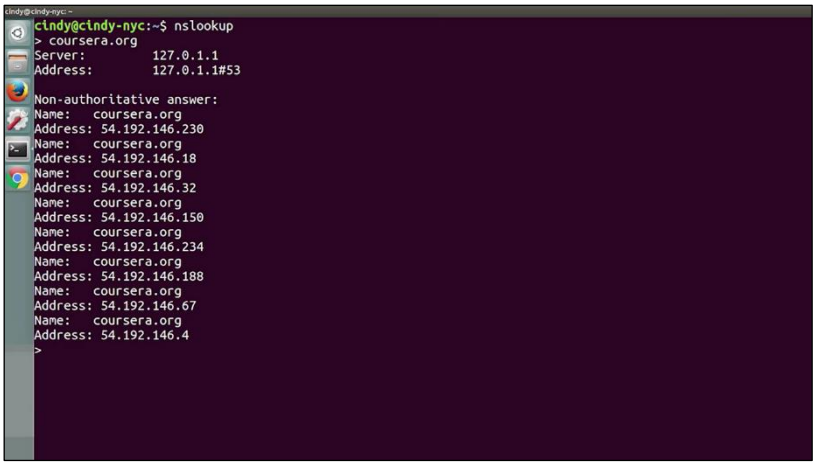
A basic use of nslookup is pretty simple. You execute the nslookup command with the host name following it. And the output displays what server was used to perform the request and the resolution result. Let's say you needed to know the IP address for a twitter.com. You would just enter nslookup twitter.com and the A record would be returned.



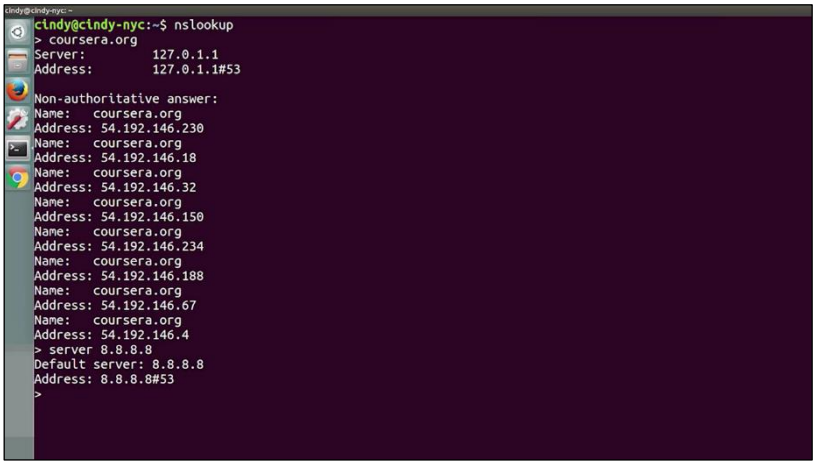
Nslookup is way more powerful than just that. It includes an interactive mode that lets you set additional options and run lots of queries in a row.



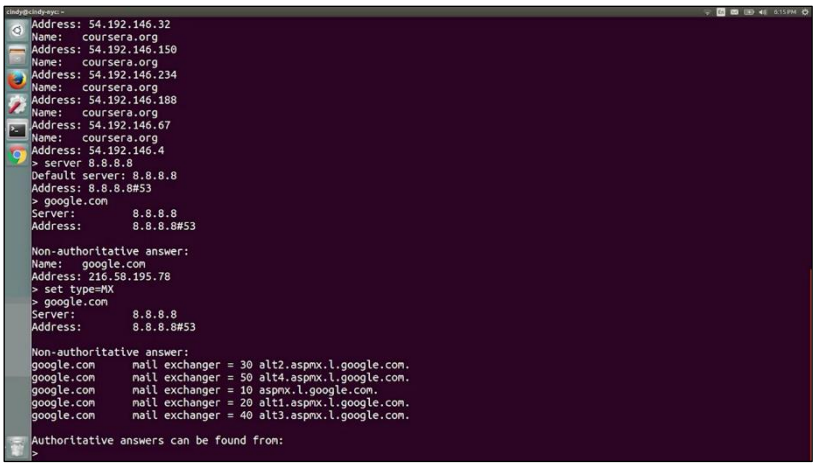
To start an interactive nslookup session, you just enter nslookup, without any hostname following it. You should see an angle bracket acting as your prompt. From interactive mode, you can make lots of requests in a row. You can also perform some extra configuration to help with more in-depth trouble 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.



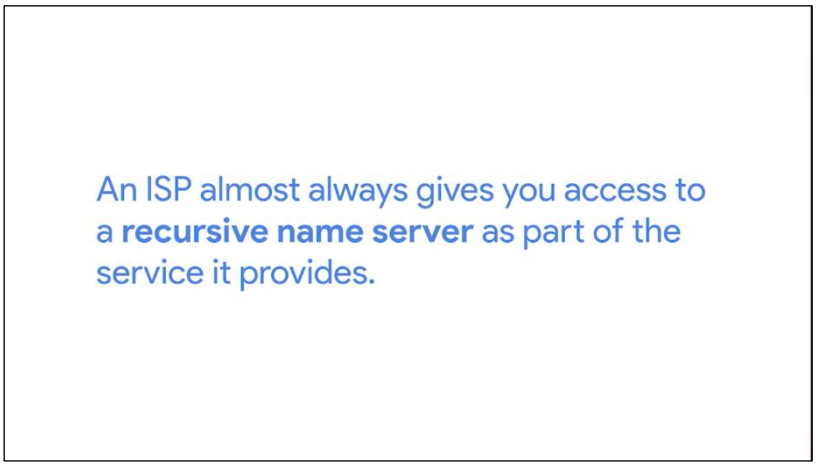
You can also enter set type= followed by a resource record type. By default, nslookup will return A records. But this lets you explicitly ask for AAAA or MX or even text records associated with the host.



If you really want to see exactly what's going on, you can enter set debug. This will allow the tool to display the full response packets, including any intermediary requests and all of their contents. Warning, this is a lot of data and can contain details like the TTL left, if it's a cached response, all the way to the serial number of the zone file the request was made against.

6.3.2 Public DNS servers

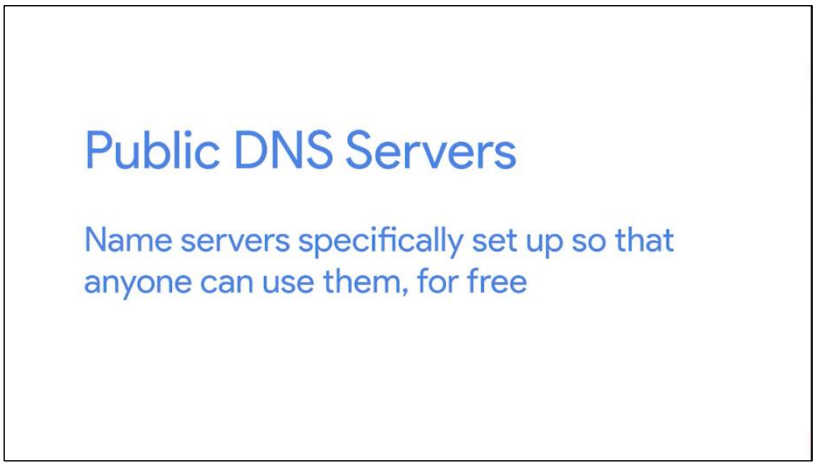
Having functional DNS is an important part of a functional network. An ISP almost always gives you access to a recursive name server as part of the service it provides.



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 dash 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. Don't worry, we'll cover that concept more in an upcoming lesson.

No matter what DNS service model you're using on your network, it's useful to have a way to test DNS functionality in case you suspect something isn't working right. It can also be super useful to have a backup DNS option in case you experience problems with your own.

You might even be in the early stages of building out a new network. And even if you plan to have your own name server eventually, it may not be ready for use. 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.



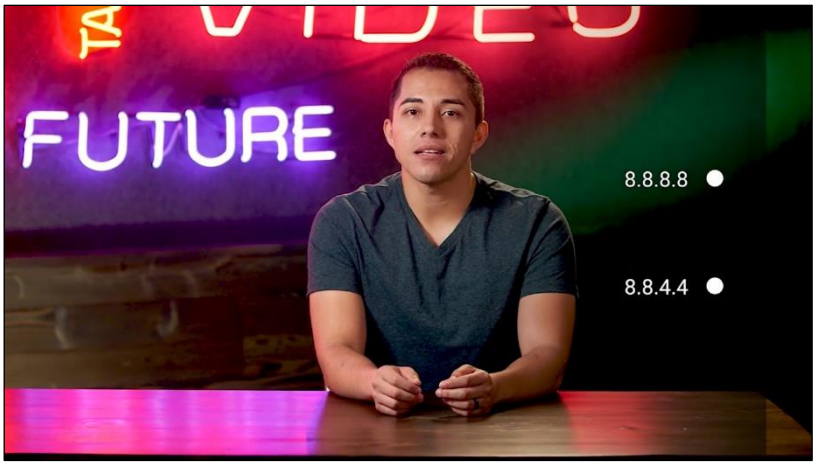
Using these public DNS servers is a handy technique for troubleshooting any kind of name resolution problems you might be experiencing. Some people just use these name servers for all the resolution needs. For a longtime, public DNS servers were a kind of tribal knowledge passed down from one sysadmin to another.

In ancient sysadmin law, it said that for many years the most commonly used public DNS servers were those run by Level 3 communications. One of the largest ISPs in the world. Level 3 is, in fact, so large. They mostly do business by selling connectivity to their network, to other ISPs that actually deal with consumers instead of dealing with end-users themselves.

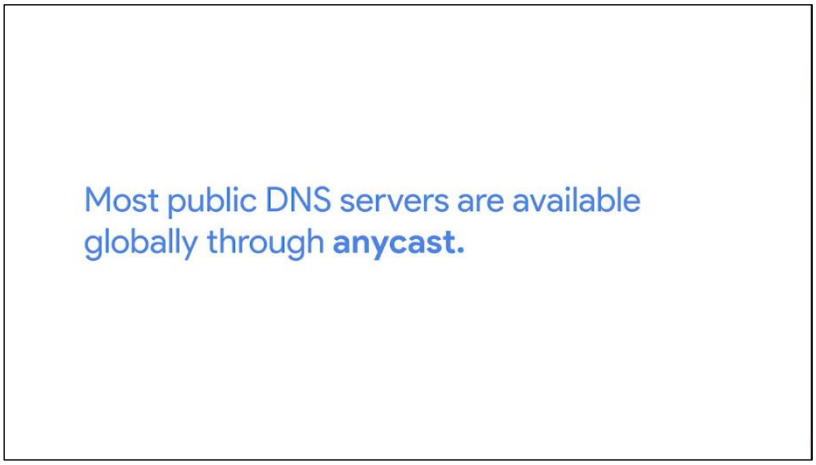
The IP addresses for Level 3s public DNS servers are 4.2.2.1 through 4.2.2.6. These IPs are easy to remember but they've always been shrouded in a bit of a mystery.

While they've been available for use by the public for almost 20 years now, it's not a service Level 3 officially has ever acknowledged or advertised. Why? We might never know. It's one of the great mysteries of our ancient sysadmin law.

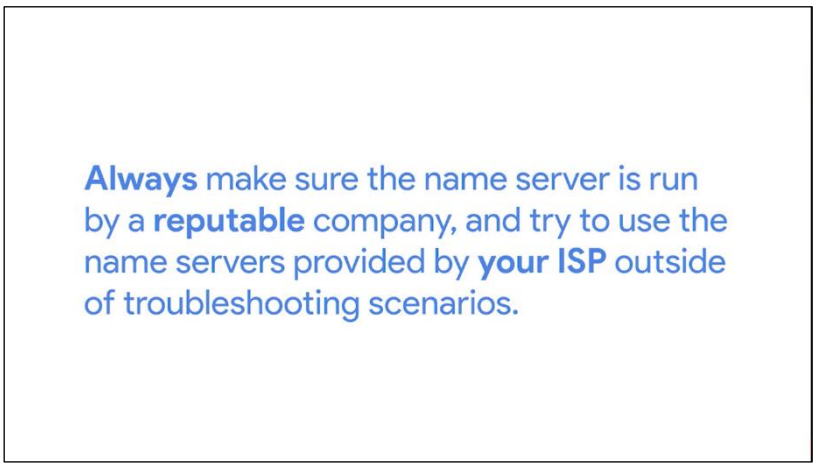
Anyway, other easy to remember options are the IPs for Google's public DNS. Google operates public name servers on the IPs 8.8.8.8 and 8.8.4.4.



Unlike the Level 3 IPs, these are officially acknowledged and documented by Google to be used for free by anyone. Most public DNS servers are available globally through anycast.



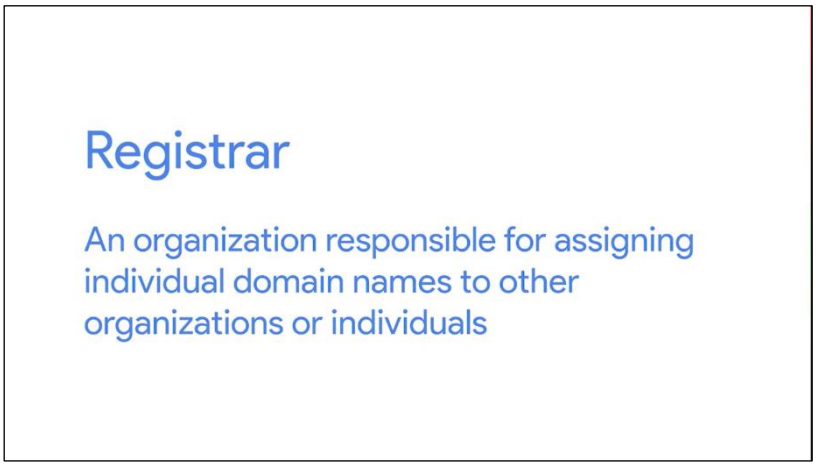
Lots of other organizations also provide public DNS servers, but few are as easy to remember as those two options. Always do your research before configuring any of your devices to use that type of name server. Hijacking outbound DNS requests with faulty responses is an easy way to redirect your users 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.

6.3.3 DNS registration and expiration

Refresher time. Remember that 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.

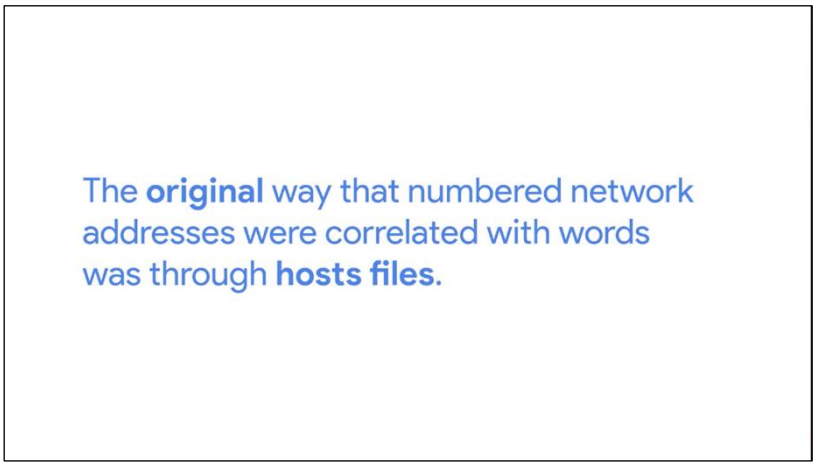
After that, ownership would move to the new owner or registrar. An important part of domain name registration is that these registrations only exist for a fixed amount of time. You typically pay to register domain names for a certain number of years. It's important to keep on top of when your domain names might expire because once they do, they're up for grabs and anyone else could register them.

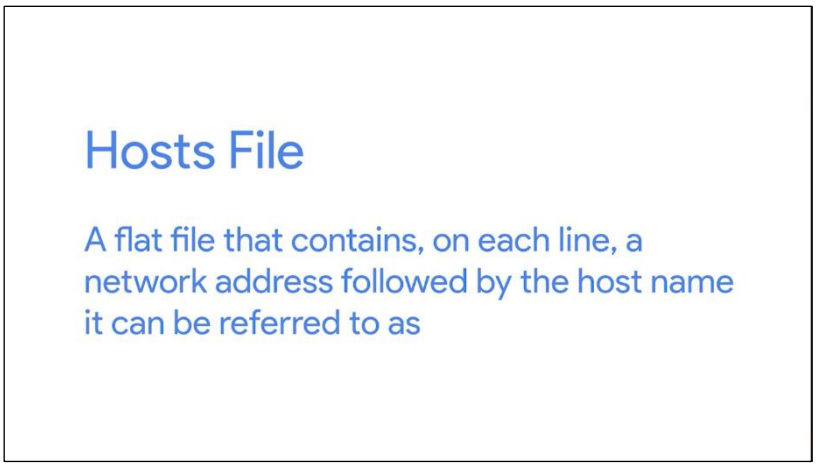
6.3.4 Host file

Long before DNS was an established and globally available technology, it was clear to computer operators that they needed a language-based system to refer to network devices. We've talked about

how humans are way better at remembering descriptive words than numbers. But numbers represent the natural way that computers think and communicate.

The original way that numbered network addresses were correlated with words was through hosts files. A host file is a flat file that contains on each line a network address followed by the host name it can be referred to as.



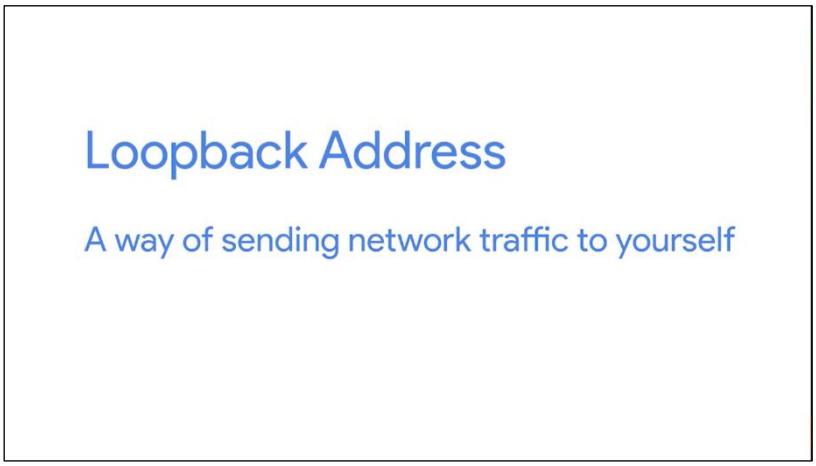


For example. A line in a host file might read, 1.2.3.4 webserver. This means that on the computer where this host file resides, a user could just refer to webserver, instead of the IP 1.2.3.4.

Hosts files are evaluated by the networking stack of the operating system itself. That means, the presence of an entry there would translate to anywhere you might refer to a networking address.

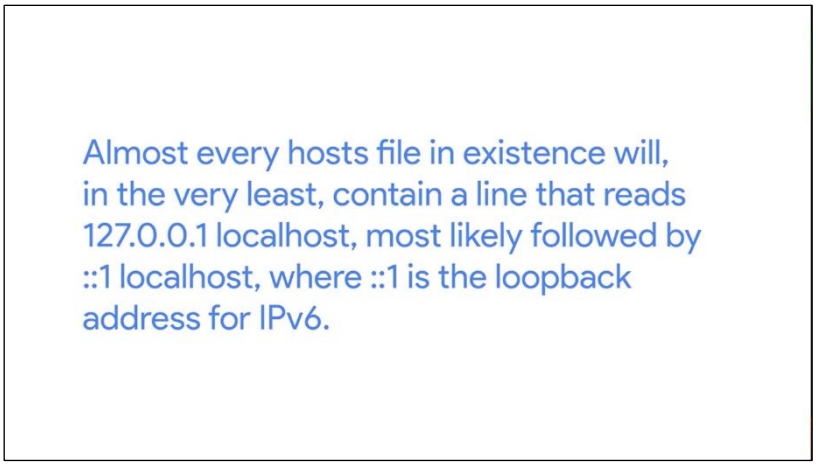
Sticking with our earlier example, a user could type webserver into a web browser URL bar, or could issue it pin web server command, and it would get translated to 1.2.3.4 in either case. Hosts files might be ancient technology but they've stuck around all this time. All modern operating systems including those that power our phone and tablets, still have hosts files.

One reason, is because of a special IP address we haven't covered yet. The loopback address. A loopback address always points to itself.

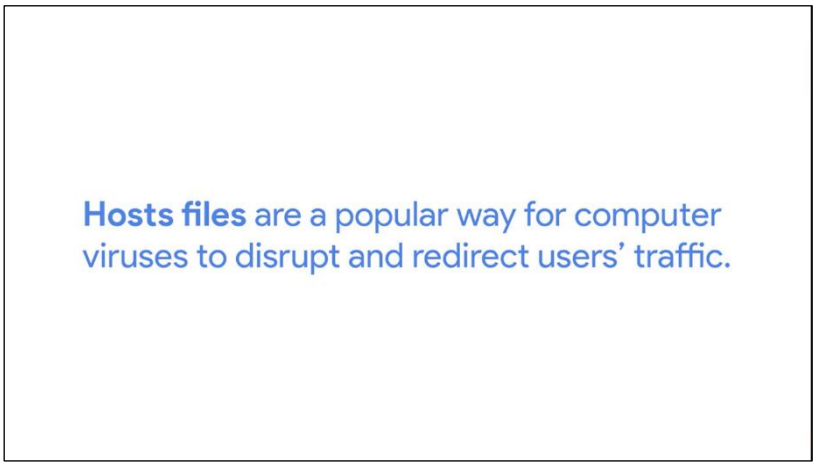


So, a loopback address is a way of sending network traffic to yourself. Sending traffic to a loopback address bypasses all network infrastructure itself, and traffic like that never leaves the node. The loopback IP for IPV4 is 127.0.0.1. And it's still, to this day, configured on every modern operating system through an entry in a hosts file.

Almost every hosts file in existence will in the very least contain a line that reads "127.0.0.1 localhost," most likely followed by "::1 localhost, " where "::1" is the loopback address for IPV6.



Since DNS is everywhere, host files aren't used much anymore. But they still exist and they're still important to know about. Some software even requires specific entries in the hosts file to operate properly as antiquated as this practice may seem. Finally, hosts files are a popular way for computer viruses to disrupt and redirect user's traffic.



It's not a great idea to use host files today. But they do have some useful troubleshooting purposes that can be helpful in IT support. Host files are examined before a DNS resolution attempt occurs on just about every major operating system. This lets you force an individual computer to think a certain domain name always points at a specific IP. Got it? We've covered a lot. So take time to go back if you need to, and make sure you understand the concepts we're discussing. Next up, a short quiz.

6.4 The cloud

6.4.1 What is the cloud

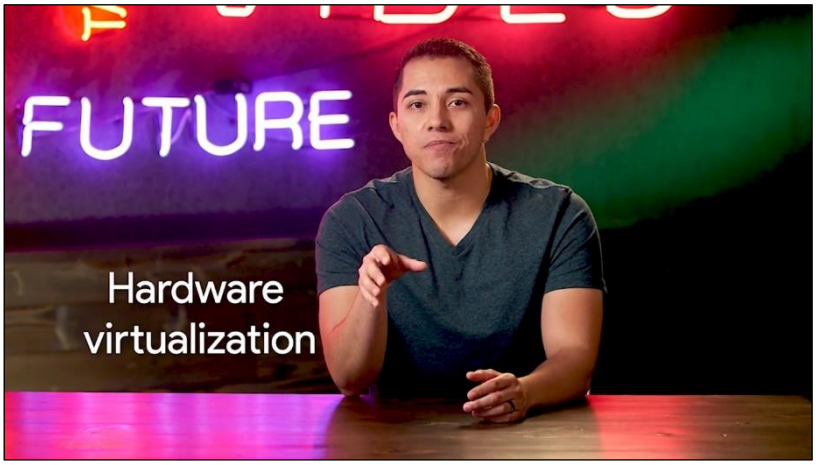
You've probably been hearing people talk about the cloud more and more. There are public clouds and private clouds and hybrid clouds and rain clouds but not really relevant here. There are cloud clients and cloud storage and cloud servers too. You might hear the cloud mentioned in newspaper headlines and TV advertisements.

The cloud is the future, so we're told, and IT support specialists really need to keep up on the latest innovations in tech in order to support them. But what exactly is the cloud? The truth is the cloud

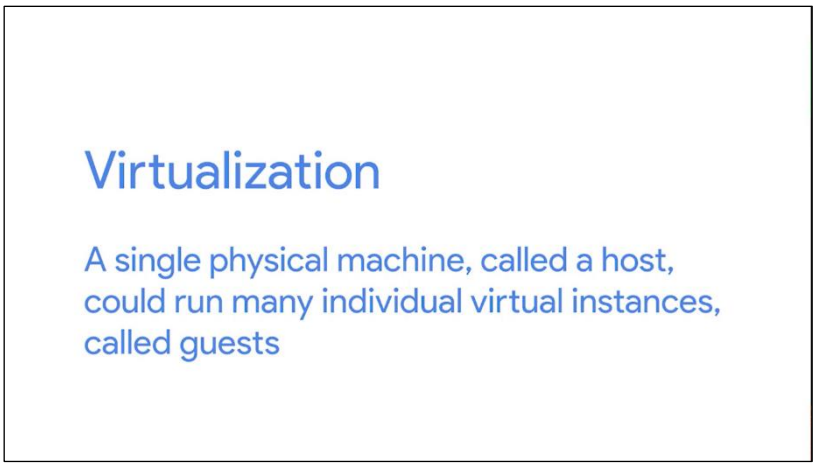
isn't a single technology or invention or anything tangible at all. It's just a concept, and to throw in another cloud joke, a pretty nebulous one at that.

The fact that the term 'the cloud' has been applied to something so difficult to define is pretty fitting. Basically, cloud computing is a technological approach where computing resources are provisioned in a shareable way so that lots of users get what they need when they need it.

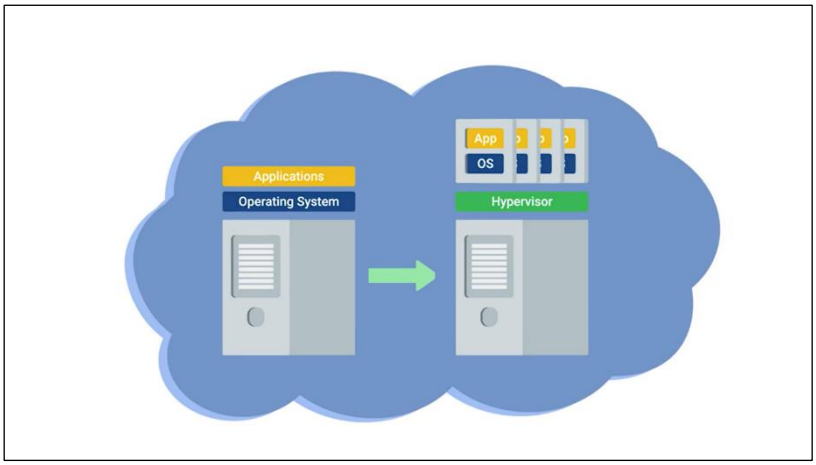
It's an approach that leans heavily on the idea that companies provide services for each other using these shared resources. At the heart of cloud computing is a technology known as hardware virtualization.



Hardware virtualization is a core concept of how cloud computing technologies work. It allows the concept of a physical machine and a logical machine to be abstracted away from each other. With virtualization, a single physical machine called a host could run many individual virtual instances called guests.

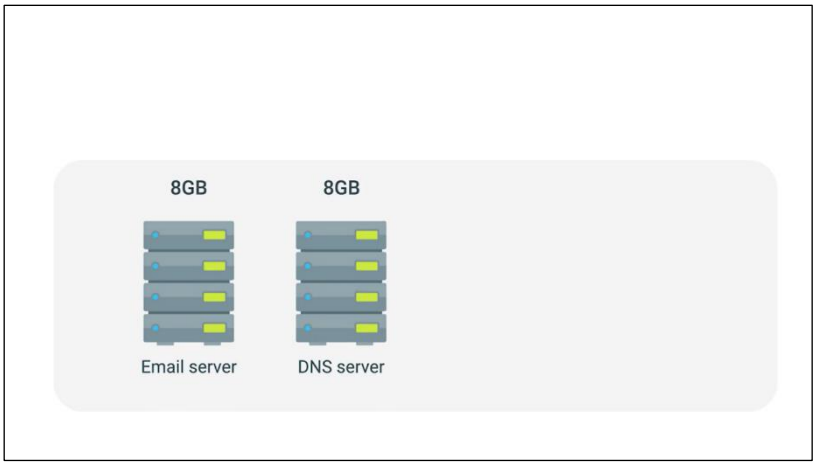


An operating system expects to be able to communicate with the underlying hardware in certain ways. Hardware virtualization platforms employ what's called a hypervisor. A hypervisor is a piece of software that runs and manages virtual machines while also offering these guests a virtual operating platform that's indistinguishable from actual hardware. With virtualization, a single physical computer can act as the host for many independent virtual instances. They each run their own independent operating system and, in many ways, are indistinguishable from the same operating systems running on physical hardware.

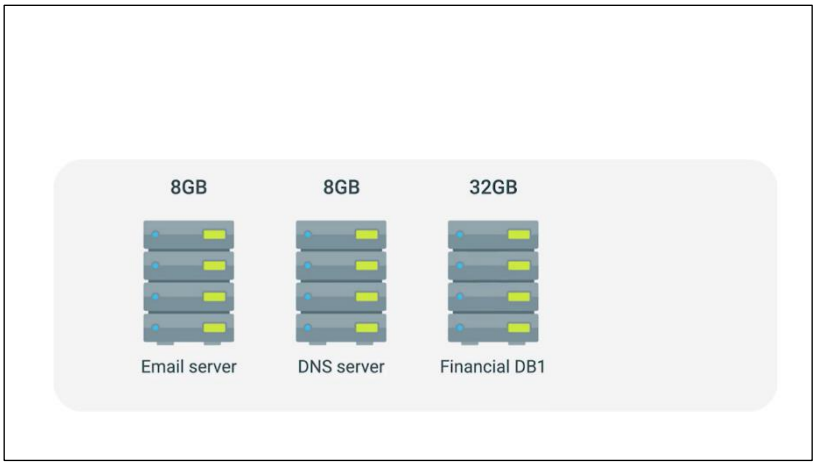


The cloud takes this concept one step further. If you build a huge cluster of interconnected machines that can all function as hosts for lots of virtual guests, you've got a system that lets you share resources among all of those instances. Let's try explaining this in a more practical way. Let's say you have the need for four servers. First, you need an email server. You've carefully analyzed things and expect this machine will need eight gigs of RAM to function properly.

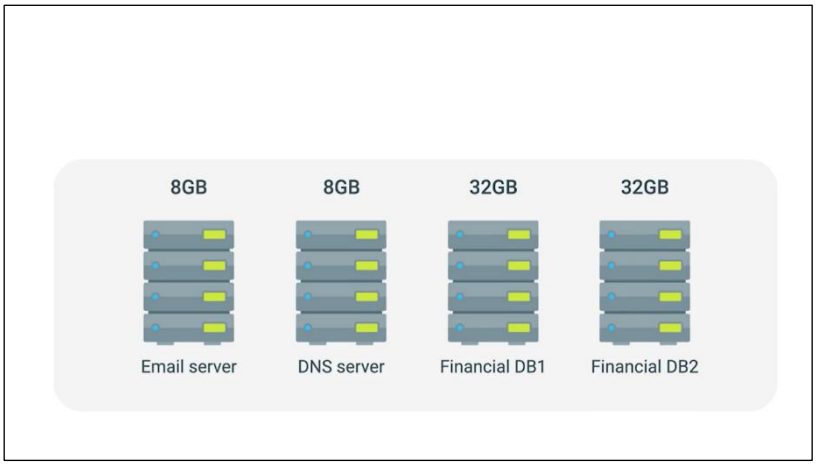
Next, you need a name server. The name server barely needs any resources since it doesn't have to perform anything really computational. But, you can't run it on the same physical machine as your email server since your email server needs to run on Windows, and your name server needs to run on Linux.



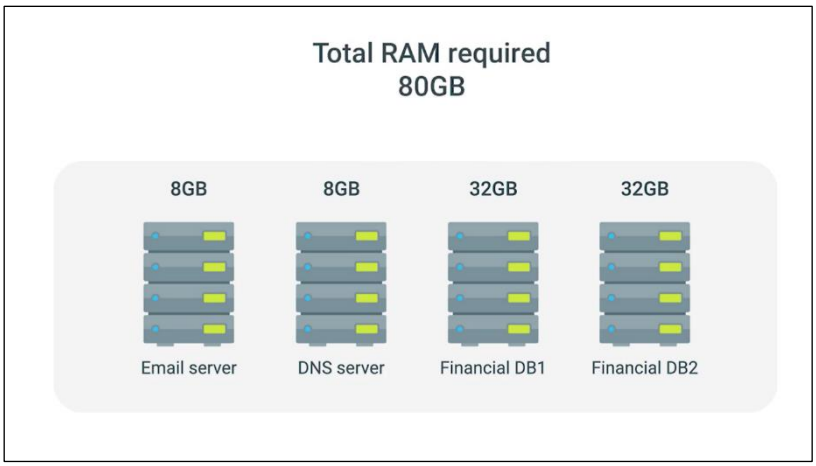
Now, the smallest server configuration your hardware vendor sells is a machine with eight gigabytes of RAM. So you have to buy another one with those specifications. Finally, you have a financial database. This database is normally pretty quiet and doesn't need too many resources during normal operations. But for your end of month billing processes to complete in a timely manner, you determine the machine would need 32 gigabytes of RAM.



It has to run on a special version of Linux designed just for the database so the name server can also run on this machine. So you order a server with that much RAM and then a second with the same specifications to act as a backup.

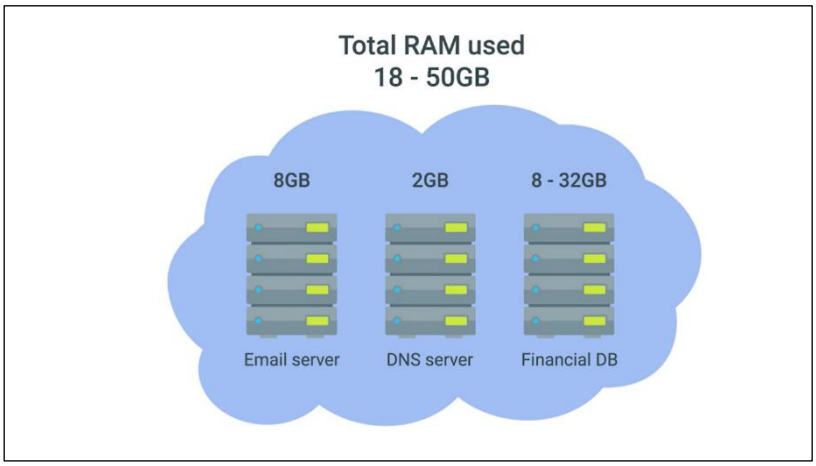


In order to run your business this way, you have to purchase four machines with a grand total of 80 gigabytes of RAM.



That seems pretty outrageous since it's likely that only 40 gigabytes of this total RAM will ever be used at one time. Most of the month you're using much less. That's a lot of money spent on resources you're either never going to use or rarely use. So let's forget about that model.

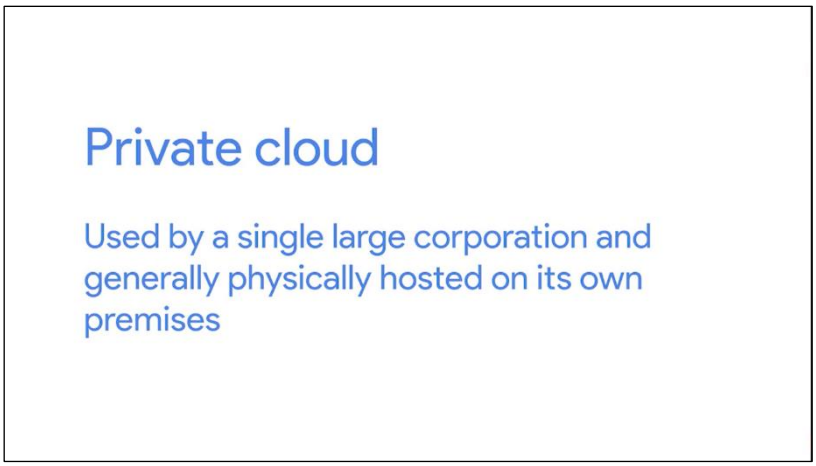
Instead, let's imagine a huge collection of interconnected servers that can host virtualized servers. These virtual instances running on this collection of servers can be given access to the underlying RAM as they need it.



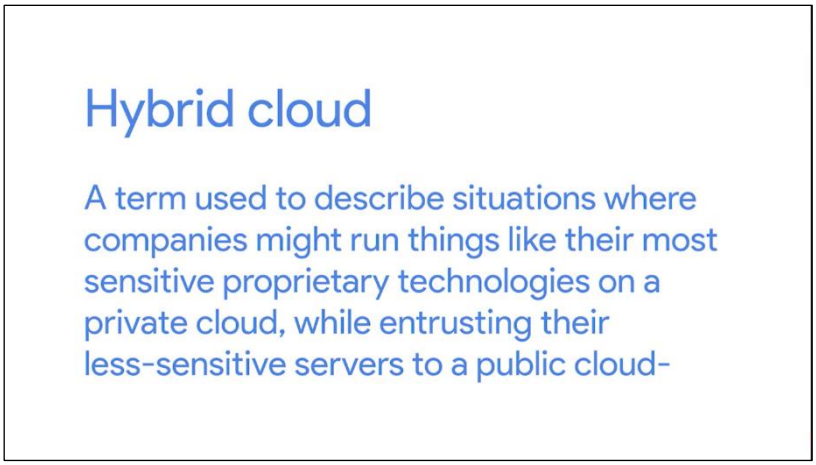
Under this model, the company that runs the collection of servers can charge you to host virtual instances of your servers instead of you buying the four physical machines. And it could cost much less than what you'd spend on the four physical servers. The benefits of the cloud are obvious.

But let's take it a step further. The cloud computing company that can host your virtualized instances also offer dozens of other services. So instead of worrying about setting up your own backup solution, you can just employ theirs. It's easy. And if you need a load balancer, you can just use their solution. Plus, if any underlying hardware breaks, they just move your virtual instance to another machine without you even noticing. To top it all off, since these are all virtual servers and services, you don't have to wait for the physical hardware you ordered to show up. You just need to click a few buttons in a web browser. That's a pretty good deal.

In our analogy, we used an example of what a public cloud is, a large cluster of machines run by another company. A private cloud takes the same concepts, but instead, it's entirely used by a single large corporation and generally physically hosted on its own premises.



Another term you might run into, a hybrid cloud, isn't really a separate concept. It's just a term used to describe situations where companies might run things like their most sensitive proprietary technologies on a private cloud while entrusting their less sensitive servers to a public cloud.

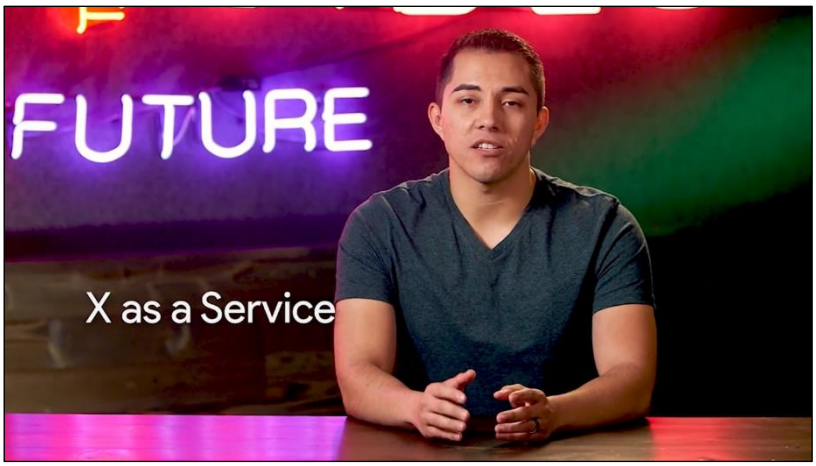


Those are the basics of what the cloud is. It's a new model in computing where large clusters of machines let us use the total resources available in a better way.

The cloud lets you provision a new server in a matter of moments and leverage lots of existing services instead of having to build your own. To sum up, its blue skies ahead for anyone using their cloud. Sorry, I couldn't resist.

6.4.2 Everything as a service

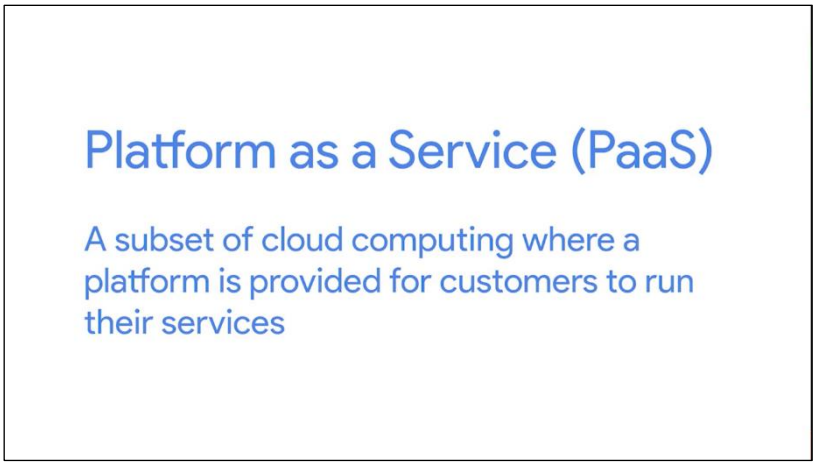
In our last video, we gave you a basic definition of what cloud computing is, but the term has really come to mean so much more than just hosting virtual machines. Another term that's been used more and more with the rise of cloud computing is X as a service. Here, the X can stand for lots of different things.



The way we've described the cloud so far would probably best be defined as infrastructure as a service or IaaS. The idea behind infrastructure as a service is that you shouldn't have to worry about building your own network or your own servers. You just pay someone else to provide you with that service.

Recently, we've seen the definition of the cloud expand well beyond infrastructure as a service. The most common of these are platform as a service, or PaaS, and software as a service, or SaaS.

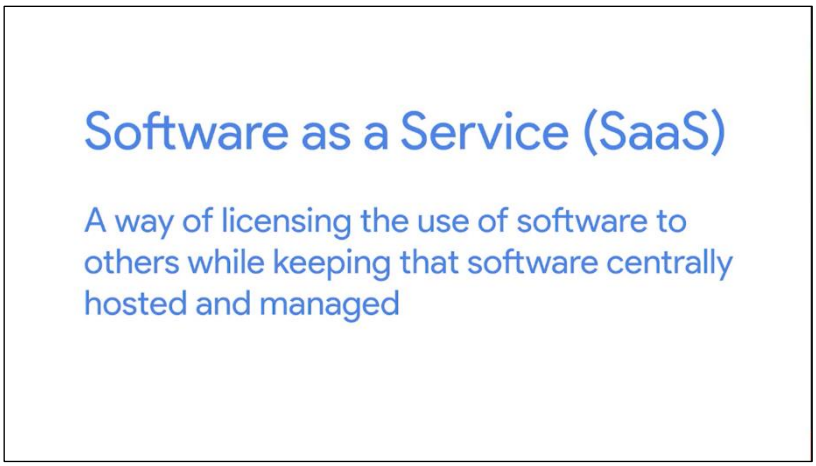
Platform as a service is a subset of cloud computing where a platform is provided for customers to run their services. This basically means that an execution engine is provided for whatever software someone wants to run.



A web developer writing a new application doesn't really need an entire server complete with a complex file system, dedicated resources, and all these other things. It doesn't matter if this server is virtual or not. They really just need an environment that their web app can run in. That is what platform as a service provides.

Software as a service takes this one step further. Infrastructure as a service abstracts away the physical infrastructure you need and platform as a service abstracts away the server instances you need.

Software as a service is essentially a way of licensing the use of software to others while keeping that software centrally hosted and managed.



Software as a service has become really popular for certain things. A great example is e-mail. Offerings like Gmail for Business from Google or Office 365 Outlook from Microsoft are really good examples of software as a service.

Using one of those services means you're trusting Google or Microsoft to handle just about everything about your email service. Software as a service is a model that's gaining a ton of traction. Web browsers have become so feature packed that lots of things that required standalone software in the past can now run well inside of a browser. And if you can run something in a browser, it's a prime candidate for SaaS.

Today, you can find everything from word processors to graphic design programs to human resource management solutions offered under a subscription based SaaS model. More and more, the point of a business' network is just to provide an internet connection to access different software or data in the cloud.

6.4.3 Cloud storage

Another popular way to use cloud technologies, is cloud storage. In a cloud storage system, a customer contracts a cloud storage provider to keep their data secure, accessible, and available.

This data could be anything from individual documents to large database backups. There are lots of benefits of cloud storage over a traditional storage mechanism. Without cloud storage, there's the general headache of managing a storage array. Hard drives are one of the most frequent components that may experience a malfunction in a computer system. That means that you'd have to carefully monitor the devices being used for storage and replace parts when needed. By using a cloud storage solution, it's up to the provider to keep the underlying physical hardware running.

Also, cloud storage providers usually operate in lots of different geographic regions. This let's you easily duplicate your data across multiple sites. Many of these providers are even global in scale which lets you make your data more readily available for users all over the world. This also provides protection against data loss, since if one region of storage experience has problems, you can probably still access your data in a different region. Cloud storage solutions also grow with you.

Typically, you'll pay for exactly how much storage you're using instead of having a fixed amount like you would with local storage. While this doesn't always mean that cloud storage is necessarily a cheaper option, it does mean that you can better manage what your expenses for storage actually are.

Not only is cloud storage useful for replacing large scale local storage arrays, it's also a good solution for backing up smaller bits of data. Your smartphone might automatically upload every picture you take to a cloud storage solution. If your phone dies, you lose it, or accidentally delete pictures, they're still there waiting for you in the cloud. That way, you'll never lose those precious photos of your pooch, Taco. Don't worry, Taco, all 2,000 of those photos of you are fully protected. While I look at some more pictures of Taco, it's time for a quick quiz.

6.5 IPV6

6.5.1 IPV6 addressing and subnetting

Time for some real talk. Here's the hard truth, 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. It's actually a fun bit of trivia. 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.

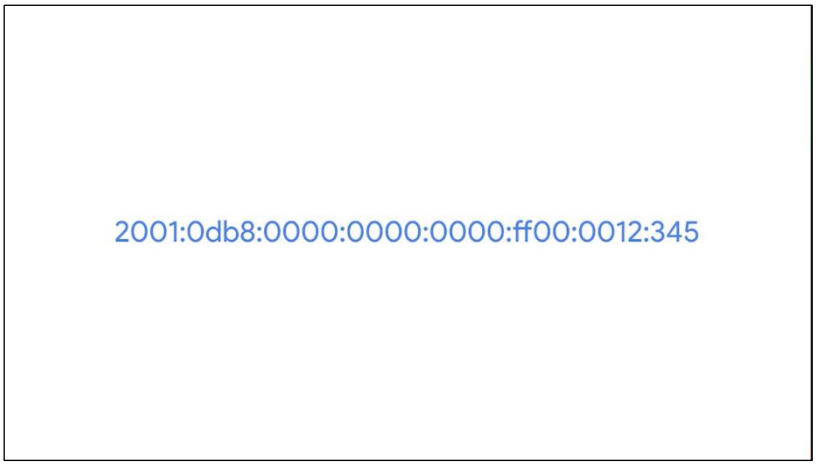
This size difference is staggering once you do the math. Don't worry, [LAUGH] we won't make you. 2 to the power of 128 would produce a 39-digit-long number. That number range has a name you've probably never even heard of, an undecillion. An undecillion isn't a number you hear a lot because it's ginormous.

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 for fun, let's look at what that number actually looks like. It looks like this.

340,282,366,920,938,463,463,374,607,431,768,211,456

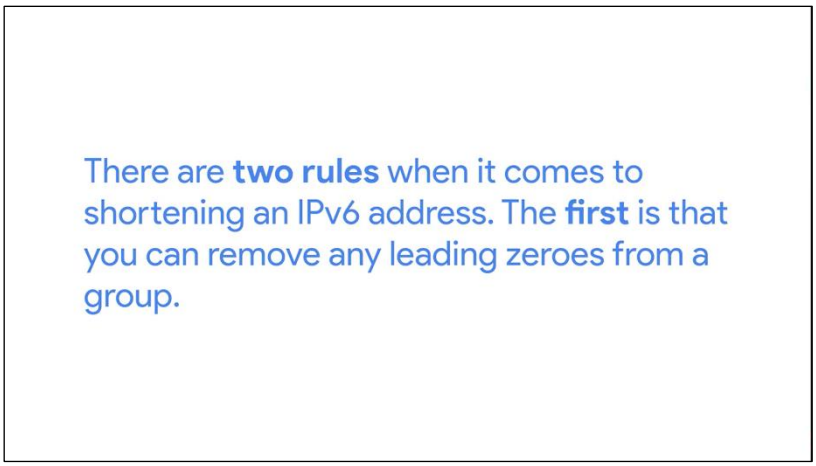
Whoa, mind blowing, right?

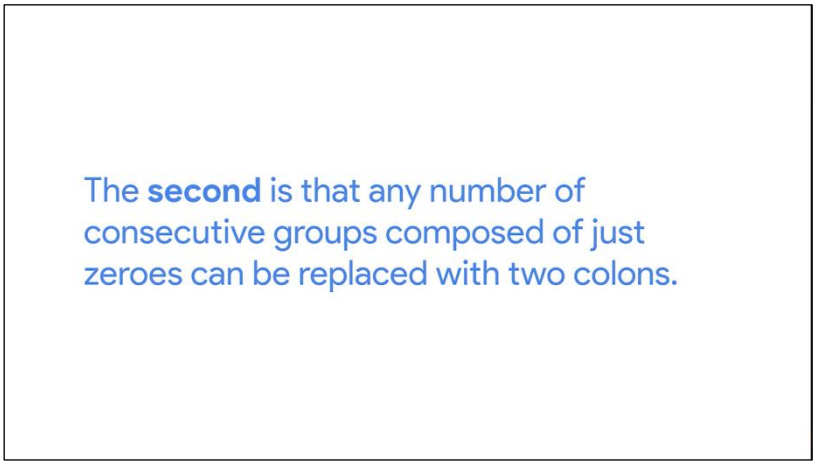
Just like how an IPv4 address is really just a 32-bit binary number, IPv6 addresses are really just 128-bit binary numbers. IPv4 addresses are written out in four octets of decimal numbers just to make them a little more readable for humans. But trying to do the same for an IPv6 address just wouldn't 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. A full IPv6 address might look something like this.



That's still way too long, 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 and courses just like this one. That's over 18 quintillion addresses, much larger than the entire IPv4 address space, reserved just for this purpose.

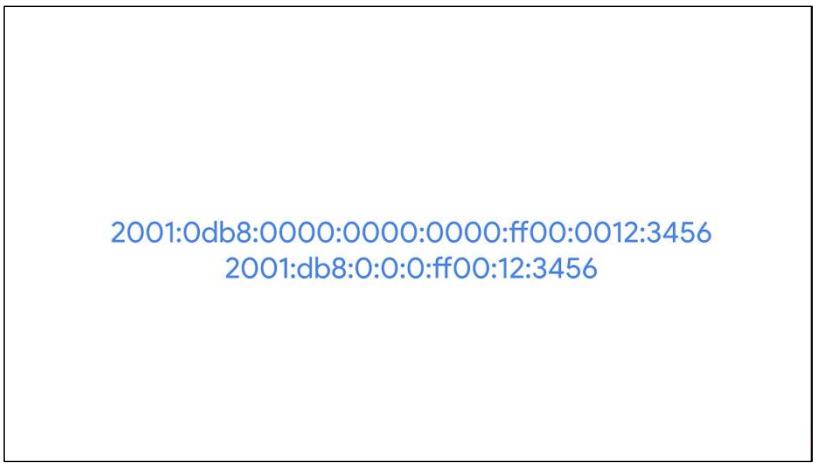
There are two rules when it comes to shortening an IPv6 address. The first is that you can remove any leading zeros from a group. The second is that any number of consecutive groups composed of just zeros can be replaced with two colons.



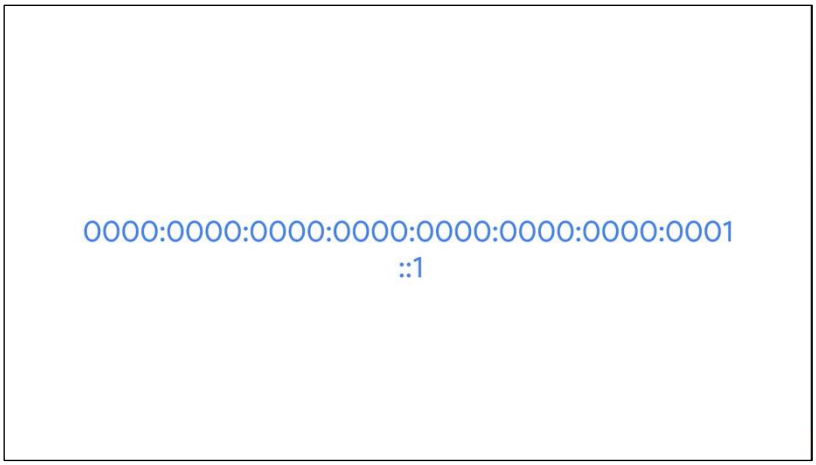


I should call out that this can only happen once for any specific address. Otherwise, you couldn't know exactly how many zeros were replaced by the double colons. For this IP, we could apply the first rule and remove all leading zeros from each group. This would leave us with this.

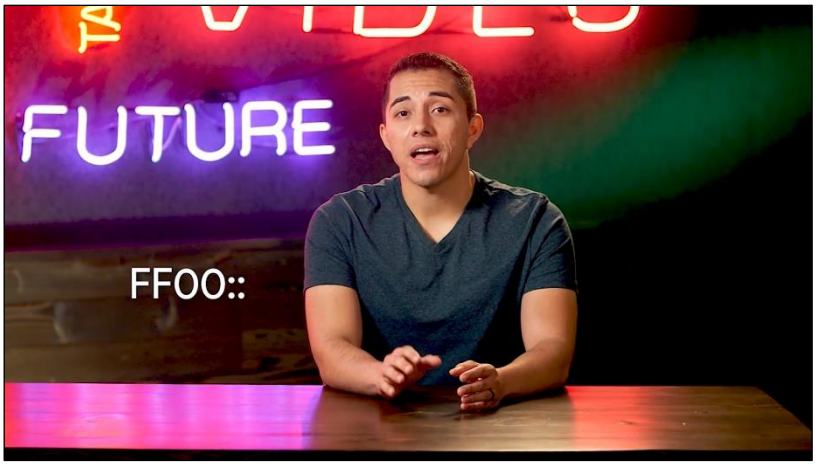
Once we apply the second rule, which is to replace consecutive sections containing just zeros with two colons, we'll end up with this.



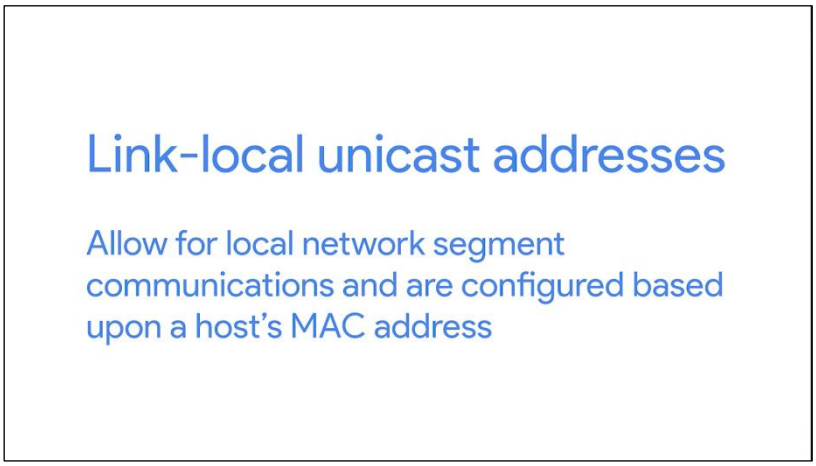
This still isn't as readable as an IPv4 address, but it's a good system that helps reduce the length a little bit. We can see this approach taken to the extreme with IPv6 loopback address. You might remember that with IPv4, this address is 127.0.0.1. With IPv6, the loopback address is 31 0s with a 1 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.

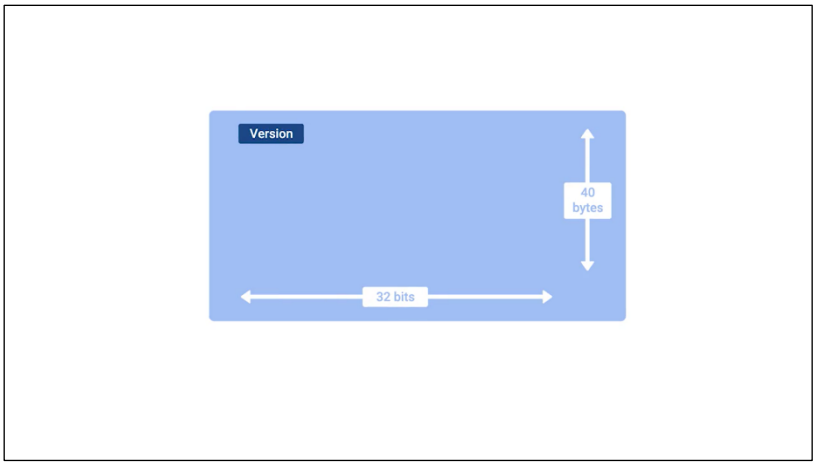
The IPv6 address space is so huge, there was never any need to think about splitting it up into address classes like we use to do with IPv4. From the very beginning, an IPv6 address had a very simple line between network ID and host ID. The first 64 bits of any IPv6 address is the network ID, and the 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.

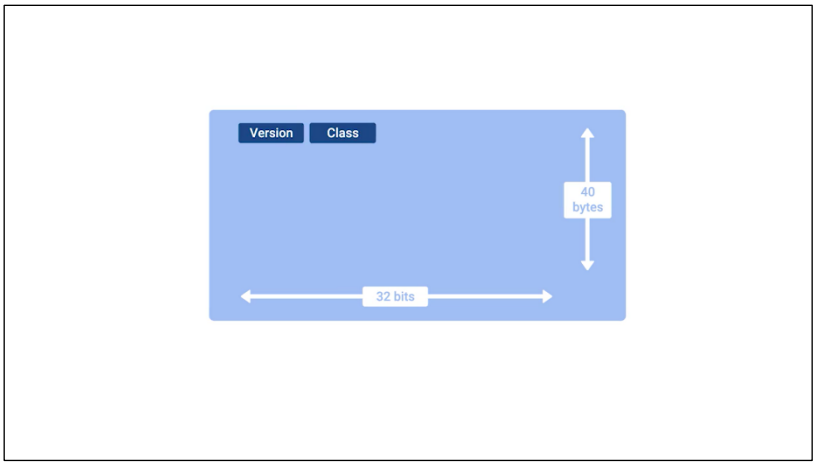
6.5.2 IPV6 headers

When IPv6 was being developed, they took the time to introduce a few improvements instead of just figuring out a way to increase the address size. This should come as a relief to you, and IT support specialists love networks that perform well. One of the most elegant improvements was made to the IPv6 header, which is much simpler than the IPv4 one.

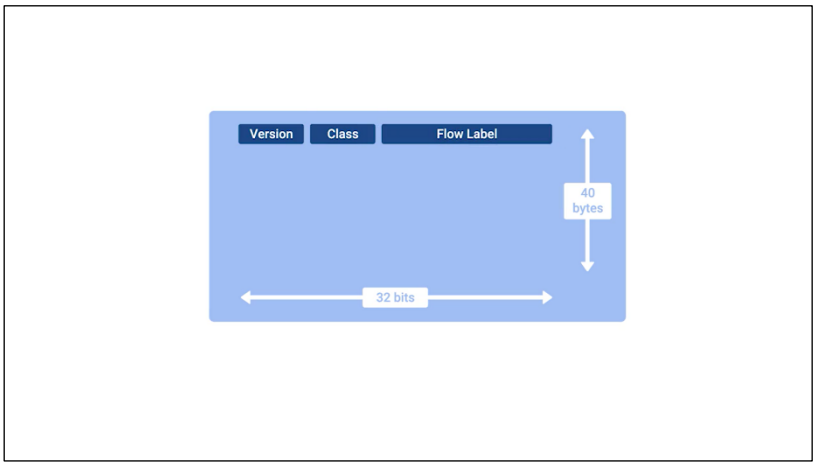
The first field in an IPv6 header is the version field. This is a 4-bit field that defines what version of IP is in use. You might remember that an IPv4 header begins with this exact same field.



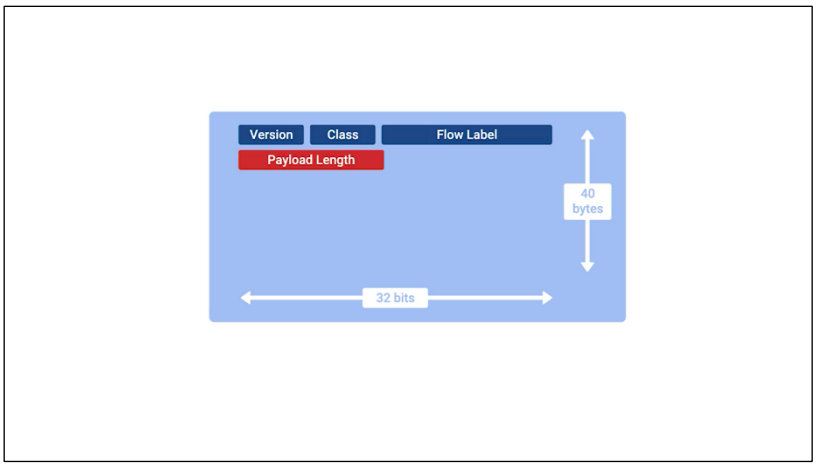
The next field is called the traffic class field. This is an 8-bit field that defines the type of traffic contained within the IP datagram and allows for different classes of traffic to receive different priorities.



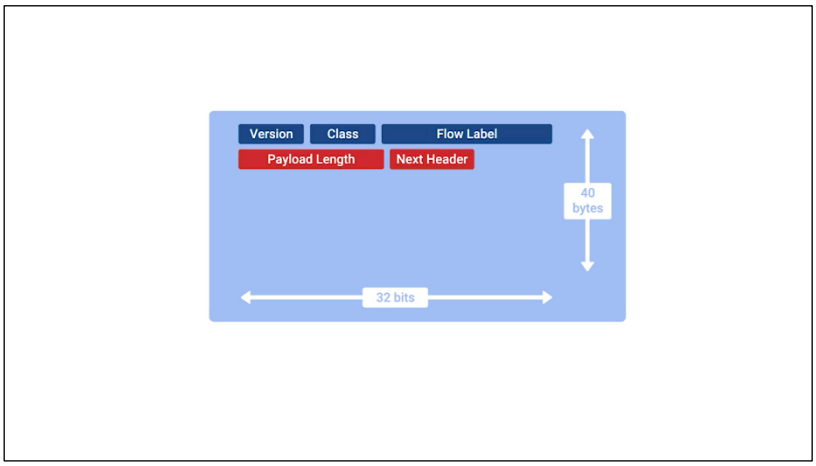
The next field is the flow label field. This is a 20-bit field that's used in conjunction with the traffic class field for routers to make decisions about the quality of service level for a specific datagram.



Next you have the payload length field. This is a 16-bit field that defines how long the data payload section of the datagram is.

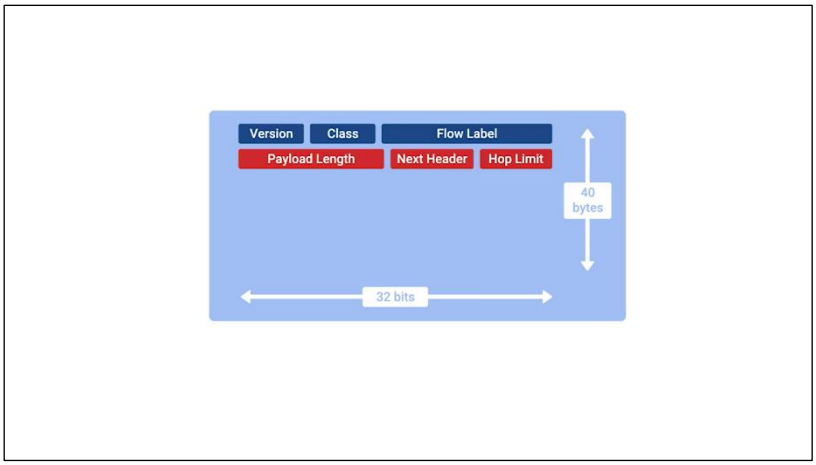


Then you have the next header field. This is a unique concept to IPv6, and needs a little extra explanation. IPv6 addresses are four times as long as IPv4 addresses. That means they have more ones and zeros, which means that they take longer to transmit across a link. To help reduce the problems with additional data that IPv6 addresses impose on the network, the IPv6 header was built to be as short as possible. One way to do that is to take all of the optional fields and abstract them away from the IPv6 header itself.

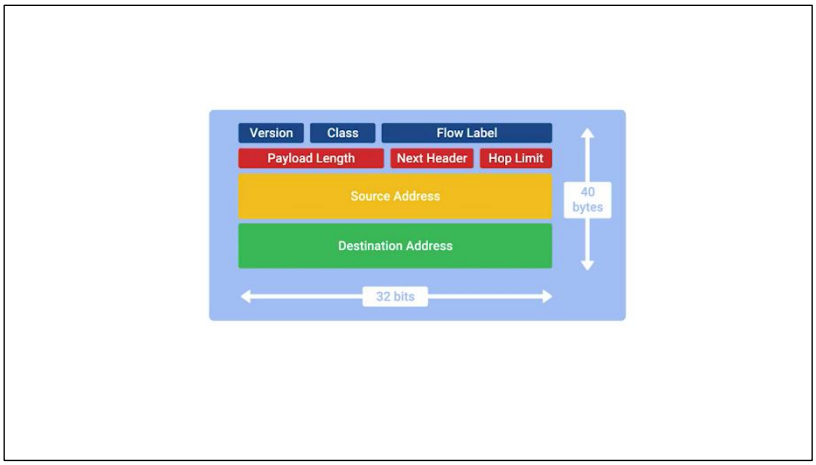


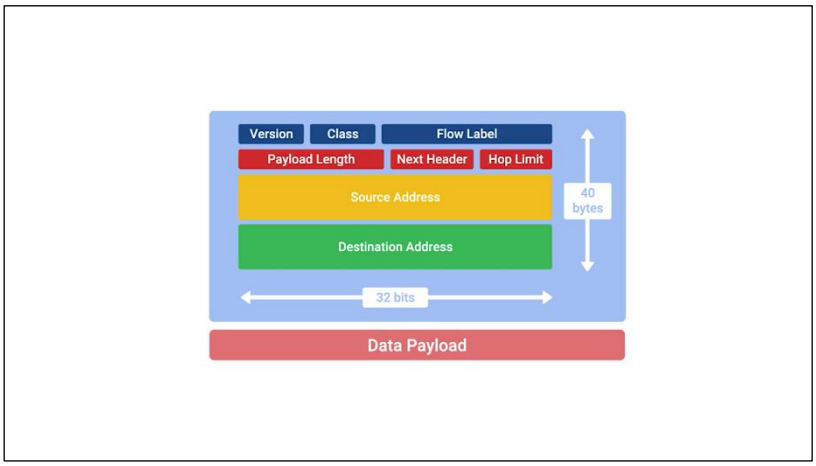
The next header field defines what kind of header is immediately after this current one. These additional headers are optional, so they're not required for a complete IPv6 datagram. Each of these additional optional headers contain a next header field and allow for a chain of headers to be formed if there's a lot of optional configuration.

Next we have what's called the hop limit field. This is an 8-bit field that's identical in purpose to the TTL field in an IPv4 header.



Finally, we have the source and destination address fields, which are each a 128 bits. If the next header field specified another header, it would follow at this time. If not, a data payload the same length as specified in the payload length field would follow.



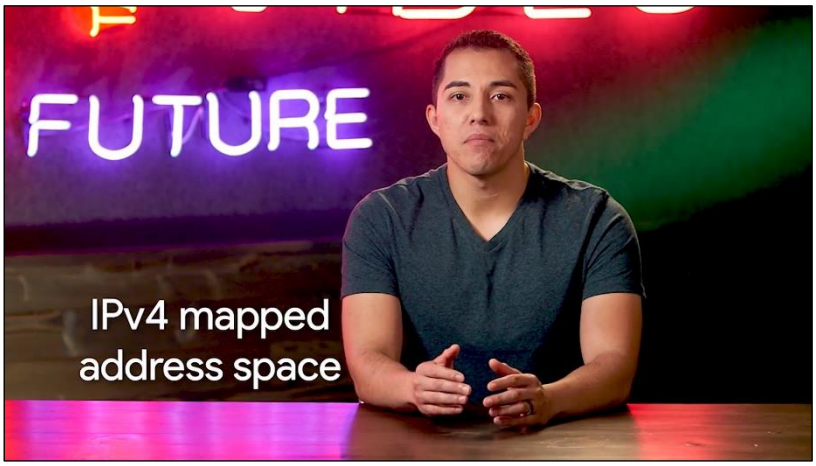


6.5.3 IPV6 and IPV4 harmony

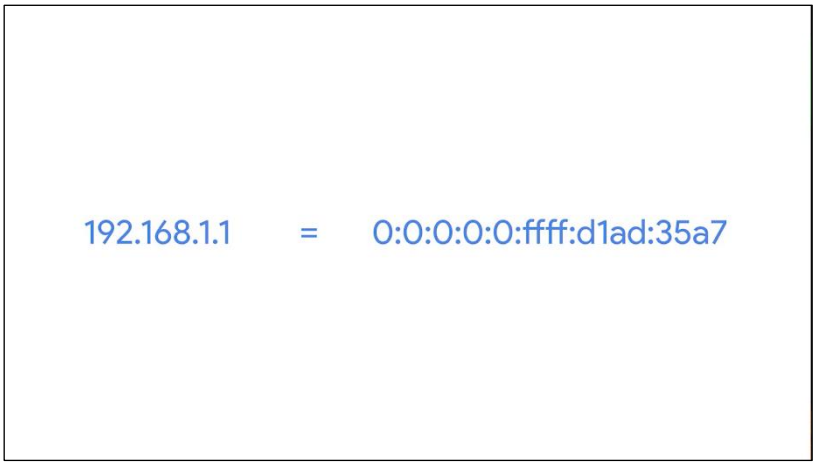
It's just not possible for the entire Internet and all connected networks to switch to IPv6 all at once. There would be way too much coordination at play. Too many old devices that might not even know how to speak IPv6 at all, still requiring connections. So the only way IPv6 will ever take hold is to

develop a way for IPv6 and IPv4 traffic to coexist at the same time. This would let individual organizations make the transition when they can.

One example of how this can work is with what's known as IPv4 mapped address space.

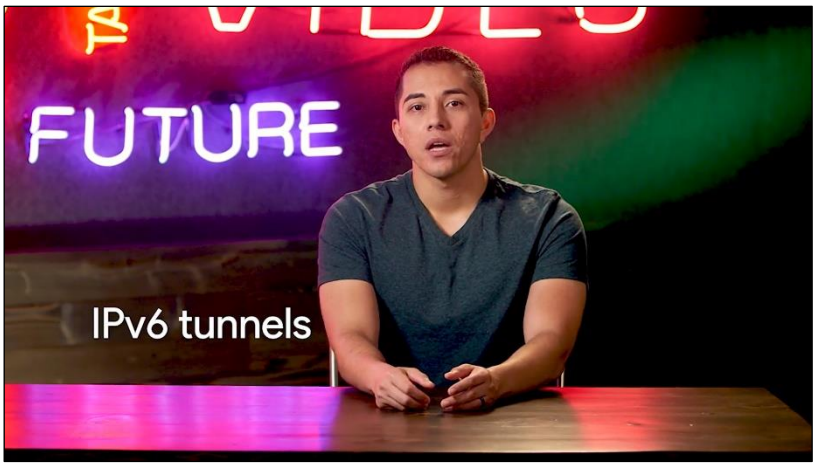


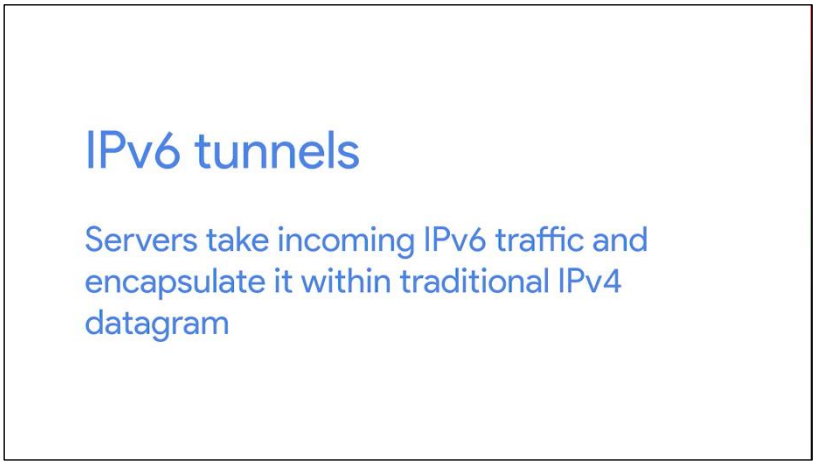
The IPv6 specifications have set aside a number of addresses that can be directly correlated to an IPv4 address. Any IPv6 address that begins with 80 zeros, and is then followed by 16 ones is understood to be part of the IPv4 mapped address space. The remaining 32 bits of the IPv6 address is just the same 32 bits of the IPv4 address it's meant to represent.



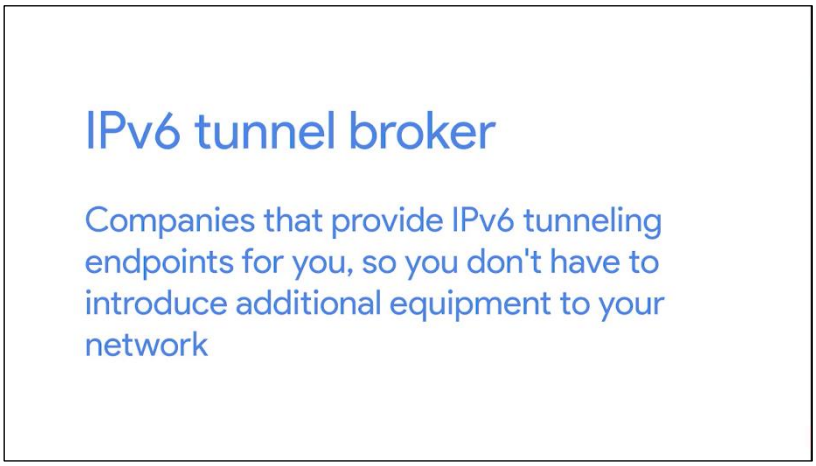
This gives us a way for IPv4 traffic to travel over an IPv6 network. But probably more important is for IPv6 traffic to have a way to travel over IPv4 networks. It's easier for an individual organization to make the move to IPv6 than it is for the networks at the core of the Internet to. So while IPv6 adoption becomes more widespread, it'll need a way to travel over the old IPv4 remnants of the Internet backbone.

The primary way this is achieve today is through IPv6 tunnels. IPv6 tunnels are conceptually pretty simple. They consist of IPv6 tunnels servers on either end of a connection. These IPv6 tunnel servers take incoming IPv6 traffic and encapsulate it within traditional IPv4 datagrams.





This is then delivered across the IPv4 Internet space where it's received by another IPv6 tunnel server. That server performs the de-encapsulation and passes the IPv6 traffic further along in the network. Along with IPv6 tunnel technologies, the concept of an IPv6 tunnel broker has also emerged. These are companies that provide IPv6 tunneling endpoints for you, so you don't have to introduce additional equipment to your network.



There are a lot of competing protocols to be used for these kinds of IPv6 tunnels. Since this is still a new and evolving space, it's not clear who the winner will be. I've left you with some links to read about the main competitors right after this video. It doesn't really matter which tunneling technology ends up becoming the most common solution. It'll probably fade away in time itself. The future of networking is the adoption of IPv6 as the main protocol at the network layer, and one day we won't need any tunnels at all. The future is limitless, and tunnelless, or something like that.

You've done an amazing job at getting through all this information. So take some time to pat yourself on the back. You've got one final quiz and a final project to get through. And then you can check this course off your to do list. Are you starting to see the light at the end of the tunnel?

7 Course wrap-up

7.1 Course wrap-up

You did it. You should be really proud of yourself because getting through all of this material is a huge accomplishment. The material we've covered has been pretty technical and super complicated and getting through it all is a real feat. Take a moment to think about just how much you've learned. You now know a lot about how computers communicate with each other, which is an essential part of how people communicate with each other.

Computer networks are used by billions of people every day and they form the backbone of the global economy. You've learned about how signals are carried across cables and how many different protocols are used in conjunction to make sure this data is delivered properly. You've learned about all sorts of network services like DNS that help humans use computers. This is all very important to learn. You'll be able to apply all of this knowledge into your I.T. support career. You can also just use it to help your own home network run better. Either way, congrats. You've given yourself a leg up.

Next time you visit a social media site or stream a video or even just chat with your friends and family online, take a moment to think about how amazing it is that so many different network devices and layers and protocols are involved with every little bit of data sent across the Internet. And you should also take a moment to marvel at the fact that you now understand how all of that works. Congratulations.

In the next course, Operating Systems and You, becoming a power user. My friend and colleague Cindy Quach, will be your guide as you navigate the Windows and Linux OSes. Get ready to have some fun and get your hands dirty as Cindy teaches you how to become a command line wizard

7.2 Alex career path

[MUSIC] So I think a big misconception a lot of people have about IT or tech work in general is that it's complicated or that it's difficult or that only some small subset of people are capable of handling these tasks. And none of that is true. These are very specific skills that you have to know, and that's absolutely true. And maybe it's true that most people don't have these skills, but that doesn't mean that people can't learn. IT is incredibly stable. You have the opportunity to make a very good living. You have a ton of directions that your career can go. So many people I know started off in desktop support and are now network engineers or site reliability engineers or software engineers. And it's an industry that's not going away. It's only growing. There's always going to be opportunities. I studied philosophy and history and actually also minored in creative writing. And while some of those skills have absolutely transferred and allowed me to excel at my job, I didn't learn anything about the technical aspects from my time in college at all. I've never taken an academic computer science course, I've never taken a course in networking. I've never taken a course in anything relating to computers in any way whatsoever. Education is important but there are other ways that you can learn the skills that you need to have a successful career in IT in fact.

7.3 Congratulations

You're probably exhausted right now, but congratulations. And I hope that all this was super exciting and that you're really excited about where you can take this now. And just congratulations. >> Congratulations. >> Congrats, that's great work.