

4

Layer 3 Network Layer and the Internet Protocol (IP)

In this section you're going to learn about the Network Layer of the OSI model, and the Internet Protocol (IP). The Network Layer of the OSI model is the layer that's responsible for end-to-end delivery of network data, as opposed to the point-to-point delivery that happens in the Data Link Layer. Almost all networks, including the Internet use a protocol called the Internet Protocol or IP at this layer. The Internet Protocol defines several things, including IP addresses and the rules for packing data from the upper layers into packets. There's quite a bit to IP, so we'll be breaking this section up into the following:

1. History of IP
2. IP Address Basics
3. IP Packets
4. Network Mask (Netmask) Basics
5. Subnetting and Netmask Basics - Delivery of IP Packets
6. Classless Inter-Domain Routing (CIDR)
7. Default Router (Gateway)
8. DNS Basics
9. IP Configuration Basics
10. IP Troubleshooting
11. IP Packet Details
12. IP Address Assignment
13. IP Address Blocks/Classes
14. Non-Routable IP Addresses
15. Subnetting – Advanced
16. DHCP
17. IPV6

Objectives

At the end of this section you will be able to:

1. Define what IP addresses are, how they are used, and differentiate them from MAC addresses.
2. Given two IP addresses and a netmask, determine whether the IP addresses are on the same or different network segments.
3. Describe the role of the default router, and why it's IP address must be part of the TCP/IP configuration.
4. Given a netmask, determine the number of networks and number of hosts.
5. Identify the sections of an IP packet.
6. List the IP address classes.
7. Describe how non-routable IP addresses function.
8. Describe how non-routable IP addresses freed up most of the IPv4 address space.
9. Describe the purpose of DNS in the network stack.
10. Configure TCP/IP settings on a Windows based computer.
11. Use Wireshark to analyze IP information.
12. Troubleshoot problems with TCP/IP configuration.

There are several resources that you should use to learn this material.

1. These Lecture Notes
2. Videos linked in Canvas.
3. Labsim Section 4

If you feel like any of the concepts presented in this section are not clear or need additional resources, you can always do your own research on the Internet or contact the instructor for help.

Introduction

In this section you're going to learn about the Network Layer and the Internet Protocol (IP) which is the main protocol used at this layer. You'll start with a quick history lesson, and then learn about the Network Layer and IP. Learning about how everything works in the Network Layer can be a little tricky because there are lots of interrelated parts, and understanding any single part requires learning about the other parts first. Rather than trying to learn all the components at the same time we're going to take cover everything three times. The first time through you'll get an overview of the components and how they work together. Next, we'll swing back around cover each separately, providing you with the basics of each component. And finally, we'll cover each component one more time, but this time covering any important details.

History of IP

Let's start with a quick look backwards at the history of the Internet Protocol (IP). You don't really need to know the names and dates, but learning about the history of the IP is important because it will help you understand how it became the main protocol used on the Internet. You'll also learn about the IETF which is the organization in charge of IP. You'll learn about how the IETF came to be in charge of maintaining the standards for all protocols used on the Internet, including IP.

In the early days of computer networking, there were several competing protocols for transmitting data over networks. However, the development of the IP proved to be a significant milestone in the history of networking, as it provided a standard way to move data from end-to-end in a network. Unlike protocols that relied on MAC addresses to move data from point-to-point on single network segments, IP allowed data to be transmitted across multiple networks, enabling the creation of the global Internet we know today.

The Internet Protocol was developed in the early 1970s by a team of researchers led by Vinton Cerf and Bob Kahn^{1,2,3}. At the time, there was a need for a new networking protocol that could connect different computer networks together, as the existing protocols were not suitable for this purpose. Cerf and



Bob Kahn and Vinton Cerf, developers of the Internet Protocol. From <https://northernvirginiamag.com/culture/culture-features/2021/07/23/meet-mclean-residents-bob-kahn-and-vint-cerf-they-invented-the-internet/>

Kahn's work on IP was part of a larger project known as the ARPANET, which was a research project funded by the US Department of Defense. The ARPANET was the precursor to the modern Internet, and the development of IP was a critical step in the creation of the Internet as we know it today.

The IP protocol defines two main items, the format for network packets and a system of addresses called IP Addresses. You'll learn the details about both these items later in this section, but for now here's a quick overview. The IP network packet contains two parts: the header and the data. The header contains information about the source and destination of the packet, as well as other information. The data contains the actual information being transmitted, such as an email message or a web page. The other key component defined by the IP protocol is the system of IP addresses. An IP address is a unique

¹ Cerf, V. G., & Kahn, R. E. (1974). A protocol for packet network intercommunication. IEEE Transactions on Communications, 22(5), 637-648

² Cerf, V. G., & Kahn, R. E. (1978). The current state of the art of packet switching: An overview. IEEE Communications Magazine, 16(2), 129-138

³ Internet Society. (n.d.). Brief history of the Internet. Retrieved from <https://www.internetsociety.org/internet/history-internet/brief-history-internet/>

identifier assigned to every device connected to the Internet and they are used to move network data from end-to-end, as opposed to MAC addresses which are used to move data from point-to-point. IP addresses are a 32-bit number that is usually represented in a dotted decimal notation, such as 192.168.0.1.

IP was designed to be a simple, lightweight protocol that could route packets of data across different networks. However, it had some limitations, such as a lack of reliability and no built-in error checking. To address these issues, Cerf and Kahn developed the Transmission Control Protocol (TCP), which was added to IP to create the TCP/IP protocol suite. TCP/IP was developed in the 1970s and adopted as the protocol standard for ARPANET (the predecessor to the Internet) in 1983. The development of TCP/IP was a significant breakthrough in the history of computer networking, as it allowed different computer networks to communicate with each other using a common set of protocols. Today, TCP/IP is the foundation of the Internet, and it is used by virtually every device that is connected to the Internet. Sadly, unlike other tech moguls like Gates, Zuckerberg, and Musk, most people won't recognize the names Cerf and Kahn even though their work made today's Internet possible. They are highly recognized and respected by the tech industry and were awarded the National Medal of Technology and Innovation in 1997.

IETF – Who's (not) the Boss

In this section you'll learn about the IETF, which is the group in charge of Internet standards and about the RFCs or Request For Comment documents which are used to publish the technical details behind the standards. This information is being presented because in any career field it's important to know who's in charge of any laws or standards governing the work.

While Cerf and Kahn were the original developers of the Internet Protocol (IP), they eventually turned the responsibility for managing the protocol to others in the ARPA project. There were two groups that had control over different aspects of IP, the Internet Configuration Control Board (ICCB)⁴ and the Defense Data Network (DDN) Network Information Center (NIC). The ICCB had been responsible for managing the early development of the Internet, while the DDN NIC had been responsible for managing

⁴ https://itlaw.fandom.com/wiki/Internet_Configuration_Control_Board

ARPANET, the defense network that the Internet had grown out of. The Internet Engineering Task Force (IETF) was created in 1986 as a result of the merging of two groups:

The IETF was tasked with developing and promoting the technical standards and protocols for the Internet. The IETF was initially composed of a small group of researchers and engineers who were working on the development of the Internet, but it has since grown into a large, international organization with thousands of members from around the world. Today the IETF is a standards organization that is open to anyone who wants to participate. The IETF develops technical standards for the Internet through a process of consensus-building and publishes the resulting standards in a series of documents known as Request for Comments (RFCs).

An important point, a critical point to understand, is that the IETF is not "in charge" of Internet technical standards in the traditional sense. Rather, it is a community-driven organization that operates through a process of open, collaborative discussion and consensus building. The IETF doesn't have any legal authority to enforce its standards or protocols, but its work is widely respected and has been adopted by industry and governments around the world. The only way for the Internet to work is for everyone that uses it to cooperate and follow the IETF's suggested standards. This means that anyone that manages a network segment connected to the Internet, and all the organizations that route packets between these network segments must agree to cooperate. If you take a minute to think about this you might be pleasantly surprised that everyone does cooperate and that the Internet does work, especially in today's highly polarized social climate where it seems that no one wants to cooperate for the greater good.

In any case, the IETF is responsible for developing and maintaining many of the protocols that are used on the Internet, including IP. The IETF publishes the standards in documents called RFCs, which stands for Request for Comments. RFCs are technical documents that contain information on how to implement various network protocols, network technologies, and network standards that make up the Internet. The RFCs are not meant for general use or explanation; instead, they serve as a blueprint for building and implementing various network technologies. They include detailed technical specifications, diagrams, and examples to help developers create and maintain interoperable systems. Some of the most well-known RFCs include:

RFC 791 - Internet Protocol (IP)

RFC 822 - Standard for the Format of ARPA Internet Text Messages (Email)

RFC 1034 and 1035 - Domain Name System (DNS)

RFC 1945 - Hypertext Transfer Protocol (HTTP/1.0)

RFC 8446 - Transport Layer Security (TLS 1.3)

These RFCs are just a few examples of the many standards and protocols that make up the Internet's infrastructure. You can find a complete list on several Internet sites, but the official repository run by the IETF can be found at <https://www.rfc-editor.org/>. While the IETF has serious responsibilities, they also have a well-developed although slightly geeky, sense of humor. There are several RFCs that are meant to be jokes and can be fun to read.⁵

One thing about the RFCs that many people find curious is the name "Request for Comments", instead of something more authoritative like "Standards" or "Technical Specifications and Requirements". The story behind the name "RFC" comes from the early days of the ARPANET, the precursor to the Internet. In 1969, Steve Crocker wrote the first RFC, which was a memo describing how to implement a simple host-to-host protocol for sending and receiving messages on the ARPANET. According to Dr. Crocker, the term "Request for Comments" was originally used as a bit of a joke. He and his colleagues were concerned that calling their publications "standards" might come across as overly presumptuous or arrogant, given that the technology they were working on was still very much in its infancy.

As Dr. Crocker recounted in a 2012 interview with the Internet Society⁶:

"The first RFC was simply an internal note, but when we decided to publish it, we needed a name. I personally was very uncomfortable with the term 'standard' because we were just a bunch of graduate students, and it seemed presumptuous to call what we were doing a 'standard.' I just suggested 'Request for Comments' partly as a joke, but partly also to say, 'Look, we're just putting this out there. We're not claiming it's perfect. We're not claiming it's a standard. We're asking for feedback.'"

⁵ <https://tangentsoft.com/rfcs/humorous.html>

⁶ <https://www.internetsociety.org/wp-content/uploads/2017/09/rfc-retrospective.pdf>

The term "Request for Comments" caught on and has since become synonymous with the process by which new Internet standards are developed and refined. Today, the RFC series includes over 9,000 documents, and remains an essential part of the Internet's infrastructure.

When the Internet was first starting the IETF developed a few "rules" that any network had to follow to connect to the Internet. If you were a network administrator that wanted to connect your network to the Internet, you had to promise to follow the IETF rules which include the following:

1. The Internet Protocol must be used.
2. Every device on the network must have a unique IP address, which for smaller networks can be obtained from an ISP, and for larger networks can be obtained from the official Internet registry, the Internet Assigned Numbers Authority (IANA), or a regional Internet registry (RIR). You'll learn the details about obtaining IP Addresses later in this section.
3. The Domain Name System (DNS), which is used to translate human-readable domain names such as google.com into IP addresses, must be used. To connect a network to the Internet, the network must have at least one DNS server that is able to resolve their DNS names into IP addresses. registry (RIR).
4. To connect a network to the Internet, the network must have a routing protocol in place that allows it to exchange routing information with other networks and routers on the Internet. You'll learn the details about routing later.

The main rule is that you must use IP as the network protocol. This isn't a rule in the sense that it's a legal standard, it's just that you must use the Internet Protocol get your network data through the Internet routers.

Once again, meeting these requirements is essential for ensuring that the network can communicate with other networks and devices on the Internet. Connecting a network to the Internet without meeting the necessary requirements can have various consequences, the main one being that devices on your network will be unable to share information with other devices on the Internet. If your network doesn't

follow the IETF standards and starts causing problems with other networks, the router(s) connecting your network to the Internet will shut down your connection. Unlike the rest of human society, the Internet realizes that the only way to work is to work together and cooperate. For this reason, all countries use the same set of IP numbers and recognize the Internet Assigned Numbers Authority (IANA) as the organization responsible for allocating IP addresses and managing the global IP address space. While some countries may have their own regulations and policies related to the use of IP addresses and the management of their own national networks, the underlying technical standards and protocols used for communication on the Internet are global and universal, they're not limited to the United States.

If a country, such as China, were to decide to ignore the Internet Assigned Numbers Authority (IANA) and set up their own set of IP addresses, it would create a parallel system that would not be compatible with the existing global Internet and devices and networks within that country would not be able to communicate with devices and networks on the global Internet. Such a move could have significant geopolitical and economic consequences, as it would likely lead to fragmentation and isolation of the Internet and prevent businesses and individuals from engaging in international commerce, research, and communication. So, even though other countries may have the technical capability to create their own set of IP addresses, they don't because doing so would likely have significant negative consequences and would not be compatible with the global nature of the Internet.

While you don't need to memorize any of the history of the Internet Protocol, there are a few things you should take away from this section. The first is the concept of how the Internet and the groups that control IP and the other protocols used on the Internet protocols came about. That is, they weren't formed by a company, and they weren't part of an international effort like the ISO. Instead, the Internet grew out of a US government research project. But even though the Internet started in the US it's now a global network that only maintains a global span because of a cooperative effort among all parties that use it.

The second point is a technical point, and this is that the Internet Protocol defines two main items, the format for network packets and a system of addresses called IP Addresses. So, let's quit talking about IP's history and talking about what it is in a vague sense, and start learning about IP Addresses and IP network packets.

Overview of the Network Layer Process and Components

Component Overview

Now let's start learning about what actually happens in the Network Layer, and the protocols and components used in the process. In this section you're going to get the 40,000 foot view of what happens in the Network Layer. The main thing to note is that the Network Layer is responsible for transmitting data between networks. That is, the Data Link Layer, ethernet frames, and MAC addresses can be used to send network data between devices on the same network segment, they aren't able to send data between devices on different networks. An analogy for this would be with snail mail, where your local postal carrier can deliver mail between houses and buildings in your town, but they can't deliver mail to houses and buildings in different counties or different states. Getting the mail to a different town requires something like the Internet Protocol or IP.

The main component used in the Network Layer is a protocol called the Internet Protocol, which defines three main things. It defines a format for the packets, called IP packets, that will be used to hold the data being transferred, a format for network addresses which are called IP addresses, and it defines something called netmask which is used to interpret how much of each IP address describes a network and how much of the address describes a specific device on the network.

The IP packet is much like an ethernet frame, in the sense that it's used to send network data and has a header section that contains things like source and destination addresses, and a data section. But when you look at the specifics IP packets and ethernet frames there are some obvious differences. One of the big differences is that IP packets will use IP addresses instead of MAC addresses.

IP addresses are made of 4 numbers, called octets, with dots or periods separating the numbers. Each octet number can range from 0 to 255, although the numbers 0 and 255 serve special purposes. An example IP address would be 12.234.1.98. Each device on a network connected to the Internet must have a unique IP address, but unlike MAC addresses which are assigned by vendor, the IP addresses are assigned to networks, so all the devices on the same network segment will have similar addresses. Each IP address actually contains two pieces of information, the network number, and a unique number assigned to each host on the network. This is kind of like MAC addresses, where the first 6 numbers

describe the vendor and the last 6 describe the specific device, but with two big differences. The first difference, as you just learned, is that the first part of the IP address describes a network instead of a vendor, and the second difference is that the portion of each IP address that describes the network can vary, and you can't tell how many of the octets describe the network without another piece of information. The network portion of an IP address may be the first octet, it may be the first two octets, or it may be the first three octets. Or in advanced cases, the network part of the IP address may be a fraction of one of the octets. The only way to know how to divide an IP address into the network portion and the host portion is by using a third piece of information called the netmask.

Another network component used by IP and the Network Layer is a router. A router is a networking device that forwards data packets between different computer networks. Hopefully you remember that switches and hubs connect devices to create a single network segment, while routers connect multiple networks or network segments. On most networks, there's a single router that connects the network to the Internet or in large organizations there will be a single router that connects a network segment to the organizations larger network. This is like in your home, where you have a single router that connects your home network to the Internet. Since the Network Layer is tasked with moving network data between networks, one of the key components in the process is this router, as this is the device that will handle any network packets that are being sent to devices on other networks. Whenever you configure the IP settings on a computer, you must provide the IP address of this router which is called default router. You should also note that in Microsoft terminology this device is called the default gateway, which isn't quite technically correct as far as network terminology goes, but that's what Microsoft calls it.

The last main component used by the Network Layer is the Domain Name System or DNS. This is the system that converts or resolves names like google.com to IP addresses. DNS is used because humans are much better at remembering names than we are at remembering long strings of numbers. You can think of DNS as being like a phone book or the contact list in your phone, but instead of converting human names to phone numbers, DNS resolves names to IP addresses. Any network that connects to the Internet needs to have access to two DNS servers that it can use to resolve names to IP addresses, and any computer connected to a network needs to be configured with the IP addresses of these two DNS servers.

Here's a summary of the protocols, components, and services used by the Network Layer:

1. IP Protocol – defines the format for IP packets.
2. IP Addresses – used to identify networks, and hosts on those networks.
3. Netmask – used to determine how to divide an IP address into the network portion and the host portion.
4. Default Gateway – IP address of the router used to transmit packets to other networks.
5. DNS Servers – Resolve host names to IP addresses.

Any computer or device that wants to communicate on an IP network, including the Internet, must have all these five things installed or configured. That is, the device must have the IP software installed, it must be assigned an IP address, it must have a netmask configured, it must have the addresses of two functioning DNS servers, and it must have the IP address of a default router (gateway). You'll learn how the IP address, netmask, default router, and DNS servers are configured later, but for now you just need to know that they must be configured on any device connected to a network using the Internet Protocol.

The following shows the network configuration information for a Windows based computer. You can see that the IP address and subnet mask have been set, and the IP addresses for the default gateway and DNS server have been configured. It also shows the MAC address, which has been read from the computer's NIC.

```
Physical Address: A0-02-A5-C6-C8-93
IPv4 Address:    192.168.1.98
Subnet Mask:     255.255.255.0
Default Gateway: 192.168.1.1
DNS Servers:     192.168.1.10
```

Process Overview

Now let's look at an overview of the process used by the Network layer, and how it uses the protocols, components, and services to package and deliver data across the network. As we go through this process remember that the network stack on the device will already know five things:

1. Its own IP address.
2. The netmask.
3. The IP address of two DNS servers.
4. The IP address of the network's default router/gateway.
5. Its own MAC address.

The first four items will be known because they have been set in the device's network configuration. And the MAC address will be known because it can be read from the firmware on the network interface card.

With that in mind, here's an overview of the process. Keep in mind that this is an overview and it's meant to show you how the various components work together to build and deliver network packets, and that the details of each step will be explained later.

1. The Network Layer receives a request to send data from the upper layers in the OSI network stack. This request includes the data to send, and the DNS name of the host to send the data to.
2. The Network Layer will build an IP Packet by:
 - a. Placing the data in the data portion of the IP Packet
 - b. Building the IP packet header, placing the source and destination IP addresses in the correct locations.
 - i. The source IP will be the IP address of this computer, which can be read from the computer's network configuration.
 - ii. The destination IP address will be the IP address associated with the DNS name of the computer we're sending the data to. To find this, the Network Layer must put the current IP packet on hold and create a new IP packet containing a DNS request.

- iii. This DNS request packet will be sent to the DNS server using the IP address from the computer's network configuration.
 - iv. The DNS system will respond with the IP address associated with the DNS name.
 - v. The original IP packet can now be completed by placing the IP address associated with the DNS name in the Destination IP Address field in the packet header.
3. The next step is to decide where to send the IP packet, either directly to recipient computer, or to a different device in the delivery chain. That is, the network stack needs to know if the recipient IP address is on the same network segment or not. If it's on the same network segment it can send the packet directly, but if it's on a different network segment it will have to send the packet to the default router/gateway. This decision is made by using the netmask to determine which numbers in each IP address describe the network, and then comparing the network numbers of the Source IP Address and the Destination IP Address. The steps in this process are:
- a. Read the netmask from the computer's network configuration. The numbers in the netmask will typically be 255s or 0s. Any of the four positions that contains a 255 will be part of the network number, and any position that contains a 0 will be a host number.
 - b. Compare the network portions of the Source and Destination IP addresses. If they are the same, the destination computer is on the same network, and the packet can be sent to it directly. If the network portion of the numbers are different, then the destination computer is on a different network and the packet must be sent to the default router for delivery.

In our example the network portion of the source IP address is: 192.168.2 and the network portion of the destination address is: 142.250.217. These two network numbers are different which means the IP packet needs to be sent to the default router.

4. After the decision on where to send the packet has been made, the IP Packet will be handed down to the Data Link Layer which will build an ethernet frame by:

- a. Placing the entire IP packet in the Data section of the frame.
- b. Placing the MAC address of this computer in the source MAC portion of the ethernet frame header.
- c. Placing the MAC address of the next computer in the delivery chain in the destination MAC. In this case, where the IP packet is being sent to google.com, the ethernet frame will be sent to the default router/gateway, so the default router's MAC address will be used as the destination MAC.
- d. The Data Link Layer checks its ARP cache for the MAC address of the default router. Remember that the ARP cache stores the MAC addresses associated with specific IP addresses, and the Data Link Layer knows the IP address of the default router since it must be set as part of the network configuration. If the MAC address of the default router isn't known, the Data Link Layer will use ARP to discover it. As you learned in the Data Link Layer, the ARP packet will be sent as an ethernet broadcast, with the default router's IP as the destination IP.

Internet Address	Physical Address	Type
192.168.1.1	3c-37-86-23-d0-c2	dynamic
192.168.1.3	00-15-99-11-41-49	dynamic
192.168.1.6	90-a8-22-93-99-6c	dynamic
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static

- 5. After the ethernet frame is built, it's handed down to the physical layer which converts it to the proper signals and transmits the signals across the network media.

Ok ... that's the overview, and as you can see there's a lot going on. Some of the steps in the process, like using the netmask or what DNS does may be a bit fuzzy now, but as you keep going and learn details of the various protocols and components, everything should start to make more sense.

IP Address Basics

In this section you'll learn the basics of IP addresses. IP addresses are at the heart of the Internet Protocol as they're used to uniquely identify a computer or device on the network and allow end-to-end delivery. IP addresses might seem just like MAC addresses as they're both unique addresses used to identify devices on a network, but there are a couple of significant differences.

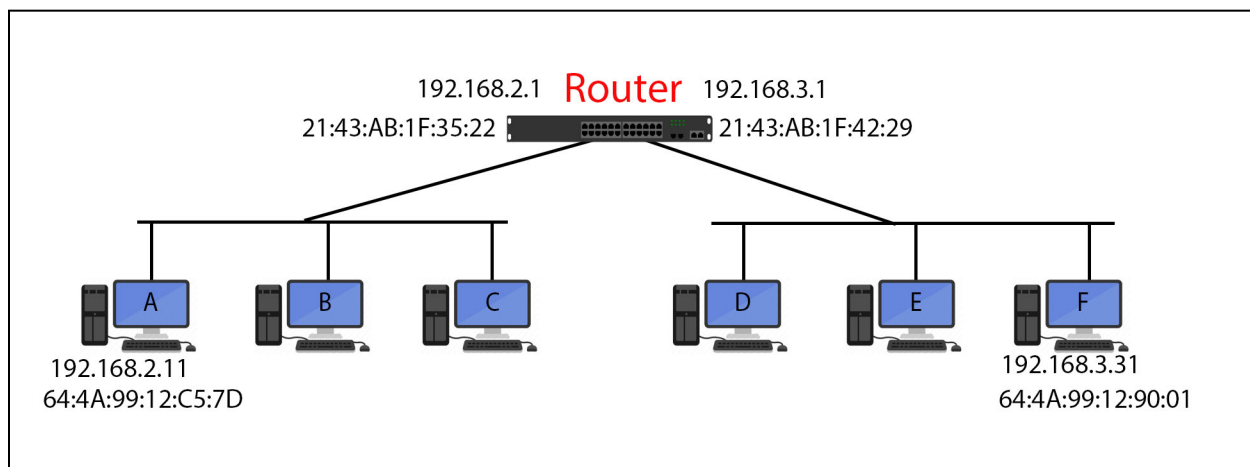
The first difference is that MAC addresses are burned onto a network card and cannot be changed. (Ok, this isn't technically correct, as the MAC address is burned onto an EEPROM that can be reprogrammed to change the MAC address, but this isn't a simple process.) IP addresses are configured for each computer and can be easily changed. You can think of this as the relationship between you and your phone number. You are like the MAC address, it doesn't matter where you go or what you do, you will always be glorious you. Your phone number is like an IP address, as it can always be changed, it's just a number that people use to call you, but it's not you. Of course, it might be painful for your friends if you change your number, but the point is it can be changed, while no matter what phone number you have you will be the same person.

You should note that just because IP addresses can be changed, it doesn't mean they are frequently changed. If you connect your computer to your home network and never move it, it will keep one IP address. But if you connect your computer to a different network, you will be assigned a new IP address. For example, if you take your computer to school or on the road and stay in a hotel and connect to a different network, it will get a different IP address. You'll learn the details of how the IP addresses are assigned when you connect to a different network later.

The second difference between IP addresses and MAC addresses is that IP addresses are used for end-to-end delivery, while MAC addresses are used for point-to-point delivery. Any computer connected to a

network will have both a MAC address and an IP address. The MAC address is used to transmit data from point-to-point, while the IP address is used to transmit data from end-to-end.

Let's look at an example of what this means, using the following figure. Say that computer A wants to send some data across the network to computer F. In this case computer A and computer F are on different networks, with a router connecting the two network segments. When we say end-to-end, we mean from computer A to computer F. But for the packets to get from end-to-end they need to go through the router, which means there will be a few point-to-point hops. The first point-to-point hop will be from computer A to the router, and the second point-to-point hop will be from the router to computer F. You'll learn the details of how this happens later, the main point to take away is that IP addresses are used for getting network data from end-to-end, while MAC addresses are used to move network data from point-to-point.



IP Address Versions

There are two versions of IP addresses, called IPv4 and IPv6. The IPv4 addresses were the original addresses and were originally just called IP addresses. As you'll learn there are only so many IPv4 addresses, over ~4.2 billion, which is a lot, but not enough if every computer and device connected to the Internet needs one. When it appeared that the Internet would need more addresses than IPv4 could provide work was started on IPv6, which mainly increased the address space to $\sim 3.4 \times 10^{38}$ but also made a few other improvements. Once IPv6 was released, we started using IPv4 to refer to the old version. But if you just mention IP addresses in most cases you can assume that this means IPv4, and if IPv6 is being used it will be called out by name.

Most of the rest of the discussion in this section will be based on IPv4, but once again I'm just going to call it IP. While IPv6 is being used in some situations, most organizations are still using IPv4. And the processes used by IPv4 will still be applicable in IPv6, so I think it will be simpler and less confusing to concentrate on IPv4. You will learn about IPv6 at the very end of this section, and that will be the only time, besides now, that we use IPv6 and not IPv4.

If you're the curious type and want to know why the only two versions are 4 and 6, you can look the explanation up on the Internet or read <https://www.ipxo.com/blog/what-happened-to-ipv5/> If this site no longer exists you can do your own search for something like IPv5.

Anatomy of an IP Address.

Now it's time to learn what an IP address looks like, and how this information is used. IP addresses are 4 part numbers, with each of the numbers separated by a dot or period. Each individual number can range from 0-255, although you never see a 0 in the first number, and 255 has a special meaning so it isn't used in a normal device address.

173.143.229.111
0-255 . 0-255 . 0-255 . 0-255

Here are a few examples of valid and invalid IP addresses. See if you can find the problems with the invalid examples.

Valid

182.34.29.100

192.168.1.1

12.13.14.15

254.1.31.254

Invalid

182.34.29

192.168.1.1.2

12.13.14.355

255.1.31.254

182.3A.29.17

173.143.229.111

1010 1100 . 1000 1111 . 1110 0101 . 0110 1111

An IP address is really 4 8-bit binary numbers, but they're easier for humans to read and understand if we convert the binary numbers to decimal numbers.

The reason the numbers range from 0-255 is because they're actually 8-bit binary numbers that range from 0000 0000 to 1111 1111. But because binary is so awkward for humans to deal with, we convert the numbers to decimal, which results in the 0-255 range. To work with IP addresses, you will need to be able to convert numbers between binary or base 2, and decimal or base 10. If you don't know how to do this, you can use the video tutorials I've created or find other tutorials on the Internet. You can also make use of a calculator, which will do the conversion for you, but if you're planning on a career in cyber security or in any aspect of computer science, you'll find that this is a necessary skill, so I suggest you take the time to learn how to work in binary on your own without the use of a calculator.

Each number is also referred to as an octet, since it's 8 bits. Using this terminology, we would say that an IP address is made up of 4 octets. You'll often hear portions of an IP address referred to as the first octet, which would be the number before the first dot, or the first two octets which would be the first two numbers.

Sidebar on Counting in Binary

If you want to earn your Junior Geek badge one thing you need to be able to do is use binary or the base 2 number system. Before we jump into binary, we'll do a quick review of base 10 or the decimal number system you're already familiar with to provide you with a perspective.

In base 10 the places in a number are assigned values. That is, the right most numeral is in what is referred to as the 1's place. Moving left, the next numeral is in what's referred to as the 10's place, followed by the 100's place, followed by the 1000's place, etc. Each place value is calculated by raising the number base, in this case 10, to a power starting with 0 on the right and increasing the power by 1 for each position as you move left. That is, the value of the first place is 10^0 , the value of the second place (moving left) is 10^1 , the value of the third place is 10^2 , and the value of the fourth place is 10^3 .

10^3	10^2	10^1	10^0
1000's place	100's place	10's place	1's place

Remember that any number to the 0th power is 1, and any number to the 1st power is itself.

Calculating the total value of a number is done by multiplying the numeral in each place by the place value, and then summing the result of all the multiplications. For example, if we had the number 4378 the total value would be:

$$(4 \times 1000) + (3 \times 100) + (7 \times 10) + (8 \times 1)$$

4	3	7	8
10^3	10^2	10^1	10^0
1000's place	100's place	10's place	1's place

Sidebar on Counting in Binary (Continued)

The point of this isn't to tell you what 4378 is. I know you know what it is, even without calculating $(4 \times 1000) + (3 \times 100) + (7 \times 10) + (8 \times 1)$. We can all pretty much look at numbers in base 10 and tell exactly what they are. The point of this to remind you how number systems work, as binary or base 2 works exactly the same way with a few differences.

In binary or base 2 there are only two numerals, 0 and 1. This makes binary very useful when working with electronic devices such as computers as 0 can be used to represent no voltage or a switch that's turned off, and 1 can be used to represent a voltage or a switch that's turned on.

The process for counting in binary is best demonstrated by repeating the process we used for figuring out the place values for decimal numbers, except in this case each place is going to be a power of 2 instead of a power of 10. The following diagram shows the values of each place in a 4-digit binary number. Once again remember that any number to the 0th power is 1 and number to the first power is itself. After that you can find the value of each place by multiplying the value to the right by 2.

2^3	2^2	2^1	2^0
8's place	4's place	2's place	1's place

Counting in binary is pretty simple, it just gets cumbersome quickly since there are only two numerals. The first number is 0 followed by, just like in decimal. When we get to 2 the binary equivalent is 102. This is because we've run out of numerals, and just like in decimal when we hit 9, the next thing to do is to add a 1 to the next place and set the values in all the other places to 0. In decimal adding 1 to 9 gives 10, or adding 1 to 999 gives 1000. In binary adding 1 to 1 results in 10 and adding 1 to 111 results in 1000. The following shows the decimal numbers from 0 to 15 and their binary equivalents.

Sidebar on Counting in Binary (Continued)

Decimal	Binary
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Converting between binary and decimal is also done using the same general process used for decimal numbers. That is, first multiply the number in each place by its place value, and then sum the results of all the multiplications. For example, 0110_2 would be $(0 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) = 6_{10}$.

0	1	1	0
2 ³	2 ²	2 ¹	2 ⁰
8's place	4's place	2's place	1's place

Since IP addresses contain 8 bits in each place, you'll need to have 8 places, with the left most place having the value of 2^7 or 128 and the right most having the value 2^0 or 1. The following shows all the place values for an 8-bit number.

128	64	32	16	8	4	2	1
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰

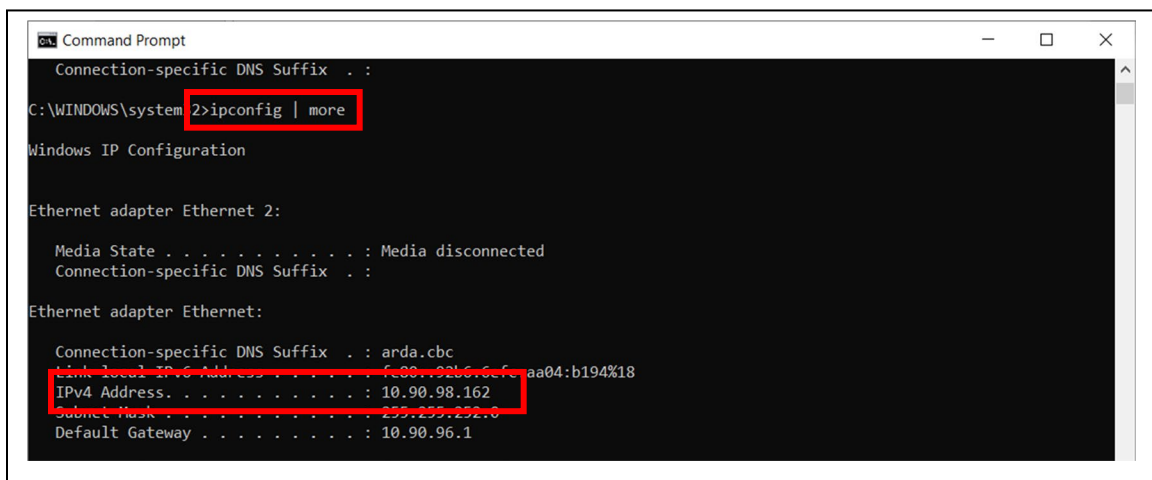
How to view a device's IP address

Now let's show you how to view a device's IP address. Every device connected to an IP based network needs to have an IP address. And, since the Internet is an IP based network, any device connected to the Internet will have an address, even your phone or home computer will have one if they're connected to the Internet through your home network.

I'll show you how to view the IP address on a Windows based computer, a Linux computer, and an Android phone, since I have those types of devices.

Windows: On a Windows computer the `ipconfig` command is used to display network settings, including the IP address. To run the `ipconfig` command, follow this process:

1. Start the Command Prompt application by going to the Windows start bar (or search area) in the lower left, and type **cmd**.
2. Windows will display the Command Prompt app. Click on this to open a command window.
3. Place the cursor in the Command Prompt window and type **ipconfig**. If you get more than one window's worth of data type **ipconfig | more**. The **|** character is typically located above the <enter> key. This will display something that looks like the following:
4. The IP address for the computer will be displayed in the IPv4 Address field. In this figure the IP address is 10.90.98.162



```
Command Prompt
Connection-specific DNS Suffix . : 
C:\WINDOWS\system 2>ipconfig | more
Windows IP Configuration

Ethernet adapter Ethernet 2:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix . : arda.cbc
    Link-local IPv6 Address . . . . . : fe80::2306:6c:feaa04:b194%18
    IPv4 Address. . . . . : 10.90.98.162
    Subnet Mask . . . . . : 255.255.252.0
    Default Gateway . . . . . : 10.90.96.1
```

Linux: On a Linux computer the `ip a` command is used to display network settings. To run the `ip a` command, follow this process:

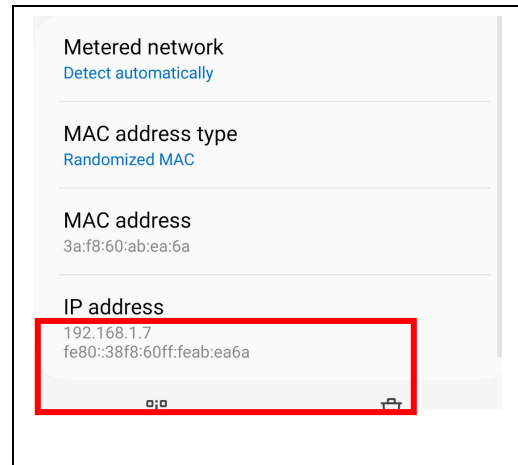
1. Get to the Linux command line, by either logging in to an account that starts with the command line, or logging in to an account that uses a GUI and then starting a Command Window.
2. Type the `ip a` command.
3. The IP address for the computer will be displayed under the `eth0` interface, as the first part of the `inet` field. In this figure the IP address is 192.168.34.19

```
[root@balrog ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:32:8e:a6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.34.19/24 brd 192.168.34.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe32:8ea6/64 scope link
        valid_lft forever preferred_lft forever
```

Displaying the IP address on a computer running Linux.

Android: This is another process you don't need to know, but I thought I'd include it in case you're curious. To see the IP address on an Android phone, follow these steps:

1. First, ensure the phone is connected to a network.
2. Go to Settings / Connections / Wi-Fi
3. Select the Gear icon by the Current Network
4. Select View More at the bottom of the screen.
5. Scroll down until you see the IP address.



Apple Devices: If you have an Apple Mac computer the process is very similar to the Linux computer since the MacOS is a version of Linux. If you have an Apple phone, I feel sorry for you, just kidding but not really, you can find tutorials on the Internet that will show you how to find the IP assigned to your phone. Just remember, your portable devices will only have an IP address if they're connected to a network.

Ok, that's it for the basics of IP addresses. You'll learn more about how they're used and how they're assigned a little later in this section.

Special IP Addresses

In this section you're going to learn about some special IP addresses. These include a few groups of IP addresses that are never assigned to individual devices as they are reserved for special functions. These include things called the loopback address, broadcast addresses, and multi-cast addresses. You'll also be introduced to blocks of IP addresses called private or non-routable addresses, and then learn the details of how they are used later in this chapter.

The loopback address is used to allow a device to send data to itself, essentially looping the data back to the sending device without sending any packets across the physical network. This is done by using the IP address 127.0.0.1, which is reserved specifically for this purpose. The loopback address is typically used for testing a network stack because packets sent to this address go all the way down the network stack to the Data Link Layer but are not sent to the Physical Layer. Instead, the Data Link Layer recognizes the packets are destined for the same host that sent them and sends the packets right back up the stack. If

the packets can go down the network stack and back up again it's a sign that the network configuration is correct. But if the packets can't be sent down the network stack and back then it's a good indication that something is wrong with the network configuration and needs to be corrected before checking any network cables or switches or routers outside of the device being tested.

Non-routable addresses are also known as private IP addresses, and they are reserved for use within private networks. As the name implies packets being sent from or to one of these addresses will not be passed by any router and will never be transmitted across the Internet. If the packets need to be delivered outside of the private network they will be forwarded to the Internet by the router, but first the router will change the source IP address so that the packet looks like it's coming from the router. This process is called Network Address Translation or NAT.

You can think of these as being like internal phone numbers or phone extensions used in a large organization. People inside the organization's buildings can call these numbers but the phone extensions cannot be reached from outside the organization. If someone inside the organization makes a call to the outside world from one of these phones the internal number will be hidden, and it will look like the call is coming from the organization's main phone number. Non-routable IP addresses are used almost all internal networks including home networks. That is, every business, government organization, educational institution etc. will use non-routable IP addresses on their internal networks. This has been done to free up the usage of most of the routable IP addresses and for the time being it's solved the problem of the limited number of addresses available in IPv4.

Non-routable IP addresses are any that start with the following octets:

10.0.0.0

172.16.0.0

192.168.0.0

Broadcast addresses are used to send IP packets to all devices within a specific network segment. You can think of a broadcast as being in a meeting with several people and saying "hey EVERYONE, please listen to this". If the IP network number is known the IP broadcast will 255 in the host portion. For example, on the 192.168.1.0 network the broadcast address would be 192.168.1.255. Or, if the IP

network number isn't known, the IP broadcast could be sent to 255.255.255.255 which ensures every device on the network segment will see it. When an IP broadcast is transmitted, every device on the network will see the broadcast and then send the IP packet further up the network stack to see if a network application should respond or not.

This might sound very similar to the ethernet broadcast which uses the MAC address of FF:FF:FF:FF:FF:FF, because it is. Both types of broadcasts are used to send data to every device on a network. The difference between the IP broadcast and the ethernet broadcast is the way they're used. This is most easily explained by using an analogy, of asking for information from a group of people in a crowded room. In this case let's assume you can ask for information from a specific person by using their name or their phone number. If you don't know the name of the person you want to speak with but do know their phone number, you can discover their name by shouting out their phone number. For example, if you know you want to speak to the person with the phone number 547-0511, you can shout "Would EVERYONE please listen, I'm looking for the person with the phone number 547-0511". Everyone in the room will hear you, and if someone has that phone number they'll respond. Asking EVERYONE to listen would be like a MAC broadcast.

Now let's say there's one person in the room who can provide a special service, and for this example let's say they can serve cheesecake. There's only one person in the room who can give you cheesecake, you just don't know who they are or their phone number. To find this person you yell "Would EVERYONE please listen. I'm looking for the person with ANY phone number that has cheesecake". Everyone in the room will hear your first sentence, which is like the MAC broadcast. Everyone will then pay attention to your second sentence which is directed to anyone with a phone, which means everyone will think about it, like the IP broadcast, and check to see if they have cheesecake. While everyone hears both things you shout, the only person that will reply is the person with the cheesecake.

Here's the difference between the two types of broadcasts at a technical level. The MAC broadcasts are processed at the Data Link Layer, which reads the broadcast data and passes it up to the Network Layer. The Network Layer looks at destination IP address, and if it matches the computer's IP address it passes the data up the network stack to the programs using the network. This means a MAC broadcast that is sent to a specific IP address will only be passed up the network stack on the one computer on the network with that specific IP address. An IP broadcast will be passed up the network stack by every

computer on the network, and the network programs on each computer will have to determine whether they should respond or not. This decision is made based on something called a port number, which you'll learn about in the sections on the Transport and Session Layers.

Oh, and I guess this might not be obvious, but there's no way to have an IP broadcast without an ethernet or MAC broadcast. That is, any IP broadcast must be sent inside an ethernet frame containing a MAC broadcast. If the ethernet frame doesn't use a MAC broadcast it will only be sent to a single device, which defeats the purpose of trying to get all devices to read the IP packet.

IP broadcasts are used for tasks such DHCP which you'll learn about below, and other types of network activities where a device needs to discover which device is offering a service and doesn't know who to talk to. A long time ago the .0 address was also used as a broadcast, but today .0 is usually used to describe a network. For example, all the devices with IP addresses ranging from 192.168.1.1 to 192.168.1.254 are said to be on the 192.168.1.0 network.

One last thing to note about IP broadcasts is they will not pass through a router. Hopefully you see the sense behind this.

Multicast addresses are similar in function to broadcasts, as they're used to send IP packets to multiple devices for things like zoom meetings, multimedia streaming, or online gaming. But unlike broadcast addresses, which deliver data to all devices within a network segment, multicast addresses target specific groups of devices interested in receiving the data. The addresses in the range 224.0.0.0 to 239.255.255.255 are reserved for multi-casting.

The last thing to learn about in this section is another term, which is unicast. Unicast simply means an IP address that's being sent to a single device. The three terms, unicast, multicast, and broadcast cover the entire range of possible IP transmissions. That is, broadcast packets are sent to every device on a network, multi-cast packets are sent to some of the devices, and unicasts are sent to a single device.

IP Packets

This section is another in the series on the Network Layer and the Internet Protocol (IP). In this video you're going to be introduced to IP packets and their format. There's a lot of detail involved so in this section you'll learn what you need to know, the basic things you need to know about IP packets, while the excruciating details are presented later where you can take a deep dive into them if you wish.

AN IP packet is a lot like ethernet frame in the sense that it's a package that will contain some data and have fields that contain addressing information. But the specific fields in an IP packet are different, and the purpose is also different, where IP packets are used to move data from end-to-end while ethernet frames are used to move data from point-to-point.

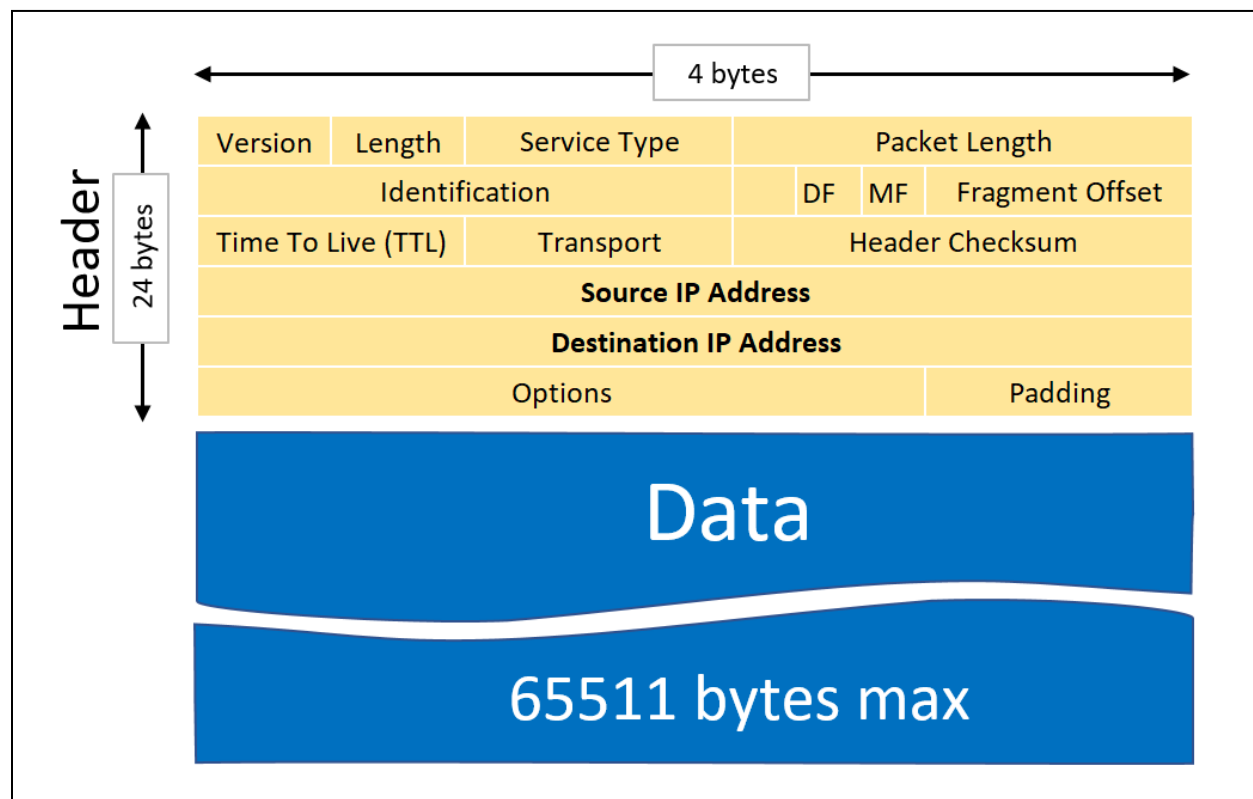
Snail mail or UPS packages make a good analogy for explaining the IP rules for creating network packets. With snail mail, anytime you want to send someone a letter you must put it in an envelope and address the envelope correctly. There are sets of rules for how many pages you can place in a single envelope, and for where and how the sender and recipient addresses must be written⁷. The addressing rules state that the recipient address must be in the center on the front of the envelope, and the sender's address must be in the upper left corner. The addressing rules also state that the format of each address must be the sender's name on the first line, street address on the second line, with the city, state, and zip code on the third line. There is also a rule about how much you can fit in a single envelope, or to be more correct there's a rule that says you can only send up to 1 ounce with a single stamp, which is typically 4-6 pages of normal weight paper and a standard envelope. For our analogy we'll assume that you're always going to use a normal size envelope, and let's say you can only put 5 pages in each envelope. So, if you want to send more than 5 pages, you'll have to break your message up into multiple envelopes, and number the envelopes so the person you're sending them to knows how to reassemble them in the correct order.

The task of building and addressing IP network packets falls to the Network Layer, but instead of placing letters inside envelopes the Network Layer takes any data from the upper layers of the network stack and place it inside IP packets. Each IP packet is divided into two parts, a header section and a data

⁷ <https://mystampguide.com/mail-pages-paper-with-one-stamp/>

section. The header section is like the outside of the snail mail envelope, and the data section is like the envelope that holds the actual message.

Now let's take a closer look at the structure of an IP packet, which is shown in the following figure. Each IP packet can be up to 65535 bytes in total and begins with a header section that requires 24 bytes followed by a data section which can hold as little as 1 byte of data or as much as 65511 bytes of data.



Taking a closer look at the header shows that it stores several different pieces of information. The most important fields are the Source IP Address and the Destination IP Address which ensure the packet can be delivered and replied to. Here's a quick description of some of the more important fields. (Complete details of all the fields are provided below.)

Version (4 bits) - This is set to either 4 or 6 and describes which version of the IP protocol 4 or 6.

Header Length (4 bits) and Packet Length (16 bits) – The Header Length is the number the of 32-bit (4-byte) words in the header, while the Total Length field specifies the total length of the IP packet in bytes, including both the header and data sections. The maximum value for the Total Length field is 65,535 bytes, which is the maximum size of an IP packet.

Identification (16 bits) and Fragment Offset (13 bits) – These fields are used when multiple packets are required to send a message. When a message is broken into pieces or fragments all the fragments will have the same ID which is a number between 0 and 65535. And you can think of the Fragment Offset as being like the sequence number for each piece so they can be reassembled in order. For example, if the data requires 5 IP packets, the Fragment Offset in the first packet would be set to 1, the Fragment Offset in the second packet would be set to 2, etc. This way the Network stack on the recipient computer can put the data back together in the correct order. Note that this explanation of the Offset isn't technically correct, but it's close enough while allowing for a simple explanation.

Time to Live (TTL) (8 bits) - This field is used to limit how many point-to-point hops a packet can make, to handle the rare cases when packets to get "lost" and wander in circles between routers. The TTL field is typically set to 64 and decremented by one by each router that processes the packet. The packet is discarded if the TTL field reaches zero.

Header Checksum (16 bits): This field is used to detect errors in the IP packet header. The checksum is calculated over the entire IP packet header, and if any errors are detected the packet is discarded.

Now let's look at the amount of data that can added to an IP packet. An IP packet can hold as little as 1 byte of data or as much as 65511 bytes. 65511 might seem like a weird number, because it is, but there's a reason why this is the maximum amount of data. The explanation is that the field in the header that holds the length of the packet is 2 byte or 16-bit binary number. This means the largest the number can be is 2^{16} or 65535. You might think that this means that the packet could hold 65535 bytes of data, but the length also includes the header. And since the header is 24 bytes, the maximum amount of data that can be placed in the packet must be reduced by 24 bytes, which means the max amount of data is 65511 bytes.

If the original amount of data is larger than 65511 bytes, the Network Layer will break the data up into multiple IP packets. The Network layer will keep track of the sequence of packets and store the sequence number for each packet in the Fragment Offset field in each IP packet's header.

The last thing to address is the size difference between ethernet frames and IP packets. The total maximum size of an IP packet is 65535 bytes, while the maximum size of an ethernet frame is 1500 bytes. The Data Link Layer will treat each IP packet as a chunk of data and try to stuff entire IP packet inside the data section of an ethernet frame. If the IP packet is larger than 1500 bytes, it will be divided into 1500 bytes chunks by the Data Link Layer each chunk will be placed inside an ethernet frame, and then all the ethernet frames will be sent to the next point in the delivery chain. Once all the frames are received, the Data Link Layer on the recipient computer will reassemble the IP packet and pass the entire IP packet up to the Network Layer on the recipient computer. One of the great things about the OSI model is that the Network Layer doesn't really need to be concerned about this size difference, it can just pass each IP packet down to the Data Link Layer and let the code at that layer handle the delivery and reassembly on the recipient computer.

That's a quick overview of the structure of an IP packet along with the header fields that everyone should know, as well as an explanation of IP packet size.

Subnetting and Netmask Basics - Delivery of IP Packets

Once the IP packet is built, the next thing the Network layer needs to do is determine whether the destination IP address is on the same network segment or a different network segment. It needs this information to tell the Data Link Layer whether to send the packet directly to the destination, or if the Data Link Layer should send the network packet to the default gateway/router. That is, the IP packet contains the end-to-end addressing, but not the point-to-point addressing. The Network layer will decide the next point in the delivery process using another piece of information called the netmask.

You just learned about IP addresses and how they are used to identify computers and devices on a network. Even though each IP address looks like a single number it actually has two parts. Part of the address, called the network portion, describes which network a device is on, and the other part, called the host portion, identifies unique computers or devices. Determining which part of each IP address is

the network portion and which part is the host portion requires using another number called the netmask. In this section you'll learn about network and host portions of an IP address, and how the netmask is used to specify which part of an IP address identifies the network and which part identifies the host.

Let's start by looking at how an IP address is used to specify two things, a network segment, and a specific host on the network segment. In some ways this is like a land line phone number which can have at two or more parts, depending on how you count. With a complete land line phone number, there will be several parts, a country code, an area code, the prefix or exchange number, and finally the line number. When you dialed a phone number with the phone system in the 1930s and 1940s, the country code, area code and exchange would get your phone call connected to a human operator in a specific geographic location, sitting at a switchboard. You would then ask the operator to connect your call to a specific line, and the operator would physically move some a cable to patch your call from the incoming line to the line you're calling⁸. All the phones in a neighborhood shared the same area code and exchange, but would have a unique line or circuit number, which was the last 4 numbers. The human operators were eventually replaced by mechanical and electronic switching devices, but the concept and the parts of the phone number remains the same, at least for land line numbers. That is, the first part of the number identifies a specific country, area of the country, and switchboard or exchange, while the last 4 numbers identify a specific line connected to that switchboard. In other words, each phone number contains multiple parts.

1	(626)	831	-	9333
country	area code	prefix or exchange		line

(You should definitely call this number)

IP addresses are similar to phone numbers in that they contain two pieces of information, part of the IP address identifies a specific network segment and part of the address identifies a specific device on that

⁸ <https://www.youtube.com/watch?v=r46zXIN3Nus>

network segment. In other words, all devices on the same network segment will share the same network number. They'll need different host numbers, but they must all share the same network number.

While the two parts of an IP address are like a phone numbers, there are two big differences in the way IP addresses and phone numbers divide their information and what we can learn from each part. The first big difference is that with land line phone numbers we can use the country code, area code and exchange to identify the geographic location of a phone. We can't do this with cell phones since they're portable, but we can with land line phones. With IP addresses, the network portion of the number does NOT tell us anything about the location of the network. There are ways we may be able to look this up, but it's not built into the system like it is with phone numbers.

The second big difference is that phone numbers are consistent in the way they divide the overall number, while the portion of the IP address that describes the network and the part that describes the host can vary. In land line phone numbers, the part that identifies a specific phone line will always be the same part of any phone number. It's different with IP addresses, where the part of the IP address that specifies a specific computer on a network segment can vary. And this is what we need and use the netmask for.

With IP addresses the network portion may be the first number or first octet, or it may be the first two octets, or it may be the first three octets. For example, if we look at the IP address 22.56.79.115 the network portion could be 22, in which case the part that identifies a specific computer would be 56.79.115. Or the network portion could be 22.56, in which case the host portion would be 79.115. Or the network portion could be 22.56.79, in which case the host portion would be 115.

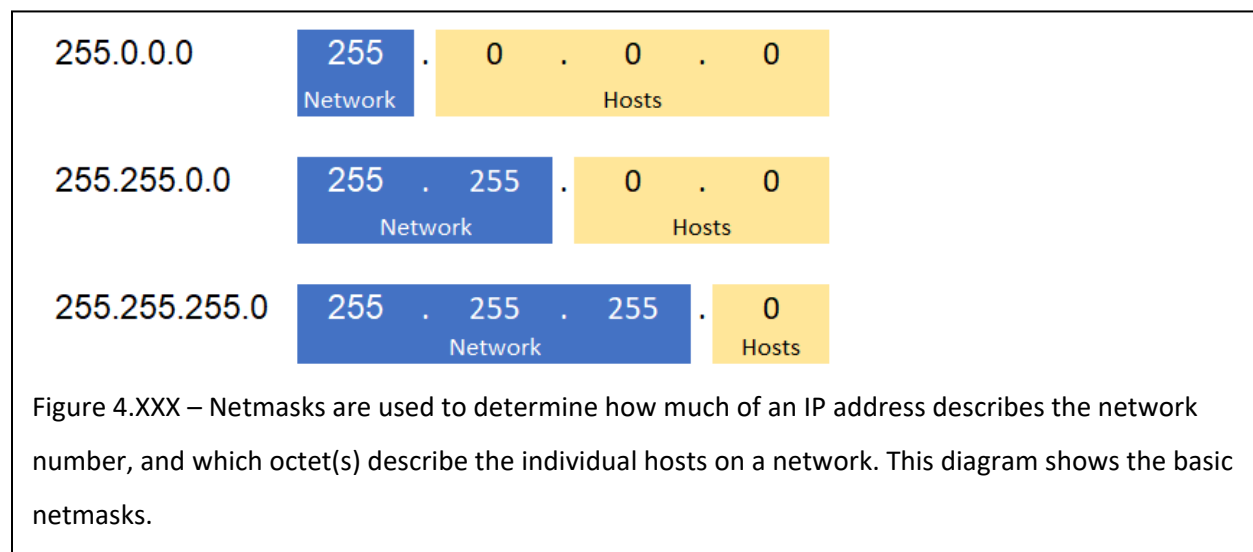
22.56.79.115

22.56.79.115

22.56.79.115

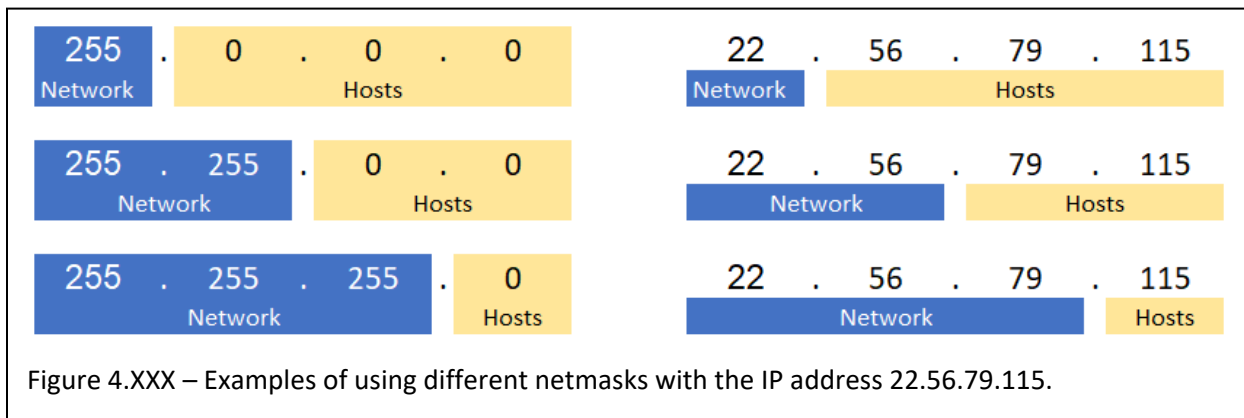
The network portion is always made up of numbers on the left, but you can't tell just by looking at the IP address. To make this determination you'll also need another number called the netmask. The netmask acts like a template, and it shows how to divide an IP address into its two parts.

Simple netmasks are also made up of 4 numbers, separated by dots or periods. However, in a simple netmask the only numbers used are 255 or 0. For example, 255.255.255.0 would be a valid netmask. To apply the netmask, just find portion of the netmask that contain the number 255. These octets will be the ones in the IP address that describe the network. Any octets in the netmask that contain a 0 will correspond to the host portion of an IP address.



For example, if we know an IP address is 22.56.79.115, and the netmask is set to 255.255.255.0, we know that the first three numbers describe the network. Which means in this example the network portion of the IP address would be 22.56.79. This also means that the host portion of the IP address would be the last octet, or in this case 115.

Here's another example using the same IP address, 22.56.79.115, but this time with a netmask of 255.255.0.0. Since the netmask has 255s in the first two octets, the network portion of the IP address is 22.56, while the host portion of the IP address is 79.115.



Here are a couple of things to note. The first is that the 255s or network portion will always be on the left and the 0s or host portion will always be on the right. That is, 255.255.255.0 and 255.255.0.0 are valid netmasks, while 255.0.0.255 or 0.0.255.255 are invalid. The second thing to note is that a netmask can actually contain numbers other than 255, which you'll learn about below in the section on advanced netmasks, but we'll stick with 255s now as it makes explaining the process of using a netmask simpler.

Hopefully you can see that the netmask is a critical piece of information for any IP network implementation, and that building and using a network with more than a single segment would be impossible without knowing the netmask. For this reason, the netmask, along with the IP address, the DNS server IP address, and the default router or default gateway are the 4 pieces of information that **must** be configured on any device that wants to connect to an IP based network.

```
Physical Address: A0-02-A5-C6-C8-93
IPv4 Address:    186.18.11.98
Subnet Mask:     255.255.255.0
Default Gateway: 186.18.11.1
DNS Servers:     212.8.91.10
```

You'll learn more about netmasks later, but for now, you just need to know the following:

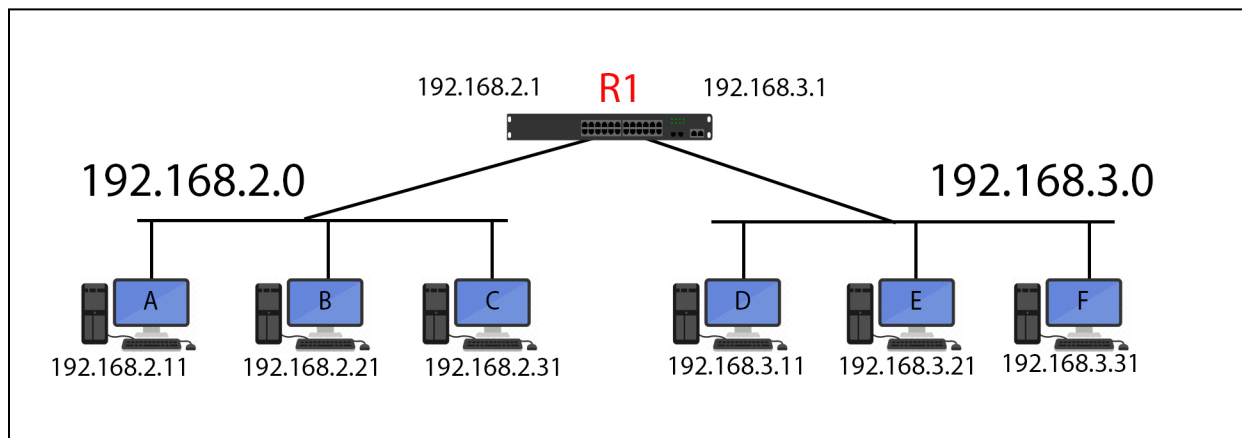
1. IP addresses are used to deliver network packets from end-to-end.

2. Each device connected to a network must have a unique IP address.
3. IP addresses are made up of 4 numbers or octets, with each number ranging from 0-255.
4. The only way to know how an IP address is divided into the network portion and the host portion is to use the netmask.
5. The numbers assigned to each octet in a netmask must be 255 or 0. Any octet that contains a 255 will describe the network and must be on the left, and any octet that contains a 0 describes the host and must be on the right.

Using the Netmask

In this section you'll learn the process for using the netmask to determine whether two IP addresses are on the same network segment or different segments. This is an important part of the overall networking process, as it allows the Network Layer to tell the Data Link Layer where to send the ethernet frames. That is, the netmask is a critical piece in the process used for determining where to send the IP packet in the point-to-point transmission chain.

Let's use the following diagram to illustrate this decision-making process, and how the netmask is used. The figure shows two network segments connected by a router, with computers A, B, and C on Network 1, and computers D, E, and F on Network 2. As you've learned, if computer A wants to send data to computer B, the data link layer will address the ethernet frames using computer A's MAC address as the source MAC and computer B's MAC address as the destination MAC. However, if computer A wants to send data to computer E, the data link layer on computer A will have to address the ethernet frames to the router's MAC. The question or problem that the netmask solves, is how does computer A's network stack decide whether or not the destination computer is on the same network segment?



This decision is made by comparing the network portion of the IP addresses of the source and destination devices. If the network portions of both the source and destination show the devices are on the same network, then the ethernet frames will be sent directly, but if the network portions show the devices are on different networks the ethernet frames must be forwarded through the router.

But once again, how do we know which portion of the IP address is the host portion? This is where we must know the netmask, as it tells us exactly how to divide the IP address into the network portion and the host portion.

Let's go through a few examples using a netmask of 255.255.255.0, which specifies that the network portion of the IP address is the first three octets.

If Computer A wants to send network packets to Computer E, it will compare its IP address of 192.168.2.11 with Computer E's IP address which is 192.168.3.21. Since the netmask is 255.255.255.0 Computer A's Network layer will compare the first three octets of the IP addresses. For Computer A this is 192.168.2 and for Computer E it's 192.168.3. Since the network numbers are different, Computer A will tell its Data Link Layer to send the ethernet frame to the router, instead of sending the ethernet frame directly to Computer E.

Now let's look at what happens if Computer A wants to send network packets to Computer C. In this case Computer A will compare its IP address of 192.168.2.11 with Computer C's IP address which is 192.168.2.31. Since the netmask is 255.255.255.0 Computer A's Network layer will compare the first three octets of the IP addresses. For Computer A this is 192.168.2 and for Computer C it's 192.168.2.

Since the network numbers are the same, Computer A will tell its Data Link Layer to send the ethernet frame directly to Computer C.

It's important to note that there's a difference between the IP Network Packet addressing and the ethernet frame addressing. The IP packet will always be addressed with the end-to-end IP addresses, while the ethernet frames will be addressed with the MAC addresses of the next point in the delivery process. Hopefully this makes sense, but if not, you'll get a detailed demonstration of this a little later.

Before we end this section, here are a few examples, where you can check to see if you are able to apply netmasks to determine whether two IP addresses are on the same network segment or different segments.

Example 1:

IP address 1: 192.168.1.10

IP address 2: 192.168.1.20

Netmask: 255.255.255.0

In this example, because the netmask is 255.255.255.0 the first three octets describe the network, and both IP addresses have the same numbers in the first three octets, 192.168.1. This means that both IP addresses belong to the same network segment.

Example 2:

IP address 1: 192.168.1.10

IP address 2: 192.168.2.20

Netmask: 255.255.255.0

In this example, because the netmask is 255.255.255.0 the first three octets describe the network. When we compare the first three octets, we find that the last number is different between the two IP addresses, 1 for the first address and 2 for the second address. This means that the IP addresses belong to different network segments.

Example 3:

IP address 1: 192.168.1.10

IP address 2: 192.168.4.20

Netmask: 255.255.0.0

In this example, the netmask is 255.255.0.0, which means that the first two octets are the network portion, and the last two octets are the host portion. This means that both IP addresses belong to the same network segment because their first two octets, 192.168, are identical.

Example 4:

IP address 1: 192.168.1.10

IP address 2: 10.3.46.20

Netmask: 255.255.255.0

In this example, the IP addresses belong to different network segments because their first octets are different 192 vs. 10. Because the first octet is different, we don't really need to even use the netmask. But if we go through the process, we find that the netmask is still 255.255.255.0, which means that the first three octets are the network portion, and the last octet is the host portion. Since the first three octets are different, the two computers are on different network segments.

Example 5:

IP address 1: 12.168.1.10

IP address 2: 12.230.76.20

Netmask: 255.0.0.0

In this example, the netmask is 255.0.0.0, which means that the first octet is the network portion, and the last three octets are the host portion. When we compare the first octets, we find that they're both set to 12, which means the two computers are on the same network segment.

Example 6:

IP address 1: 211.68.19.10

IP address 2: 211.68.19.156

Netmask: 255.255.255.0

In this example, because the netmask is 255.255.255.0 the first three octets describe the network. Both IP addresses have the same numbers in the first three octets, 211.68.19. This means that both IP addresses belong to the same network segment.

Example 7:

IP address 1: 10.2.78.1

IP address 2: 10.10.1.1

Subnet mask: 255.255.0.0

In this example, the subnet mask is 255.255.0.0, which means that the first two octets of the IP address represent the network portion. When we compare the first two octets of the two IP addresses, IP address 1's network portion is 10.2 while IP address 2's network portion is 10.10. Since these two network numbers are different the devices are on different networks.

Example 8:

IP address 1: 10.20.10.11

IP address 2: 10.20.59.166

Subnet mask: 255.255.0.0

In this example, the subnet mask is 255.255.0.0, which means that the first two octets of the IP address represent the network portion. When we compare the first two octets of the two IP addresses, we see that they are both 10.20, which means that they are on the same network segment.

Netmasks, Numbers of Hosts, and Broadcast Domains

In this section you'll take a closer look at the relationship between the netmask, number of hosts on a network, and the broadcast domain. This will help you see that even though it may be possible to have netmasks like 255.0.0.0 and 255.255.0.0, you'll never see these in practice.

As you've learned, for any IP address the netmask defines what portion of the IP address describes the network the device is on, and what portion of the address is the number identifying the specific host on

that network. Using a netmask of 255.255.255.0 means that the last octet can be used to identify hosts, which means that there can be 254 different possible host numbers. For example, if the network number is 12.45.67.0, devices on this network can be assigned host numbers 12.45.67.1 through 12.45.67.254.

The question we need to look at now is whether it's practical, or even possible to have 254 different devices on the same network. The short answer is that this is probably too many devices for a single network. The longer answer requires reviewing how networks are physically constructed and reviewing the concept of collision domains from ethernet and CSMA/CD.

Remember that a collision domain is the group of devices that share the same section of physical media and will see each other's network traffic. As more devices are connected and included in the same collision domain, the greater the chance that a collision may occur. Going back to IP addresses, if all 254 devices using the same network number are connected using the old ethernet cable or using a hub it means that every device will see all the network traffic, which also means that the odds of collisions will be very high. An online zoom meeting is a good analogy for this situation, as the chance of two people talking at the same time increases with each new person that joins the call. At some point, there will be so many people in the zoom meeting that it will be difficult for anyone to say anything with being interrupted. That is, having 254 people in a zoom meeting would almost certainly be too many. Determining an exact optimal number of participants would depend on how much each person spoke during the meeting, but assuming each person wanted to actively participate even 50 people would be pushing the limits.

One of the key factors to note about network devices and collision domains is that a higher number of devices is a problem when we're talking about using *hubs*, not *switches*. Remember that hubs pass all network traffic seen on any port to every other port on the switch. If switches are used, the switch will segregate the network traffic and only send traffic out a port if the ethernet frame is addressed to the MAC address of the device connected to the port. This reduces the collision domain to the single device on each port and practically eliminates any problems with collisions that can be experienced when hubs are used. When switches are used the number of devices that can use the same IP network number and be on the same network segment will be limited by two features of the switch.

The first feature is the number of ports on the switch. Most switches come with a fixed number of ports, starting with 8, 12, 24, and ranging up to 48. Each port on a switch can accommodate one network device, such as a computer or printer. Larger organizations may utilize stackable switches which allow multiple physical switches to be "stacked" together and function as a single logical switch. For example, it is possible to stack four 48-port switches to connect 192 devices and make it seem as if they were connected to the same switch.

Beyond the number of ports, another switch feature that limits the number of devices that can be connected to the switch is an architectural feature called the backplane capacity. Each switch contains hardware and software called the backplane that performs tasks like maintaining the table of MAC addresses connected to each port, deciding where to send each incoming frame, and sending each frame to the correct port. The maximum volume of network traffic each switch's backplane can handle will vary by switch, but there will be a maximum value. If the amount of network traffic exceeds the maximum value, it will lead to network congestion, delays, and reduced performance.

The two factors that impact the amount of network traffic the backplane must process at any time are the number of connected devices and the amount of network traffic generated by each device. To explain the amount of work that can be done by the backplane in a switch, let's use an analogy of a waiter or waitress working in a restaurant. Assume there's a single person who can handle taking customer orders, delivering food and drinks, and handling bills and taking payments. This wait person is like the switch backplane and the network traffic the backplane needs to deliver to and from the switch ports is like the things the wait staff needs to deliver to and from the customers.

The number of customers a single wait person can keep happy is affected by the number of tables in the wait person's section, and how much service is required by the customers at each table. The wait person can easily handle two or three tables if the customers require the average amount of attention and may be able to handle a dozen tables if the customers don't need much help. But the wait person could be overwhelmed by just a few tables if the customers order unusually large amounts of food and drink, or are demanding in other ways that require more time and attention than usual.

For our analogy, the switch backplane is like the wait person, the switch ports are like the tables and customers the wait person is assigned. The number of devices the back plane can serve without any

delays is affected by the number of ports and the amount of network traffic that needs to be transmitted to or from each port. If the device connected to a port is the web server for a large company, it could generate a lot of network traffic. But if the device is a computer that's only used a few hours each day, and mainly used for word processing, it won't generate as much network traffic. Like the food server, the backplane in most switches will be fast enough to handle normal amounts of network traffic without delay, but the backplane could have problems if too many devices try and send or receive large volumes of network traffic. Putting this in technical terms, let's assume we have a switch where the backplane has a capacity of 10 Gbps and this switch has 24 ports which can each run at 1 Gbps. If the devices on 10 ports transmit data at their maximum speed this will be 10 times 1 Gbps, or 10 Gbps which is the maximum rate the backplane can handle. But if devices on more than 10 ports try to transfer data at their maximum speeds the backplane won't be able to keep up and there will be delays.

The point of all this is that even though it may be physically possible to stack five 48 port switches to create a network with 250 devices, doing so will almost certainly degrade the overall network performance. Just the sheer number of devices alone could overwhelm the backplane even if none of the devices needs to transmit a large amount of network data. Returning to the restaurant analogy, the wait person could easily become too busy to provide timely service if they're asked to take care of 250 tables.

Getting back to the question whether it's practical, or even possible to have 254 different devices on the same network, this longer explanation shows why it's probably not practical. That being said, the netmask 255.255.255.0 is actually used quite often and it's probably the netmask you'll see in most homes and small offices. It's the most commonly used netmask because even though a netmask of 255.255.255.0 means it's possible to have 254 hosts on the same network it doesn't mean there has to be that many. So, in most cases, especially home and small office networks, you'll see a netmask of 255.255.255.0 but only a handful of connected devices.

And, if using a netmask of 255.255.255.0 isn't practical because it means there can be 254 hosts on the same network, then using a netmask of 255.255.0.0 with ~65500 hosts on the same network or using a netmask of 255.0.0.0 with ~17.6 million hosts on the same network are extremely impractical. These

netmasks are used to teach how netmasks work, and may have been used at the very start of the Internet, but today you'll never see them used outside of a test lab or classroom.

Classless Inter-Domain Routing (CIDR)⁹

Another way to specify the netmask uses something called Classless Inter-Domain Routing or CIDR notation. Using CIDR notation doesn't cause the netmask to behave any differently, it's just a more concise way to write an IP address and netmask. The CIDR notation combines an IP address and the netmask in a single string, with the IP address listed first, followed by a slash, then a single number which specifies number of network bits. To get from some number of bits to the 255's we've been using in netmasks remember that netmasks and IP addresses are really made up of 4 octets, where each octet is an 8 bit number. This means a CIDR number of 8 is saying the first octet or first 8 bits in the netmask are set to 255, a CIDR number of 16 is 2x8, which means the first two octets in the netmask are set to 255, and a CIDR number of 24 is 3x8, which means the first 3 octets in the netmask are set to 255.

Let's look at an example of 22.56.79.115/24. In this case the /24 on the end is the CIDR notation that specifies that the first 24 bits of the address describe the network. This means that for this IP address the network portion is 22.56.79.

Here are the basic subnet masks and their equivalent CIDR notations for an IP address 13.45.234.17.

	Netmask	CIDR	Network Address
13.45.234.17	255.0.0.0	13.45.234.17/8	13.0.0.0
13.45.234.17	255.255.0.0	13.45.234.17/16	13.45.0.0
13.45.234.17	255.255.255.0	13.45.234.17/24	13.45.234.0

The nice thing about the CIDR notation is that it conveys the same information as the IP address and a separate subnet mask in a much more compact form. That is, it's much easier to write "13.45.234.17/24" than it is to write "13.45.234.17 and 255.255.255.0"

Here are a few examples of comparing IP network numbers from two IP addresses using CIDR notation:

⁹ https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Example 1:

IP address 1: 199.16.1.100/24

IP address 2: 199.16.2.200/24

In this example, the /24 means that the subnet mask is 255.255.255.0, which means that the first three octets, or first 24 bits, of each IP address represent the network portion. When we compare the first three octets of the two IP addresses, 199.16.1 and 199.16.2, we see that they are different. This means that the two IP addresses are on different network segments.

Example 2:

IP address 1: 212.57.1.100/16

IP address 2: 212.57.19.200/16

In this example, the /16 means that the subnet mask is 255.255.0.0, which means that the first two octets (16 bits) of the IP address represent the network portion. When we compare the first two octets of the two IP addresses, we see that they are the same 212.57, which means that the two computers are on the same network segment.

Example 3:

IP address 1: 10.20.30.142/8

IP address 2: 10.168.231.11/8

In this example, /8 means the subnet mask is 255.0.0.0, which means that the first octet, or first 8 bits, of each IP address represents the network portion. In both IP addresses the first octet is 10, which means that they are on the same network segment.

Default Gateway (Router) Basics

The next component to learn about is the default gateway or default router. You'll first learn the general function that routers provide, and then learn about the function of the default gateway and how it's used by the Network Layer. Note that this section only introduces routers and routing and only presents

the information you need to see how the Network Layer uses routers. You will learn the details of routers and routing in a later section.

Let's start with learning what routers do in a network. The main concept to understand about routers is that they are used to connect different networks as opposed to a switch which is used to connect different devices on the same network. That is, if you only have one network segment, you can connect the devices on that segment using a hub or a switch. Routers on the other hand, are built specifically for passing network packets between networks.

For routers to provide this important function they need to process network data in a different way than other network devices. When a router receives network data, the network stack on the router acts much like any other network stack with one important difference. Normally, when the Network Layer receives data, it checks the destination IP address and discards any packets where the IP address doesn't match the device's IP address. But the destination IP address won't match the router's IP address for most of the packets the router processes. The destination IP address in the packets that are being routed between networks will be the IP address of the final destination device, not the router's IP address.

This is where the router's network stack will behave differently than other network devices. Rather than discarding network packet, the Network Layer on the router will assume that it's meant to forward the packet to the next router in the point-to-point delivery chain and pass the IP packet up to the routing application. The routing application code on the router will look at the destination IP address and use something called a routing table to decide which network or router the IP packet should be sent to next. The data will be passed back down the network stack and the router will build a new ethernet frame that will be used to send the data to the next device in the point-to-point delivery chain. The router places the original unchanged IP packet in the data section of the new frame and sends the frame to the next device in the delivery chain. The routing process, where each router builds a new ethernet frame to deliver the IP packet repeats in each router in the delivery chain until the IP packet reaches the final destination device.

The part of this process, where the IP packet remains unchanged while the ethernet frame is continuously changed is an important concept, so you should ensure that you understand why this happens.

Now let's look at how we tell devices on a network segment about the router they should use to send data to a different network. For this system to work each device connected to a network needs to know which routers can be used to reach different networks. Most networks, including home and office networks, have a single router, which is called the default router. Note that Microsoft decided to call this the default gateway, which is really a confusing mistake, since there are network devices called gateways that perform a completely different function. But, regardless of what it's called, the default router/gateway is the device that will take network packets from one network and deliver them to a different network and the network stack on each device must be configured with the IP address of the default router.

Note that each device must be configured the IP address assigned to the default router, not the router's DNS name or the router's MAC address. The information we really need to send ethernet frames to the router is the router's MAC address so you might think that it would be added to each device's network configuration, but the IP address is specified instead. Here's the explanation why. The reason we don't configure the MAC address of the default router/gateway, which is the piece of information we really want, is because using the IP address makes the system much more flexible. If we use MAC addresses, then any time the router, or the NIC in the router is changed, we'll have to change the network settings of every device on the network and add the router's new MAC address. But, if we use the IP address, we can swap out the router or the router's NIC without making any changes to the network settings on any of the network devices. Note that this flexibility is only possible because of ARP. That is, if we swap out the router or the router's NIC, all devices on the network can use ARP and the router's IP address to discover the new MAC. And since the ARP process happens automatically, we won't need to reconfigure the network settings on any other devices on the network.

The reason you can't use the DNS name is a short logic puzzle. If you could use the DNS name, then before sending any network packets to the router you would have to first send network packets to a DNS server to resolve the router's name to an IP address. But, sending the network packets for the DNS request requires sending those packets through the router itself, which you can't do if you don't know

the router's IP address. But if you already have the router's IP address, you'll be able to use it to send packets to the DNS server and avoid this Catch-22.

You should also note that most, but not all network administrators follow the convention of giving the router the host number 1 for any network. That is, if the network number is 143.33.41.0, the router will be assigned the IP address 143.33.41.1.

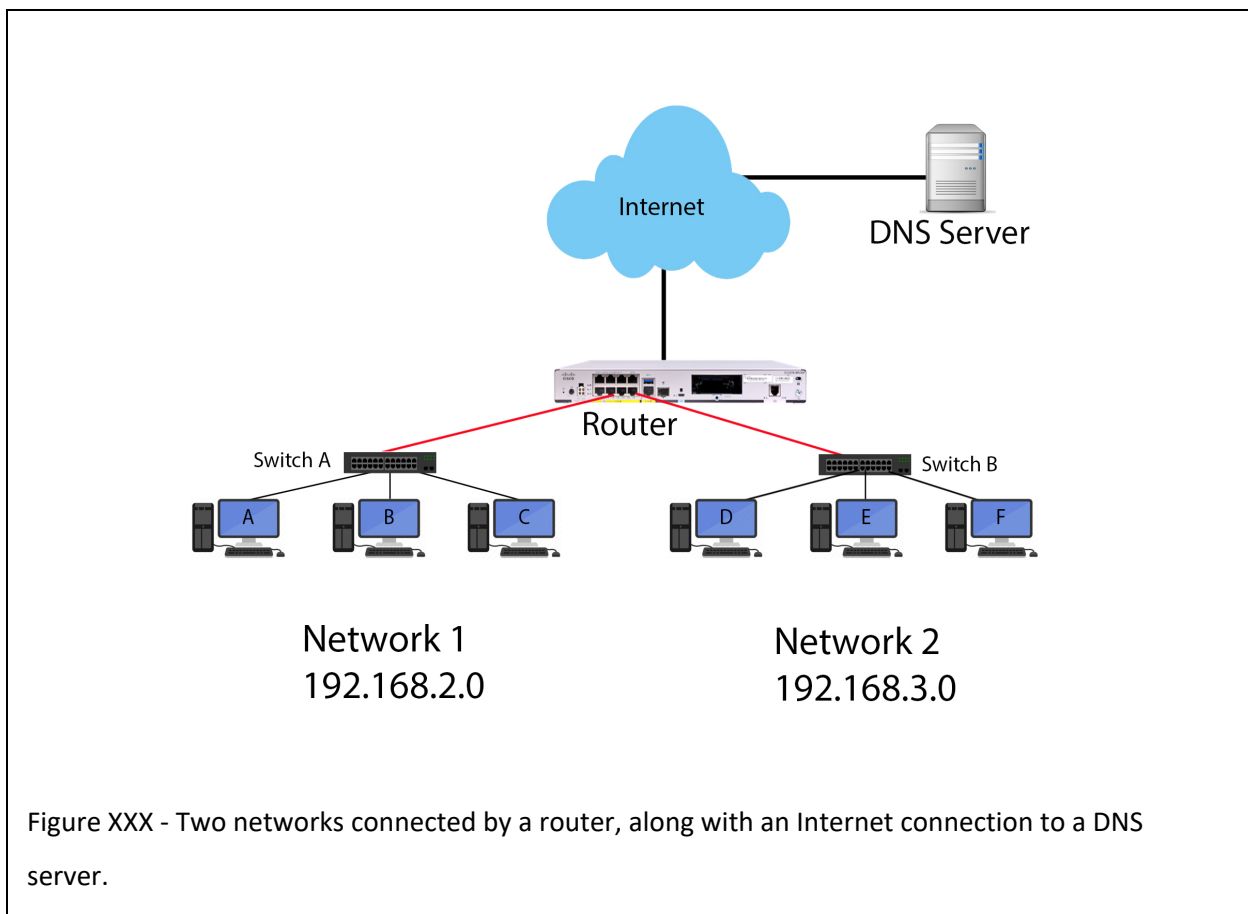
Now let's look at how the default router/gateway is used by the Network Layer. When any device on a network segment begins the process of sending network data the Network Layer uses the netmask to decide if the recipient device is on the same network or not. The Data Link Layer then sends the data, or ethernet frames to be technically correct, directly to the recipient if the destination device is on the same network, or it sends them to the default gateway/router if the destination device is on a different network.

This discussion about the default router/gateway might be a little confusing if your only exposure to networks has been your home network, because in a home network the router that connects your home network to the Internet typically also works as a switch, and maybe even as a modem. Instead of having a separate box for the home network switch and home network router, most wireless routers used in home networks act as a switch and a router. Your home router will act as a switch, connecting the devices in your home network so you can do things like access your home printer or media server, but it will also act as your network's router, moving network packets between the devices on your home network and the Internet.

You'll learn more details about routers later in the class, but for now make sure that you know the basic function of the default router/gateway, which is moving IP packets off your network to other networks and the Internet, and why any computer connected to a network must know the IP address of its default gateway/router.

Complete Delivery Demonstration

Here's a complete walkthrough of the steps involved in delivering network packets from end-to-end, as the packets go through a router, ARP is used to find MAC addresses, and DNS is used to resolve a hostname to an IP address. While each step is relatively straightforward, there are several steps involved and it can be helpful to see the entire process and how each step is performed. The demonstration will use the devices addresses shown in the following figure, concentrating on how network data is sent from Computer A to Computer F and back again.

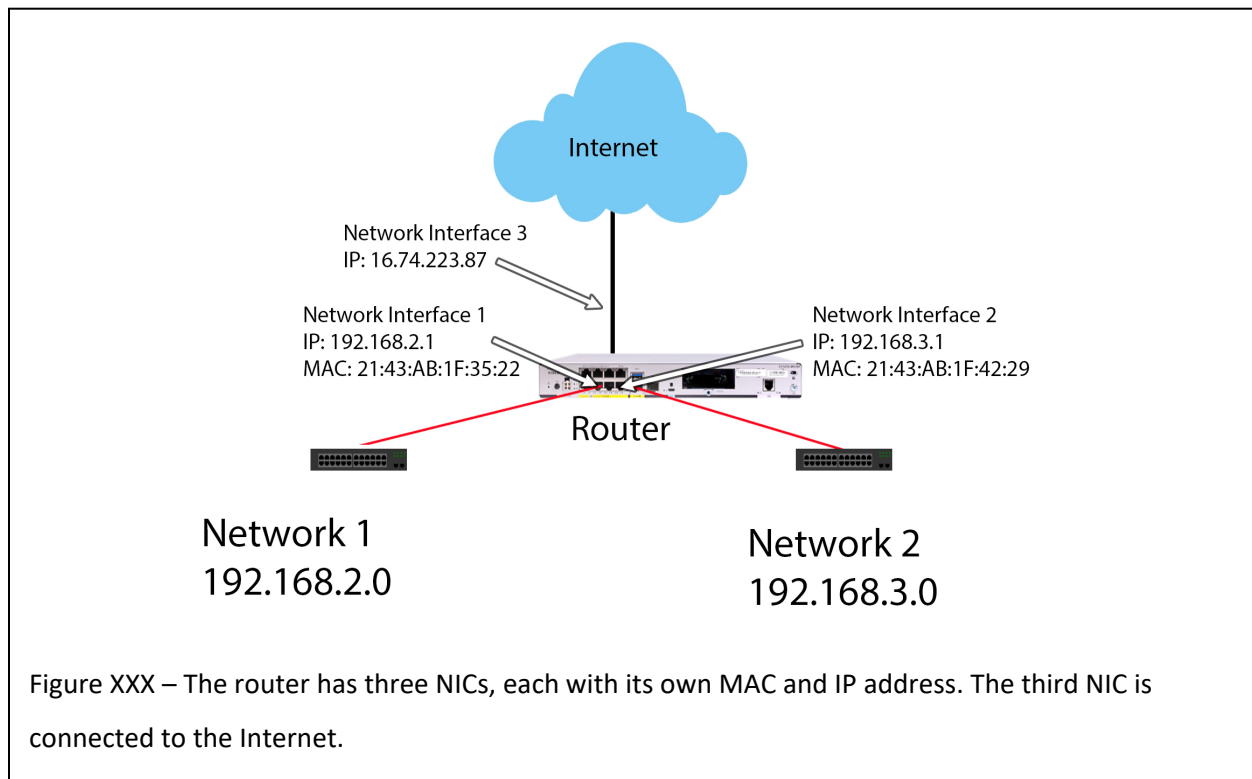


At the start of this process each computer will be configured to use TCP/IP, which means each computer will know the following information about its own network settings:

1. The MAC address on its network interface card (NIC)
2. Its IP address
3. The netmask
4. The IP address of the default router/gateway

5. The IP address of a Primary and Secondary DNS server

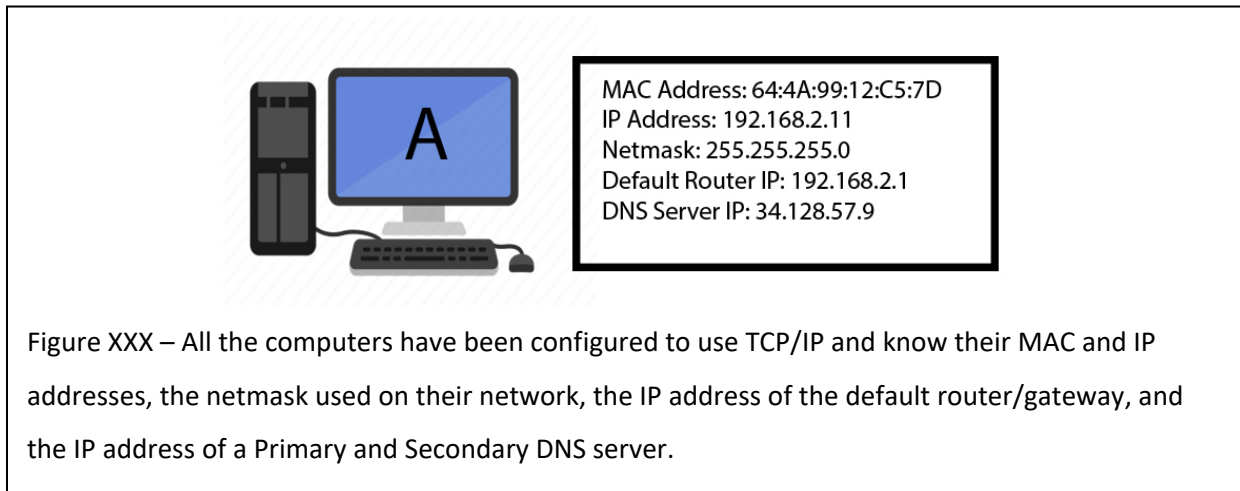
In addition to the computers, there's a router that connects the two networks. The Router has three NICs, one connected to Network 1, one connected to Network 2, and one connected to the Internet. The Router has been configured so the NIC connected to Network 1 uses an IP address of 192.168.2.1 and has a MAC address of 21:43:AB:1F:35:22, the NIC connected to Network 2 uses an IP address of 192.168.3.1 and a MAC address of 21:43:AB:1F:42:29, and the NIC connected to the Internet uses an IP address of 16.74.223.87 and a MAC address of 21:43:AB:1F:72:27.



Let's assume that Computer A wants to send a message to Computer F which has a DNS name of compF.Net2.org. To start, let's look at what the network stack on Computer A knows about its network settings:

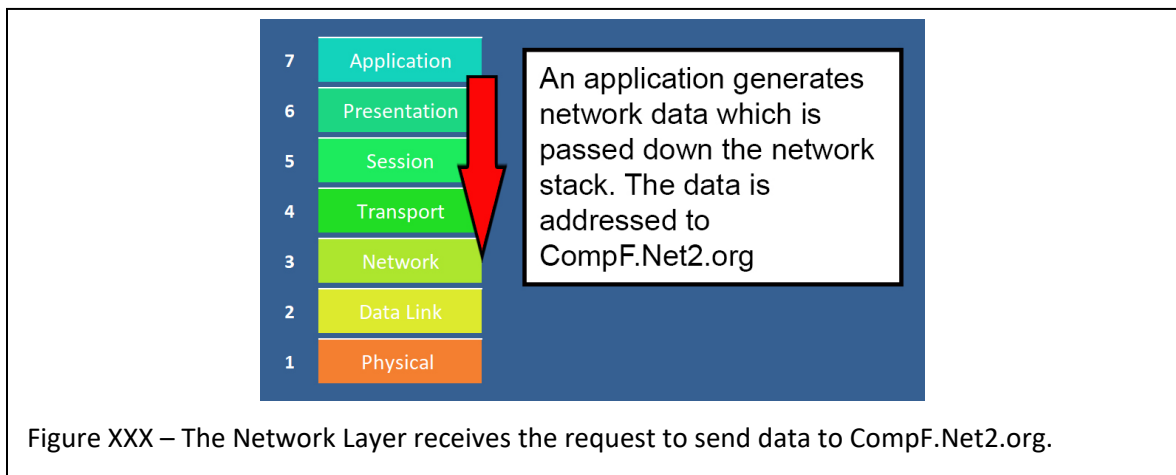
1. The MAC address on its network interface card (NIC) is 64:4A:99:12:C5:7D
2. Its IP address 192.168.2.11
3. The netmask is 255.255.255.0
4. The IP address of the default router/gateway 192.168.2.1

5. The IP address of a Primary DNS server 34.128.57.9

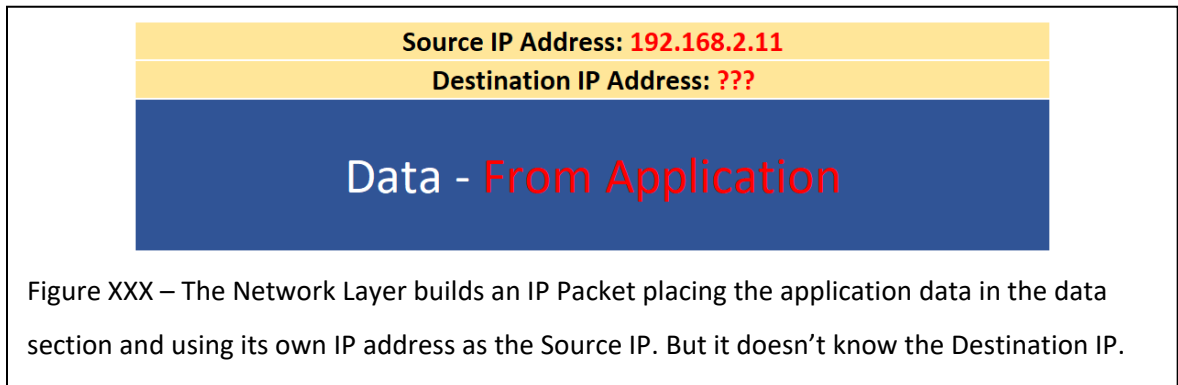


To deliver the data across the network Computer A is also going to need to know the MAC address of the default router/gateway, which it can get using ARP, and the IP address of Computer F which it can get using DNS. Here are the steps that the network stack on Computer A uses to discover the unknown information, and to make this network transmission:

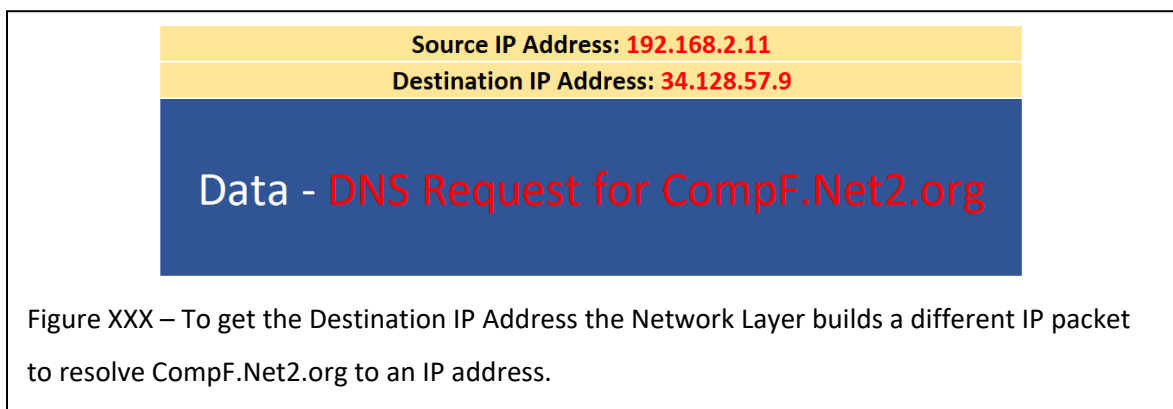
1. The process starts with an application on Computer A asking the network stack to send some data to Computer F. This data is passed down the network stack to the Network Layer.



2. The Network Layer of the stack receives this request and starts to build an IP packet. The data is placed in the data portion of the IP packet and the Source IP in the header is set to Computer A's IP address which the network stack knows is 192.168.2.11. When the Network Layer tries to write the Destination IP address for Computer F it runs into a problem as the IP address is not known. The Network Layer only knows Computer F's DNS name which is CompF.Net2.org. However, the Network Layer knows it can use DNS to resolve the name CompF.Net2.org to an IP address.



3. To get Computer F's IP address, the Network Layer on Computer A has to put the original network message on hold while it builds an IP packet with the DNS request for Computer F's IP address. The IP packet containing the DNS request uses the Source IP Address of 192.168.2.11, which is Computer A's IP address, and the Destination IP Address of the Primary DNS server which is 34.128.57.9. The data portion of this IP packet holds the DNS request for CompF.Net2.org.



4. Once the IP packet with the DNS request is built, the Network Layer must decide whether to ask the Data Link Layer to send it directly to the DNS Server, or if it needs to be sent to the default router. That is, the Network Layer must decide whether the DNS Server is on the same network segment or on a different network segment.
 - a. The network stack on Computer A makes this decision by using the netmask of 255.255.255.0 to compare Computer A's IP Address which is 192.168.2.11 and the DNS Server's IP address which is 34.128.57.9. This comparison shows that the computers are on different networks, which means the ethernet frame containing the IP packet should be sent to the default router.

255	.	255	.	255	.	0
Network						Host
192	.	168	.	2	.	11
34	.	128	.	57	.	9

Figure XXX – The netmask is used to determine if Computer A and the DNS server are on the same network or not.

- b. Because Computer A and the DNS Server are on different networks, the ethernet frame containing the IP packet will be sent to the default router. The Data Link Layer on Computer A builds an ethernet frame using its own MAC address, 64:4A:99:12:C5:7D, as the source MAC.

Destination MAC Address	Source MAC Address	Data
???	64:4A:99:12:C5:7D	IP Packet with DNS Request

Figure XXX – The Source MAC and the IP Packet with the DNS Request are added to the ethernet frame.

- c. The Data Link Layer on Computer A does not know the MAC of the default router, so it puts the ethernet frame with the DNS request on hold while it builds an ARP packet to discover the default router's MAC address. The ARP process requires the following steps:

- i. Computer A builds the ARP Request using its own IP address of 192.168.2.11 as the Source IP and its own MAC address as the Source MAC. Computer A also knows the IP address of the default router, since this is one of the items that must be configured, so it sets the Destination IP address to 192.168.2.1. The piece of information Computer A does NOT know, and the thing we're trying to find, is the MAC address of the default router. Since Computer A doesn't know this it uses a MAC broadcast of FF:FF:FF:FF:FF:FF as the destination MAC.
- ii. Computer A's NIC places the ARP packet on the network where it is seen by all devices connected to the Network 1, including NIC 1 on the default router.
- iii. The Data Link Layer on all devices on Network 1 look at the destination MAC address, and since it's a broadcast they all pass it up their network stacks to the Network Layer. The Network Layer on all devices check the destination IP address, and since it's addressed to 192.168.2.1, which is the IP address of the default router, all the Network Layers except the default router's discard the ARP packet. Since this packet is addressed to the default router, it unpacks the data with the ARP request and sends it up the network stack.
- iv. The Network Layer on the default router reads the ARP request and knows to build an ARP response. To send the ARP response back to Computer A, it uses Computer A's IP address as the destination IP and Computer A's MAC address as the destination MAC. It knows both addresses as it received them in the ARP request. The default router sets the source IP in the ARP response to its own IP address, and the source MAC in the ARP response to its own MAC.
- v. NIC 1 on the default router places the ethernet frame containing the ARP response on Network 1.
- vi. The ethernet frame is seen by Computer A's Data Link Layer, and since it is addressed to Computer A's MAC the data is unpacked and sent up the network stack. The Network Layer on Computer A reads the IP Packet header and sees

that the destination IP address matches its IP address, so it reads the IP Packet data, extracting the MAC address of the default router.

Computer A now knows the following:

1. The MAC address on its network interface card (NIC) is 64:4A:99:12:C5:7D
2. Its IP address 192.168.2.11
3. The netmask is 255.255.255.0
4. The IP address of the default router/gateway 192.168.2.1
5. The IP address of a Primary DNS server 34.128.57.9
6. The MAC address of the NIC connected to Network 1 on the Router
21:43:AB:1F:35:22

- d. Computer A's network stack now has the MAC address of the default router, so its Data Link Layer can finish building the ethernet frame holding the DNS Request by adding the default router's MAC address as the Destination MAC. Remember that the data portion of this ethernet frame contains the IP packet with the DNS Request for CompF.Net2.org.
- e. The Physical Layer on Computer A places the ethernet frame on the network media, where it is seen by NIC 1 on the default router.
- f. The Physical Layer on the default router reads the ethernet frame, and hands it to the Data Link Layer. Since the destination MAC address matches the default router's own MAC address it unpacks the data from the ethernet frame and passes it up the network stack to the Network Layer.
- g. The Network Layer on the default router sees that the Destination IP address is not its own IP address, so it builds another ethernet frame to send the IP Packet with DNS request to the next router in the chain of routers on the path to the DNS server. NIC 2 on the router transmits this ethernet frame, sending the network data to the next router in the transmission chain.

- h. The DNS request is passed from router to router, or from point to point, with each router building a new ethernet frame to hold the IP packet with the DNS Request, until the IP packet finally reaches the DNS server.
- i. The Data Link Layer on the DNS server reads the ethernet frame, and since the Destination MAC address matches its MAC address it unpacks the data from the ethernet frame and passes it up the network stack to the Network Layer.
- j. The Network Layer on the DNS server sees that the Destination IP address matches its IP address, so it unpacks the data in the IP packet and sends it up the network stack to the DNS server application.
- k. The DNS server reads the DNS Request and builds a DNS Response containing the IP address for CompF.Net2.org. The DNS server sends the Response back down the network stack, asking that it be sent to the IP address for Computer A.
- l. The Network Layer on the DNS server builds an IP packet placing the DNS response in the data section, setting the Source IP to the DNS server's IP address, and setting the Destination IP address to Computer A's IP address. It knows Computer A's IP address because it was contained in the IP packet with the DNS request.
- m. The Network Layer on the DNS server compares the Source and Destination IP addresses using the netmask of 255.255.255.0 and determines that they are on different networks. Since they're on different networks the DNS server's Data Link Layer builds an ethernet frame to send to its default router. It places the IP packet containing the DNS response inside the ethernet frame, sets the Source MAC to its own MAC, and sets the Destination MAC to the MAC of the default router for its network, which it knows as it was contained in the ethernet frame with the DNS Request.
- n. The DNS server sends the ethernet frame to its default router. This router looks at the frame, sees that the destination MAC matches its MAC, so it passes the data up its network stack. Since the Destination IP address doesn't match its own, it checks its

routing table for the next router in the chain and passes the data back down to the Data Link Layer for shipment to the next router. The Data Link Layer places the IP packet with the DNS Response in a new ethernet frame which is sent to the next router in the chain. This process is repeated as the IP packet containing the DNS Response is passed from point to point in the delivery chain, finally making it to Computer A.

- o. The Data Link Layer on Computer A reads the ethernet frame, and since the Destination MAC address matches its MAC address it unpacks the data from the ethernet frame and passes it up the network stack to the Network Layer.
- p. The Network Layer on Computer A sees that the Destination IP address matches its IP address, so it unpacks the IP data and sees that it contains the DNS Response with the IP address for CompF.Net2.org.

Computer A now knows the following and has everything it needs to build the IP packet to send to Computer F:

- 1. The MAC address on its network interface card (NIC) is 64:4A:99:12:C5:7D
 - 2. Its IP address 192.168.2.11
 - 3. The netmask is 255.255.255.0
 - 4. The IP address of the default router/gateway 192.168.2.1
 - 5. The IP address of a Primary DNS server 34.128.57.9
 - 6. The MAC address of the NIC connected to Network 1 on the Router
21:43:AB:1F:35:22
 - 7. The IP address of Computer F 192.168.3.31
-
- 5. The Network Layer on Computer A now has the IP address of COMpF.Net2.org, 192.168.3.31, and can return to building the IP packet to send the data to Computer F. The Network Layer writes 192.168.3.31 in the Destination IP address field of the IP packet.
 - 6. Computer A's Network Layer determines whether to ask its Data Link Layer to send the ethernet frame directly to Computer F, or if the ethernet frame should go to the default router. It does

this by using the netmask to find the network portion of 192.168.2.11 and 192.168.3.31. Since the netmask is 255.255.255.0 it uses the first three octets and compares 192.168.2 with 192.168.3. Since the network numbers are different, it asks the Data Link Layer to send the ethernet frame to the default router.

7. The Data Link Layer on Computer A builds an ethernet frame to send to the default router. It uses its own MAC address of 64:4A:99:12:C5:7D as the source MAC. At this point Computer A probably has the MAC address of the default router in its ARP cache, and it can set the destination MAC to the cached MAC address in the ethernet frame. If Computer A doesn't know the default router's MAC address it uses ARP to find it using the process detailed in step 2B. The Data Link Layer on Computer A places the IP packet in the data section of the ethernet frame.
8. The Data Link Layer on Computer A hands the ethernet frame to the Physical Layer for transmission on the network media for Network 1.
9. The Physical Layer on the Router's 1st NIC sees the network transmission and hands it up to its Data Link Layer.
10. The Data Link Layer on the Router checks the ethernet frame and sees that the destination MAC address matches its MAC, so it unpacks the IP packet data and hands it up the network stack to the Router's Network Layer.
11. The Network Layer on the Router checks the destination IP address and sees that it doesn't match its own IP address. The router knows that this means it is supposed to route the packet to another network. Using its routing tables, the Router determines that the packet should be sent out across its other network interface, the NIC connected to Network 2. The Router then builds another ethernet frame, adding the IP packet to the data section of the ethernet frame. The Router sets the source MAC as the MAC address of its second NIC, which is 21:43:AB:1F:42:29. If the Router doesn't know Computer F's MAC address it uses ARP to find it. Once the Router has Computer F's MAC address, 64:4A:99:12:90:01, it puts it in the ethernet frame as the destination MAC address.

12. The Router hands the ethernet frame to the Physical Layer for its 2nd NIC, which places the transmission on Network 2.
13. Computer F's NIC sees the transmission and hands the ethernet frame to its Data Link Layer which checks the destination MAC address and sees that it matches its own MAC address. Since it's a match, the Data Link Layer unpacks the IP packet from the ethernet frame and hands it up to the Network Layer.
14. The Network Layer on Computer F checks the destination IP address and sees that it matches its IP address, so it knows it should further process the network transmission. It unpacks the data from the IP packet and passes it up the network stack to the appropriate program.
15. If the application on Computer F needs to reply to Computer A, it builds the reply and passes it back down the network stack to the Network Layer.
16. The Network Layer on Computer F builds an IP packet using its own IP address as the source IP and Computer A's IP address as the destination IP. It knows Computer A's IP address because it was in the IP packet Computer F received from Computer A. The reply passed down the network stack from the application is placed in the data portion of the IP packet.
17. Computer F's Network Layer determines whether to ask its Data Link Layer to send the ethernet frame directly to Computer A, or if the ethernet frame should go to the Router. It does this by using the netmask to find the network portion of 192.168.3.31 and 192.168.2.11. Since the netmask is 255.255.255.0 it uses the first three octets and compares 192.168.3 with 192.168.2. Since the network numbers are different, it asks the Data Link Layer to send the ethernet frame to the Router.
18. The Data Link Layer on Computer F builds an ethernet frame to send to the Router. It uses its own MAC address of 64:4A:99:12:90:01 as the source MAC. At this point Computer F probably has the MAC address of the default router's 2nd NIC in its ARP cache, and it can set the destination MAC to the cached MAC address in the ethernet frame. If Computer F doesn't know the MAC address of the Router's 2nd NIC it uses ARP to find it using the process detailed in step

2B. The Data Link Layer on Computer F places the IP packet in the data section of the ethernet frame.

19. The Data Link Layer on Computer F hands the ethernet frame to the Physical Layer for transmission on the network media for the second network segment.
20. The Physical Layer on the Router's 2nd NIC sees the network transmission and hands it up to its Data Link Layer.
21. The Data Link Layer on the Router checks the ethernet frame and sees that the destination MAC address matches its destination MAC. Since the ethernet frame is addressed to the Router, it unpacks the IP packet data and hands it to the Router's Network Layer.
22. The Network Layer on the Router checks the destination IP address and sees that it doesn't match its own IP address. The router knows that this means it is supposed to route the packet to another network. Using its routing tables, the Router determines that the packet should be sent out across its first NIC. The Router then builds another ethernet frame, adding the IP packet to the data section of the ethernet frame. The Router sets the source MAC as the MAC address of its first NIC and the destination MAC to the MAC address of Computer A which it should have in its ARP cache.
23. The Router hands the ethernet frame to the Physical Layer for its first NIC, which places the transmission on the network media for the first network segment.
24. Computer A's NIC see's the transmission and hands the ethernet frame to its Data Link Layer which checks the destination MAC address and sees that it matches its own MAC address. Since it's a match, the Data Link Layer unpacks the IP packet from the ethernet frame and hands it up to the Network Layer.
25. The Network Layer on Computer A checks the destination IP address and sees that it matches its IP address, so it knows it should further process the network transmission. It unpacks the data from the IP packet and passes it up the network stack to the appropriate program.

This demonstration shows how everything you've learned up to this point is used to move network data from end-to-end, by making more than one point-to-point transmission, and how this is made possible using the IP addresses, netmask, MAC addresses, DNS server IP address, default router/gateway IP address, and ARP. While the process might seem complicated because of the number of steps, each step is relatively straightforward. If you find the entire process confusing, ensure that you understand what is being accomplished in each individual step and why this happens.

Viewing and Configuring Network Settings on a Windows Computer

Now that you've learned the basics about IP addresses, netmasks, the default gateway/router, and DNS, and how they work together at the Network Layer, let's look at configuring and checking these network settings on a Windows based computer. First, you'll learn two ways to check the network settings on a Windows based computer, and then you'll learn the two or three ways to configure the network settings. The two ways to configure the network settings are to do the configuration manually or to use a network service called Dynamic Hardware Configuration Protocol (DHCP) to automatically do the configuration. There's also a third way to configure some, but not all, of the settings using something called Automatic Private IP Addressing (APIPA).

To start, let's look at the two ways to view the current network settings. The first is to open a command window and use the `ipconfig` command. To do this, first open a Command Window using these steps:

1. Go to Windows Search and type **cmd**, then select **Command Prompt**. This opens the Command Prompt application.

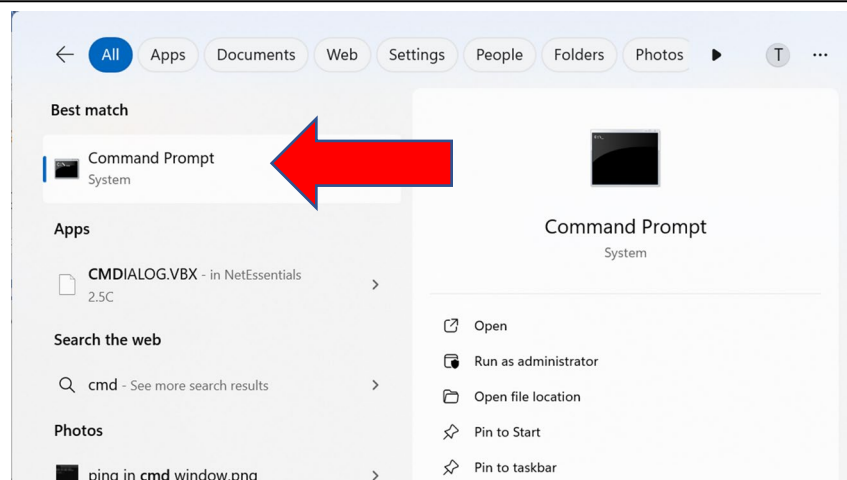


Figure XXX – Starting the Command Prompt application.

2. Place the cursor in the Command Prompt window and type: **ipconfig /all | more** . (The pipe character “|” is usually found just above the <enter> key on the keyboard.) This will display the network settings, one page at a time.

```

Command Prompt
Physical Address. . . . . : A2-02-A5-C6-C8-93
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : attlocal.net
Description . . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
Physical Address. . . . . : A0-02-A5-C6-C8-93
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : 2600:1700:38d1:5290::41(Preferred)
Lease Obtained. . . . . : Wednesday, April 16, 2025 6:24:09 AM
Lease Expires . . . . . : Wednesday, April 16, 2025 8:54:08 AM
Link-local IPv6 Address . . . . : fe80::f382:7927:c284:153f%6(Preferred)
IPv4 Address. . . . . : 192.168.1.66(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Friday, April 11, 2025 3:22:08 PM
Lease Expires . . . . . : Wednesday, April 16, 2025 9:33:16 PM
Default Gateway . . . . . : 192.168.1.254
DHCP Server . . . . . : 192.168.1.254
DHCPv6 IAID . . . . . : 127926949
DHCPv6 Client DUID. . . . . : 00-01-00-01-2E-2E-80-90-A0-02-A5-C6-C8-93
DNS Servers . . . . . : 2600:1700:38d1:5290::1
                       : 192.168.1.254
NetBIOS over Tcpip. . . . . : Enabled
Connection-specific DNS Suffix Search List :
                                           attlocal.net

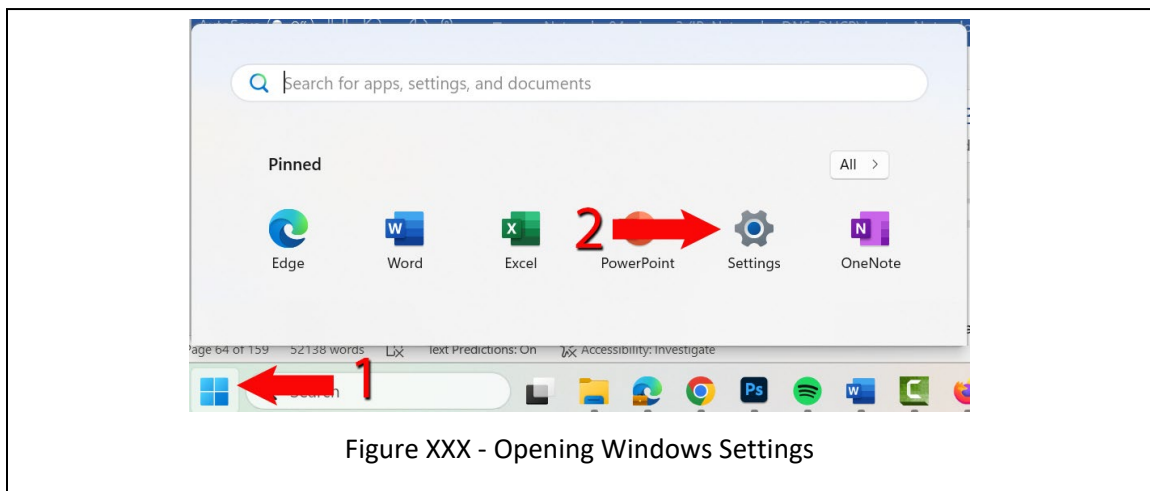
```

Figure XXX – Running the **ipconfig /all | more** command.

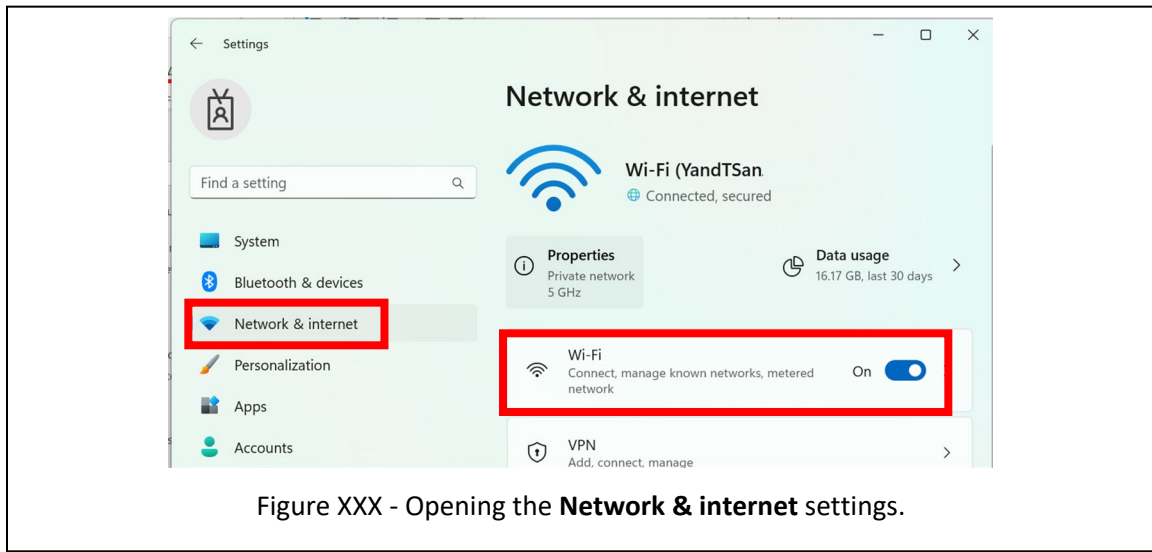
3. Use the <enter> key to scroll through the file until you find settings for your currently active network adapter. If you are connected to the network via a wired connection this will probably be the Ethernet adapter section. If you're connected to a wireless network this will be the Wireless LAN adapter Wi-Fi section.

The second method for viewing the network configuration is to use the Windows Settings using this process:

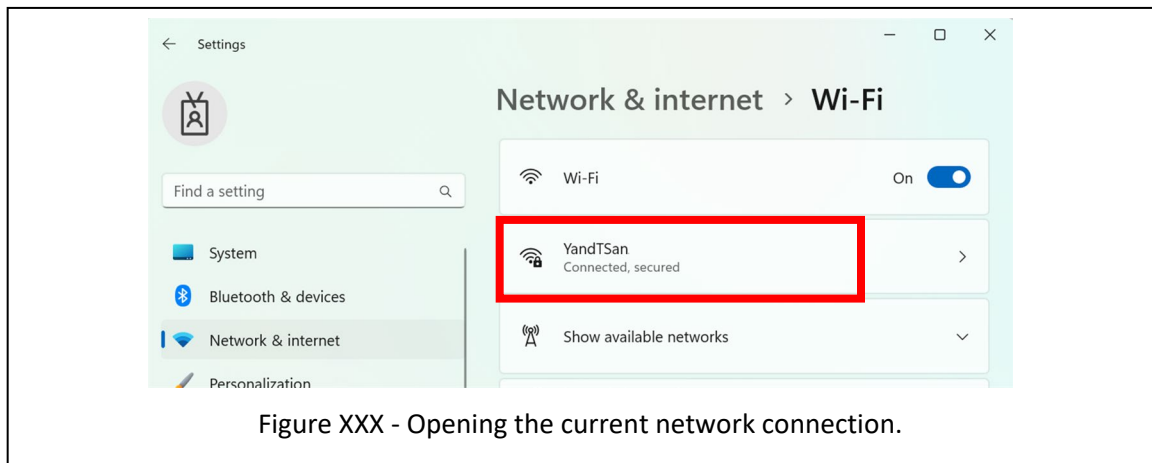
1. Open Windows Settings by going to the **Windows Start button** , then select **Settings** .



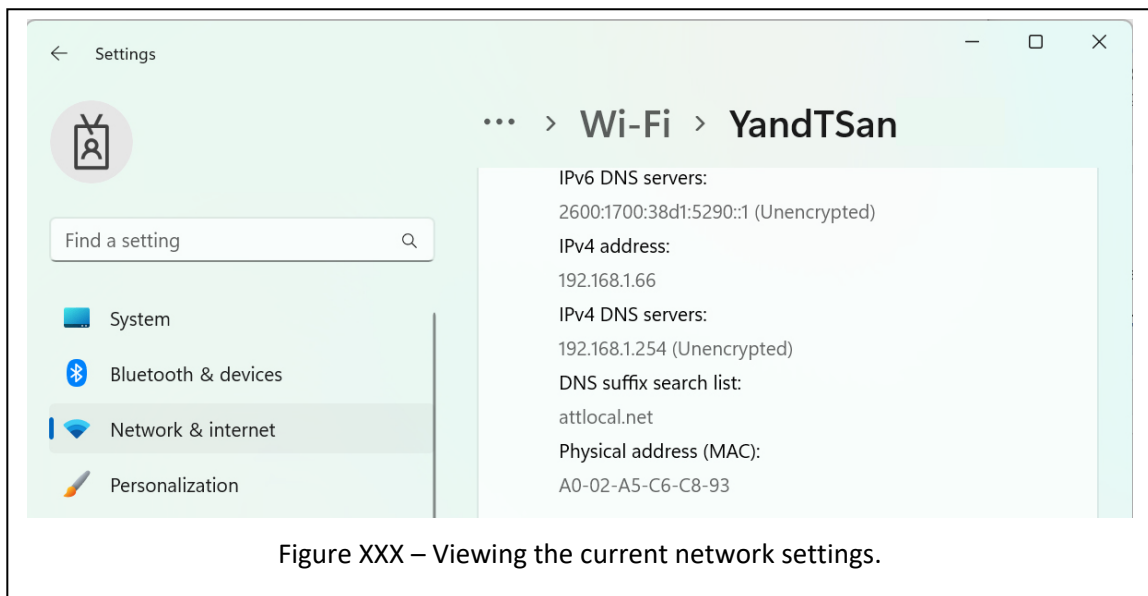
2. Click on **Network & internet**, then click on **Wi-Fi** if you're connected to a wireless network or **Ethernet** if you're connected to a wired network.



3. Open the network you're connected to by clicking the box with the network name.



4. Scroll down to view the network settings.



Manual Configuration

Manually configuring the network settings on a computer or phone requires knowing a couple of things. The first is the network settings themselves. You'll need to know the IP address, the netmask, the IP address of the default gateway, and the IP address for the DNS resolver(s), of which there are typically two, a primary and a secondary which will act as the backup in case the primary is not available.

The second thing you'll need to know is where to go to get the values for the network settings. Remember you can't just make these up, you'll need to get them from someone who got them from IANA. If you're configuring the settings manually, then you're probably going to be working as a network administrator, where someone above you in the network chain of command will provide you with the information you need. This might be a head network administrator for an organization, or if you're the head net admin you'll get your IP addresses from someone at your network's ISP.

If you're looking at your home network and want to manually configure the network settings you should first note that this is typically done using DHCP, which automates and simplifies the process. If you still want to do the configuration manually, you'll need to login to your router and get the IP address of the router and the IP address of the DNS resolver(s). The set of IP addresses you assign to each device on

the network will typically come from the non-routable network 192.168.0.0., and the netmask will typically be 255.255.255.0. (You'll learn about the non-routable IP addresses later in this section.)

Once you have the network settings, the last thing you'll need to know is how to configure them on your device. You'll learn how to do this for Windows based computers below, but the process will be different in every OS and on every device. If you need to configure the settings on a different OS or device, you can undoubtedly find help on the Internet.

DHCP

Dynamic Host Configuration Protocol (DHCP) is a network protocol and process that automatically assigns IP addresses, netmasks, and the IP addresses of the default gateway and DNS servers to devices on a network. Using DHCP to automatically configure the network settings makes it much easier to manage a large network, and it also makes it easier for non-technical users, so it's used on most home networks.

DHCP works by having a DHCP server on the network that manages a pool of available IP addresses. When a new device connects to the network, it sends a broadcast message looking for a DHCP server and asking for an IP address. The DHCP server receives this message and responds with an offer of an available IP address. The device then accepts the offer and the DHCP server assigns the IP address to the device for a set amount of time, known as the lease time.

DHCP is particularly useful in large networks because it reduces the amount of manual configuration that needs to be done. It's not difficult to configure the network settings on any single device, but on a large network with hundreds or thousands of devices, configuring and managing the network settings on all the devices would take a lot of time.

DHCP also makes connecting to home networks or networks outside the home much easier. Think about your home network and the devices you connect. Typically, the only thing you, or anyone that connects to your home network, needs to do is connect to your wireless router. You don't need to set or change the IP address, netmask, default gateway, and DNS server. This is because your computer is set to DHCP, and when you connect the computer to your home network your router acts as a DHCP server and gives

your computer the IP address to use, along with the correct netmask, default gateway IP address, and IP addresses for the primary and secondary DNS servers. The same thing happens when you travel with your computer and connect to a network at a business, motel, or airport. That is, you don't have to configure any of the network settings, instead your computer sends out a DHCP broadcast when you initially connect to the network and gets the network configuration settings from a DHCP server.

Another benefit of DHCP is that it allows for efficient use of available IP addresses, as devices are only assigned an IP address when they need it, and the address can be released when it is no longer needed.

You'll learn the details about DHCP later in the class.

APIPA

Automatic Private IP Addressing (APIPA) is a feature in Microsoft Windows operating systems that provides a fallback mechanism for devices when they cannot obtain an IP address from a DHCP server.

APIPA is designed to allow devices on a local network to communicate with each other even when there's no DHCP server available. When a device is unable to obtain an IP address from a DHCP server, if it's using APIPA it automatically assigns itself an IP address in the range of 169.254.0.1 to 169.254.255.254. This address is known as an APIPA address. Devices with APIPA addresses can communicate with each other on the local network, but they can't communicate with devices on other networks or access the Internet. APIPA does not provide a default gateway or DNS server, so devices with APIPA addresses cannot access resources outside of their local network.

APIPA isn't meant to be a replacement for DHCP, it's just meant to be used to connect to a network "in case of emergency", which means in situations where a DHCP server is not available. APIPA provides enough to get a computer connected to the local network segment, but even then, the computer may not be able to communicate with other network devices such as file servers or printers if the other devices don't have IP addresses that use the same network number. And this will probably be the case if your network has a DHCP server that unexpectedly becomes unavailable. That is, devices that connected to the network using settings from the DHCP server will use one set of network numbers, while devices

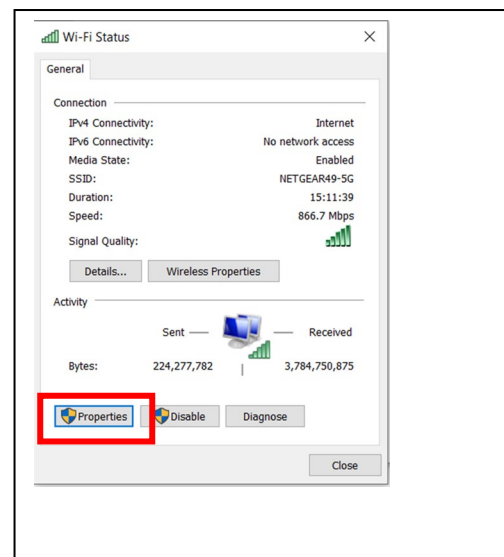
that try to connect to the network after the DHCP server goes down will use a different set of network numbers.

In general, APIPA should only be used as a last resort, in emergencies when a DHCP server is not available, and it should not be relied upon as a long-term solution for network addressing.

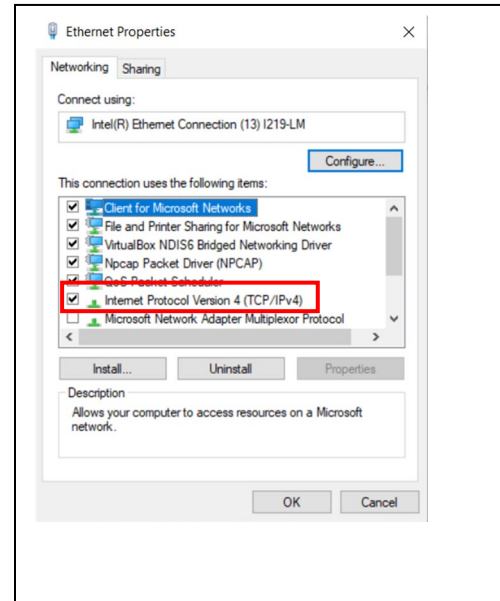
Manual Configuration and DHCP Configuration in Windows

Here's the process for manually configuring the network settings or selecting DHCP on a computer running Microsoft Windows:

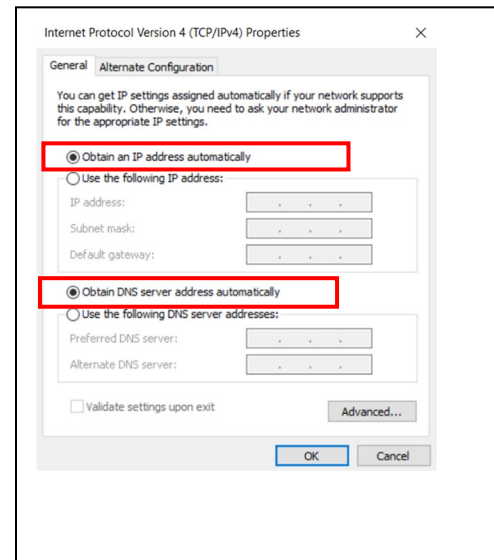
1. Open the **Control Panel** from the **Start** menu.
2. Click on **Network and Sharing Center**. Note that there are several ways to get the Network and Sharing Center, but these steps are the quickest.
3. In the left-hand menu, click on **Change adapter settings**.
4. If you're checking a wired network connection double-click the network adapter you want to configure, or right-click on the network adapter and select **Properties**. If you're checking a wireless connection, double-click the wireless adapter, then select the **Properties** button at the bottom of the dialog box.



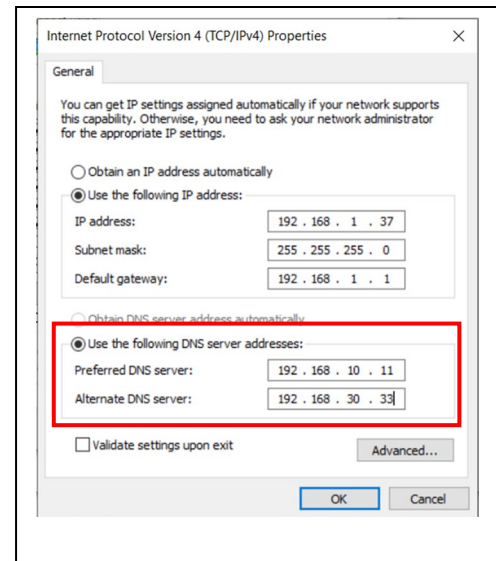
5. Find Internet **Protocol Version 4 (TCP/IPv4)** in the list of items in the middle of the dialog box, then select it and click **Properties** or double-click it.



6. In the new dialog box, ensure that the **General** tab is selected. This is where you will set the IP address, Subnet mask, Default gateway, and DNS server values. You can see that there are two sections in the dialog box for entering these values, one for the IP address information and one for the DNS server addresses. If these are set to **Obtain an IP address automatically** and **Obtain DNS server address automatically** it means the computer is set to use DHCP. Note that if you select DHCP, you won't see any values for the IP address settings of the DNS servers. If the computer uses DHCP and it's connected to the network, it will have appropriate values for these settings, but they will remain blank in this dialog box.



7. To manually set the network values, select **Use the following IP address** and **Use the following DNS server addresses**. Note that for DNS, Microsoft uses the terms **Preferred DNS server** and **Alternate DNS server** for what we called DNS Resolvers in the explanation of the overall DNS system.
8. Once the network settings are configured, click **OK** to exit the Properties window.



Basic Network Troubleshooting

Next, let's look at troubleshooting possible problems with the IP configuration. As you've seen there are a lot of settings that could be wrong, and several components such as DHCP, the default gateway, or DNS that could be having issues, so having a good troubleshooting process is important. In this section you'll learn about a few commands and tools used for troubleshooting, and a special IP address called the localhost or loopback address that's also used for troubleshooting the network stack. We'll start by going through a troubleshooting process that checks every possible problem, and then finish by looking at what steps of the process would be used to check specific problems.

In our general troubleshooting process, we'll do a set of checks that starts with checking things on the local computer, then work our way out to check network components like DNS or the default gateway. Hopefully you can see the sense in checking to make sure the IP settings are correct on a computer before we start checking DNS servers. It's kind of like if we were having trouble sending snail mail. It would be smarter to make sure we put a stamp on the envelope first, before we start checking for problems with mail sorting machines in Spokane and Seattle.

Here are the steps I suggest you use for troubleshooting network connections:

1. **Check the physical connections.** Ensure that all cables are securely connected and that the network interface is enabled. Look for LED lights on the network interface and if you have access to it, check for lights on the switch or router to verify that end of the cable is also connected and working.
2. **Check the IP settings.** Verify that the computer has a valid IP address, subnet mask, default gateway, and DNS server address. This is done by using the `ipconfig /all` command on Windows.

```
Ethernet adapter Ethernet:
    Connection-specific DNS Suffix . :
    Description . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
    Physical Address. . . . . : 84-5C-F3-CA-82-B7
    DHCP Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . . : Yes
    Link-local IPv6 Address . . . . : fe80::f631:51fb:eb16:9e62%4(Preferred)
    IPv4 Address. . . . . : 192.168.1.6(Preferred)
    Subnet Mask . . . . . : 255.255.255.0
    Lease Obtained. . . . . : Wednesday, May 3, 2023 3:56:44 PM
    Lease Expires . . . . . : Friday, May 5, 2023 6:32:53 AM
    Default Gateway . . . . . : 192.168.1.1
    DHCP Server . . . . . : 192.168.1.1
    DHCPv6 IAID . . . . . : 59006195
    DHCPv6 Client DUID. . . . . : 00-01-00-01-28-2E-19-CC-34-73-5A-DE-7A-3B
    DNS Servers . . . . . : 192.168.1.1
```

3. **Use ping to check the network stack.** This test will check to see if the network stack is functioning by asking the computer to talk to itself through the computer's network interface. If this test fails, it's a sign that some part of the computer's network software is not configured right or not working as expected. If the test is successful, it means that the network stack is functioning correctly.

The ping tool is a network utility used to test the connectivity and reachability of a host on an Internet Protocol (IP) network. When you start ping, you give it the name or IP address, and then ping sends a series of small packets of data, called Internet Control Message Protocol

(ICMP) echo requests, to the target host. Ping waits for a response for each request, which should be an ICMP echo reply. The ICMP echo requests don't have any data, they just used to "knock on the door" of network devices and see if anyone answers or not.

When you run the ping command, the program creates a series of ICMP packets and sends them to the target IP address. The ping tool measures the round-trip time for each packet to travel from the source host to the target host and back again. It also displays information about the number of packets sent, received, and lost, as well as the minimum, maximum, and average round-trip time (RTT) for each packet.

The ping tool is commonly used for testing and troubleshooting network connectivity issues, such as identifying network latency, packet loss, or connectivity problems, but in this case, we're having a computer ping itself, just to test to see if the network stack is operating. This is accomplished by telling ping to target a special IP address, 127.0.0.1, which all network software use to refer to the computer the software is running on. We also use the names localhost or loopback to refer to 127.0.0.1.

When packets are sent to the loopback address, they are processed by the network stack in the same way as packets that are sent to any other IP address. The network stack is the set of protocols and services that are responsible for sending and receiving network data on a computer.

When an application like ping sends data to the loopback address, 127.0.0.1, the data is first processed by the upper layers of the network stack, then passed down to the network layer, where it is encapsulated in an IP packet and the source and destination IP addresses are set to 127.0.0.1. The IP packet is then passed down to the data link layer, where it is encapsulated in an ethernet frame where the source and destination MAC addresses are set to the computer's MAC address. The ethernet frame is then sent out on the loopback interface, which is a virtual network interface that represents the loopback address. The packet is not actually sent out on any physical network but is instead looped back to the computer's own network stack.

When the packet is received by the loopback interface, it is processed by the network stack in the reverse order that it was sent. The data link layer unpacks the ethernet frame and hands the IP packet to the network layer. The Network layer unpacks the IP packet and passes the data back up to the upper layers in the network stack.

Sending packets to 127.0.0.1 or the loopback interface allows applications on a computer to communicate with itself, testing the network stack as if the packets were being sent and received from the network.

```
C:\WINDOWS\system32>ping 127.0.0.1
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\WINDOWS\system32>ping 127.0.0.1
Pinging 127.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4
    (100% loss),
```

Pinging 127.0.0.1. A successful ping is shown on the left, and an unsuccessful ping is shown on the right.

4. **Check local network connections and default gateway.** If the computer can ping itself, you can be assured that the network stack is functioning. The next thing to do is to try and connect to other devices on the same network segment. The network connection we typically test next is the one to the default gateway. Testing this connection makes sense since we'll need the default gateway in later tests, so we might as well test it now.

The first step in performing this test is to find the IP address of the computer's default gateway, which can be found using the `ipconfig /all` command, which you did in step 2. Next, run `ping` using the default gateway's IP address as the target for ping. For example, if the default gateway's IP address is 192.168.1.1 you would run the command:

```
ping 192.168.1.1
```

As in the previous step, if this is successful the default gateway will send back replies, but if it isn't successful ping will display a request timed out message. If this test succeeds it means your computer can make network connections with other devices on the local network segment and reach the default gateway. If this test fails, it means that there's an issue making connections to the local network segment, or there are problems with the default gateway. To narrow this down you can:

- A. Check other devices on the local network segment and see if you can ping those devices from your computer. If you can ping another device, it's a good sign that you either have the wrong IP address for the default gateway, or there's an issue with the default gateway.
 - B. Log in to another device on the same network segment and ping the default gateway. If you can ping the default gateway from another device, it's an indicator that there's a hardware issue on your computer, the IP address of the default gateway is incorrect on your computer, the netmask on your computer is incorrect, or possibly the host firewall is blocking the network traffic.
5. **Check the DNS server network connection.** If the computer can successfully ping the default gateway, you can be assured that the computer can make connections to other devices on the same network segment, and probably make connections to computers on the Internet or computers on other networks. If you're still having problems connecting to computers on other networks the next thing to test is the connection to the DNS server. This will help eliminate the possibility that the problem is in resolving a DNS name to an IP address.

The first step in performing this test is to find the IP address of the computer's DNS server, which can be found using the `ipconfig /all` command, which you did in step 2. Let's assume the IP address for the DNS server is 192.168.24.21. Next, run ping using the DNS server's IP address as the target for ping. For example:

```
ping 192.168.24.21
```

If the DNS server replies to the ping packets, it means the test is successful and you can proceed to the next step. If this test fails, it means that there's an issue making network connections to the DNS server, or there are problems with the DNS server. To narrow this down you can try connecting to the DNS server from another device on the local network segment. If you can ping the DNS server from a different computer, it's a good sign that you either have the wrong IP address for the DNS server, or there's an issue with the DNS server. If all the computers on the local network segment have issues connecting to the DNS server, you should get in touch with the organization that manages the server and make them aware of the issue.

6. **Ask the DNS server to Resolve a DNS Name.** If the computer can successfully ping the DNS server, the next thing to test is to ensure the DNS server returns an IP address if you send it a DNS name. On windows systems this is done using the `nslookup` utility, while on Linux systems and Macs the `dig` utility program is used. To run `nslookup` on Windows, start the Command Prompt application, then type `nslookup` followed by a DNS name. For example:

```
C:\WINDOWS\system32>nslookup google.com
```

```
Server: UnKnown
```

```
Address: 192.168.1.1
```

```
Non-authoritative answer:
```

```
Name: google.com
```

```
Addresses: 2607:f8b0:400a:806::200e
```

```
142.251.211.238
```

If the `nslookup` query returns an error message saying that it can't find the DNS name you asked for, chances are high that you mistyped the name. If this happens you should try a common name that's easy to type, such as `google.com`.

```
C:\WINDOWS\system32>nslookup nosuchhostmadeupname.com
```

```
Server: UnKnown
```

Address: 192.168.1.1

*** UnKnown can't find nosuchhostmadeupname.com: Server failed

If the nslookup query times out, it's an indication that you have the wrong IP address for the DNS server. At this point you should double check the DNS server's IP address or try running nslookup from another computer. If you can't connect to the DNS server from any computers on the network segment you will have to get in touch with the system administrator for the DNS server.

```
C:\WINDOWS\system32>nslookup google.com
```

DNS request timed out.

timeout was 2 seconds.

Server: UnKnown

Address: 192.168.1.14

DNS request timed out.

timeout was 2 seconds.

DNS request timed out.

timeout was 2 seconds.

DNS request timed out.

timeout was 2 seconds.

DNS request timed out.

timeout was 2 seconds.

*** Request to UnKnown timed-out

7. **Check the Entire Internet.** If you're able to get the DNS server to resolve DNS names, and still can't connect to the Internet or a specific computer on the Internet the next step is to check all the routers between your network and the Internet, or between your network and the specific computer. You'll learn the details about routers in a later section, but there are typically several routers between any home network and the Internet or between any local network segment and the Internet. Any of these routers could be having trouble, preventing network packets from being transmitted. To check this, we use a program called `tracert` which is short for traceroute.

The traceroute program allows you to see the path your network packets take between your computer and a specific network host, reporting back on each successive router that handles the packets. Each time a network packet is sent to another router it's called a hop. Traceroute works by sending ICMP packets, like those used by ping, with ever increasing Time To Live (TTL) values. The TTL is a setting that's used to keep network packets from wandering around the network forever. This is done by setting a max TTL when a packet is first sent, then decreasing it by one each time it makes another hop to a new router. The first packet that traceroute sends out will have TTL=1. The first router that handles the packet, which will be the default router, will send an ICMP response back to traceroute, then decrement the TTL, which in this case will set it to 0. Since the TTL is 0, the router will not forward the packet to the next router, it will simply delete it. Note that traceroute actually sends 3 packets to each router, not one, and will display the round-trip times for each packet.

When the traceroute program receives the ICMP response, it will generate a second ICMP request but this time set the TTL=2. This packet will go to the first router, which will decrement the TTL, making it 1, then send the packet to the next router on the path to the final destination. This second router will send back an ICMP response and decrement the TTL. Since the TTL is 0, the second router will drop the network packet.

This process continues, with the traceroute program sending successive packets and receiving successive ICMP response packets from each router as the packets take one additional hop on each iteration.

```
C:\WINDOWS\system32>tracert google.com
Tracing route to google.com [142.251.211.238]
over a maximum of 30 hops:
  1  1 ms   1 ms   1 ms  192.168.1.1
  2  4 ms   3 ms   4 ms  fdr01.knwc.wa.nwestnet.net [50.46.181.118]
  3  3 ms   44 ms  4 ms  cr1-knwcwaxa-a-be500.bb.as20055.net [64.52.98.0]
  4  10 ms  10 ms  10 ms  lr1-umtloraz-b-be-12.bb.as20055.net [198.179.53.18]
  5  10 ms   9 ms   8 ms  lr1-yakmwaafp-a-be-17.bb.as20055.net [204.11.64.133]
  6  *      *      *      Request timed out.
```

```
7  13 ms  11 ms  32 ms google-sttlwawb-a.pni.as20055.net [107.191.239.0]
8  14 ms  11 ms  11 ms 142.251.229.137
9  11 ms  11 ms   9 ms 216.239.43.121
10 10 ms   9 ms   9 ms sea30s13-in-f14.1e100.net [142.251.211.238]
```

Trace complete.

The output from `tracert` shows the number of hops in the first column, the round-trip times for each of the three ICMP packets sent to each router, the DNS name of the router if it's available, and the IP address of the router. The DNS names for most routers will be hard to decipher because they're typically strings of characters the ISPs and backbone providers use to identify the router, and not something meant for normal user consumption. Sometimes you can tease the location of the router from the name, but not always. This might give you an idea of where the router is physically located. For example, I'm guessing that **lr1-yakmwaftp-a-be-17** is in Yakima Washington, and **google-sttlwawb-a** is in Seattle Washington, but these are just educated guesses.

Another thing to note about the `tracert` output is that many routers on the Internet are configured not to respond to ICMP packets. When `tracert` encounters one of these routers, it will display an asterisk (*) in place of the RTT value for that router. This can make it difficult to pinpoint the exact location of the problem.

If you're having problems connecting to the Internet and use `tracert` as a diagnostic tool, it can be difficult to pinpoint the problem if the problematic router lies outside your network. If the problem with your connection is caused by a router within your LAN, then `tracert` will be able to identify it and provide valuable information for troubleshooting. However, if the problem is caused by a router outside of your LAN, then the results from `tracert` may not be as helpful, especially if the router doesn't respond to the ICMP requests. So, while the bad news may be that you can't connect to the Internet because someone else's router is having problems, the good news is that it's someone else's router having problems, not your router. And, typically these types of router issues will be fixed relatively quickly as they will impact many different networks and users, and there will be lots of unhappy people notifying the router owner of the problem.

There are a few different versions of the tracert tool that you can run on Windows. There's the command line version, or you can download and run a version with a graphical user interface (GUI), or you can find online versions of traceroute, although most online tools show the route back from their web site to a DNS name you provide as opposed to showing the route from your computer to their web site. The GUI based tools are nice because they may have extra features, like showing you where each router is located on a map.

As you troubleshoot network connections, here are a few other things you should keep in mind, and may have to check:

1. Check for any recent changes or updates that could have caused the issue.
2. Disable any firewalls or antivirus software temporarily to see if they are blocking the connection.
3. Check for any known issues or outages with the network or specific websites/services being accessed.

Network Layer and IP Basics – Summary

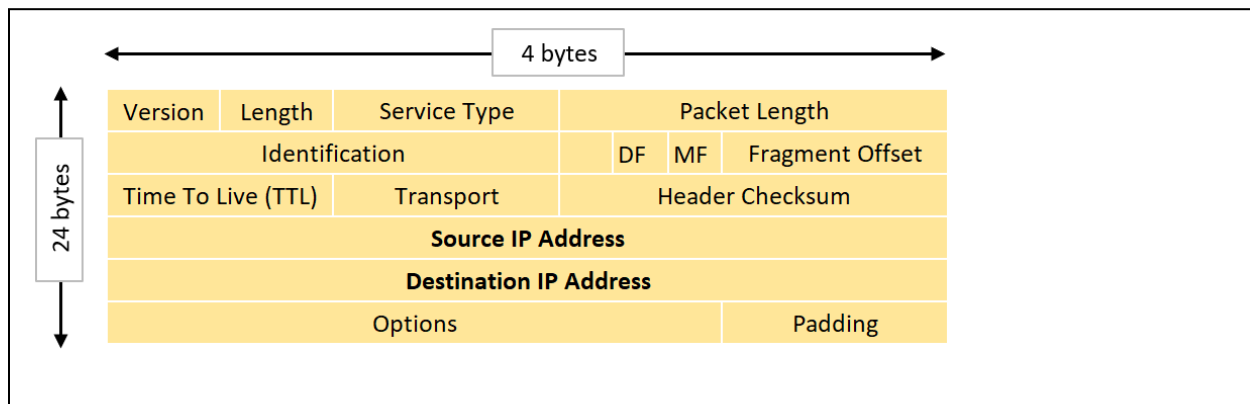
This wraps up our tour of the basics of the Network Layer of the OSI Network Model, and the Internet Protocol. During this initial look at these items, you learned the following:

1. The Network Layer is responsible for end-to-end delivery of network packets, as opposed to point-to-point delivery which is done by the Data Link Layer.
2. The main protocol used at the Network Layer is the Internet Protocol, which defines several things including the format of IP packets, and IP addresses and netmasks.
3. IANA is the group responsible for the overall management of IP addresses, but you will most likely receive an IP address from your ISP.
4. IP addresses are built from 4 octets, with each octet ranging from 0 to 255.
5. An IP address describes two things, a network, and a host number on that network. The netmask is used to determine how to divide and read an IP address.
6. Every computer or device that uses IP must be configured with the IP address of a default gateway/router, which will handle transmitting packets to other network segments.

7. The DNS system consists of many servers whose main role is to resolve DNS names to IP addresses. Every computer on an IP network must be configured with the IP address of a primary and secondary DNS server.
8. Configuring IP on a computer requires setting the IP address, netmask, default gateway/router IP address, and IP addresses of the primary and secondary DNS servers. This configuration can be done manually but is typically done with DHCP.
9. Using DHCP requires setting up a server that will manage and distribute the IP settings for devices on a network.

IP Packet Details

Now that you have a basic understanding of the components used by the Network Layer and the Internet Protocol, and how these components work together to build and deliver network packets, it's time to loop back around and take a deeper dive into the components. The first thing we'll look at is the IP packet format, and all the fields in the IP packet header. Note that it isn't critical that you know this information unless you end up working as a network administrator or plan on taking a certification exam.



Here are the fields in the header, along with a description of what the field contains:

Version (4 bits) - This field indicates the version of the IP protocol being used. This can be 4 or 6, but if it's 6 the rest of the header fields will be different than what's listed here.

Header Length (4 bits) - This field indicates the length of the IP packet header in 32-bit words. This field is necessary because the length of the IP packet header can vary because there are a few optional items

that may or may not be included in fields at the end of the header. If the options field is not present, the header length will be fixed at 20 bytes. However, if options are included, the header length will increase in multiples of 4 bytes, up to a maximum of 60 bytes.

Type of Service (TOS) (8 bits) - This field was meant to be used to indicate the priority and type of service requested for the packet. The TOS field was later replaced by the Differentiated Services (DiffServ) field in IPv4, and the Traffic Class field in IPv6, and most routers ignore the TOS information, so its use is limited and it's not a reliable means of specifying QoS for IP packets in modern networks.

The TOS field is divided into two subfields: the Precedence subfield and the TOS bits subfield. The Precedence subfield is a 3-bit field that is used to specify the priority of the packet. The TOS bits subfield is a 5-bit field that is used to specify the type of service required by the packet, such as low delay, high throughput, or high reliability.

In practice, the TOS field is not widely used and is often ignored by routers. This is because routers primarily use the destination IP address and the protocol type (TCP, UDP, etc.) to determine how to handle a packet. Additionally, modern networks typically use QoS mechanisms such as DiffServ and traffic shaping to prioritize traffic, rather than relying on the TOS field. However, some legacy networks may still make use of the TOS field for QoS purposes.

Total Length (16 bits) - The Total Length field in an IPv4 header is a 16-bit field that specifies the total length of the IP packet in bytes, including both the header and data sections. The minimum value for the Total Length field is 20 bytes, which corresponds to a packet with no data, only the IP header. The maximum value for the Total Length field is 65,535 bytes, which is the maximum size of an IP packet.

Identification (16 bits) - This field is a number that's used to identify fragments of an original packet. When a packet is too large to be sent in a single transmission, it must be divided into smaller fragments that can be reassembled at the destination. The identification field is set to a unique value for each original packet, and all fragments of that packet will have the same value in this field, which tells the recipient they're all part of the same fragmented packet. The receiving device will use this information to reassemble the fragments, also using the Fragment Offset field (see below), to assemble them in the correct order.

The identification field has a minimum value of 0 and a maximum value of 65,535. When the maximum value is reached, the value wraps around to 0 and continues counting.

Flags (3 bits) - This field is used to indicate whether the packet can be fragmented or not by the routers as they forward the packet. Some routers and network links use a smaller Maximum Transmission Unit (MTU), so this field is used to tell them whether the packet can be fragmented, or if the router should just return an error if it can only transmit the packet by breaking it up.

The three bits in this field are:

Reserved: This is always set to zero.

Don't Fragment (DF): If this bit is set to 1, it indicates that the packet should not be fragmented. If a router receives a packet with the DF flag set and the packet needs to be fragmented to be forwarded, the router will drop the packet and send an ICMP "Destination Unreachable: Fragmentation Needed" message back to the sender. This message includes the MTU of the next hop, so the sender can reduce the size of the packet before sending it again.

More Fragments (MF): If this bit is set to 1, it indicates that there are more fragments to follow. If this bit is set to 0, it indicates that this is the last fragment.

Fragment Offset (13 bits) - This field is used along with the identification and Flag fields to determine the position of the current packet within the original packet and reassemble the fragments of the original packet. Here's an example of using the fragment offset field in an IPv4 header:

Suppose that a sender has a large data packet to send to a receiver, but the packet is too large to be sent in a single transmission unit. The sender can break up the packet into smaller fragments, each with its own IP header. The first fragment will have a "more fragments" flag set to 1, indicating that there are more fragments to follow. Each fragment, except the last one, will have a "fragment offset" field indicating the position of that fragment within the original packet. The offset is specified in units of 8

bytes, so if the first fragment is 1500 bytes long, the fragment offset of the second fragment would be $1500/8 = 187.5$, rounded down to 187.

When the receiver receives the fragments, it will use the identification field in the IP header to identify which fragments belong to the same original packet. The receiver will then use the fragment offset fields to reconstruct the original packet by concatenating the fragments in the correct order. The last packet sent will have the "more fragments" flag set to 0 to indicate that this is the last fragment.

Time to Live (TTL) (8 bits) - This field is used to limit the lifetime of a packet. This is necessary because it's possible for packets to get "lost" and wander in circles between routers. The number of hops a packet can make is limited by the TTL field, with each router subtracting one from the TTL as it moves the packet from one network to the next. The TTL field is typically set to 64 and decremented by one by each router that processes the packet, and the packet is discarded if the TTL field reaches zero.

Protocol (8 bits) - This field indicates the protocol used in the data portion of the IP packet. Common protocol values include TCP, UDP, ICMP, and IGMP. You'll learn more about TCP, UDP, and IGMP in later sections.

Header Checksum (16 bits): This field is used to detect errors in the IP packet header. The checksum is calculated over the entire IP packet header, and if any errors are detected the packet is discarded.

It is calculated by taking the 16-bit one's complement of the one's complement sum of all the 16-bit words in the header, with the checksum field itself being set to zero before the calculation is performed. To calculate the checksum, the header is first divided into 16-bit words, and then the sum of all these words is computed, including any carryover from the most significant bit of the previous addition. Once the sum has been computed, the one's complement of the result is taken, and this value is used as the checksum value.

When the packet arrives at the destination, the same calculation is performed on the received packet header, and the calculated checksum is compared with the value in the header. If the two values match then the header is considered valid, and the packet is forwarded. If the checksums don't match, the packet is discarded.

The reason why a checksum is used in the IPv4 header instead of a hash is that a checksum can be computed quickly, with relatively low computational overhead. A hash function, on the other hand, is typically more computationally expensive to compute and would slow down the packet processing.

Source IP Address (32 bits) - This field contains the IP address of the sender of the packet.

Destination IP Address (32 bits) - This field contains the IP address of the intended recipient of the packet. Normally IP packets are sent to a single IP address, in which case we call the transmission a unicast. But there are also ways to send IP packets to multiple devices using something called broadcasts, and multicasts. In IPv4, there are two special addresses that can be used in the destination field for multicast or broadcast:

Broadcast Address: The broadcast address is used to send a packet to all hosts on a particular network segment. The IPv4 broadcast address is typically the highest address in the subnet, where all host bits are set to 1. For example, in a subnet with a netmask of 255.255.255.0, the broadcast address would be 192.168.1.255.

Multicast Address: The multicast address is used to send a packet to a group of hosts that belong to a specific multicast group. The IPv4 multicast address range is from 224.0.0.0 to 239.255.255.255 and is divided into two parts: well-known multicast addresses and dynamically allocated multicast addresses. Well-known multicast addresses are reserved for specific purposes, such as routing protocols, while dynamically allocated multicast addresses are assigned as needed by applications.

Options (variable length) - This field is used to provide additional information about the IP packet. Options are rarely used, but they can include things like security-related information or hints for routers. If included, each option has a two-byte "Option Type" field that identifies the type of option and a one-byte "Option Length" field that specifies the length of the option in bytes. The remainder of the option field contains the actual option data.

Some examples of options that can be included in the options field include:

"Timestamp": This option includes a timestamp that can be used to measure the round-trip time of packets.

"Record Route": This option instructs routers to record their IP addresses in the packet as it passes through them, allowing the sender to trace the route the packet took.

"Security": This option provides security-related information about the packet.

"Strict Source Route": This option specifies a strict path that the packet must follow through the network, allowing the sender to specify the exact route that the packet should take.

Note that the options field is not commonly used in typical IPv4 traffic, and its use is generally limited to specialized applications or situations where additional information or control over the routing of packets is needed.

IP Address Details - How are IP Addresses Assigned

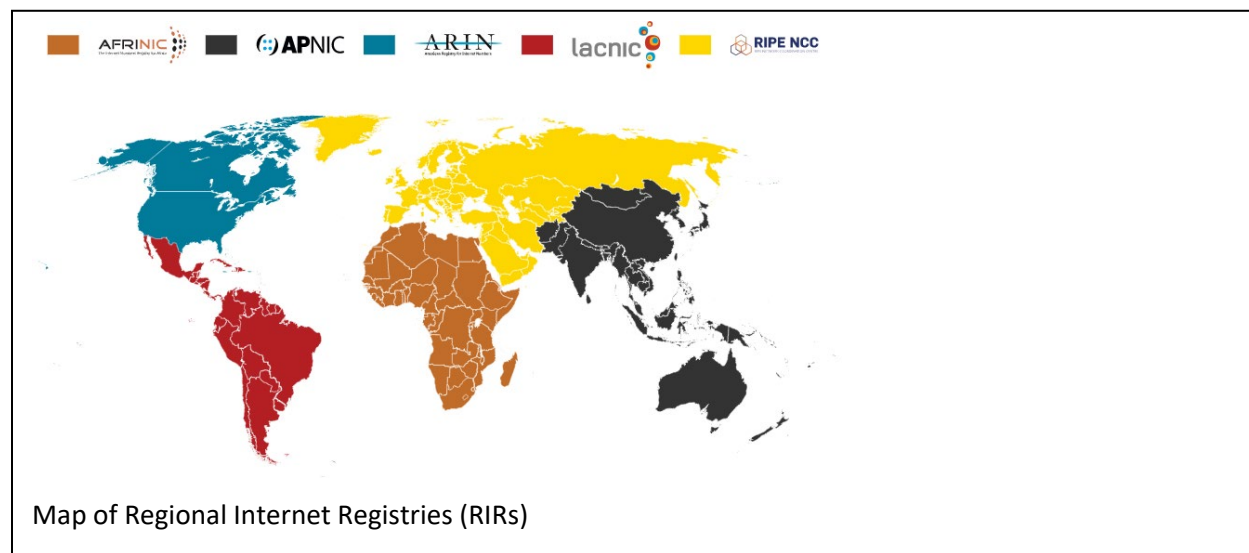
Now let's return and look at the details of IP addresses. We'll start by looking at how IP addresses are assigned. Any computer that wants to communicate on the Internet needs an IP address, but if you want an IP address, or a block of IP addresses, where do you go to get them?

There are two ways or three ways that IP Addresses are assigned. Like phone numbers, you can't just make them up, otherwise there would be the possibility of two devices having the same IP address, and for the Internet to function each device must have a unique address. The process for getting an IP address is a little like getting a phone number. That is, when you buy a new phone, you also need to get a phone number from whichever phone company you select. They will assign you a phone number from their pool of available numbers.

When you connect a computer to the Internet, that computer will be assigned an IP address by the network provider. For home networks and smaller businesses, the network provider is typically an ISP. Larger businesses and organizations will typically connect to the network run by a parent organization.

For example, CBC connects to a network called the K20 network, which is run by the state of Washington, so the college receives their bucket of IP addresses from K20 network admins. Large companies like Microsoft or Boeing will also have a hierarchical network structure, where the company gets a block of IP addresses, which it then subdivides to its networks in different subsidiaries and locations. So, no matter what the situation you will get your IP address(es) from someone above you in the network hierarchy.

But where does your ISP or the person in charge of a large network get their block of addresses? The group in charge of handing out IP addresses is the IANA or Internet Assigned Numbers Authority. The IANA has the responsibility for managing the overall IP address pool, but they farm out the job of actually assigning numbers to groups called the Regional Internet Registries (RIRs)¹⁰. The RIRs are groups within a geographic region that have been given large blocks of IP addresses to distribute by the IANA. If you're a large company or organization that needs a large number of IP addresses to connect computers and devices to the Internet, you must submit an application to one of the RIRs. The number of available IP address blocks is limited, so the RIRs won't hand them out to just anyone. Before assigning a block of addresses the RIR will perform some background checks to authenticate your organization and your



need for the addresses, as well as ensure that your organization has met the technical requirements for connecting your network to the Internet.

¹⁰ <https://www.internetsociety.org/resources/deploy360/2015/short-guide-ip-addressing/>

IP Address Details – Address Blocks

If your organization is approved, the RIR will assign your company a block of addresses¹¹. The block will be one of three sizes, small, medium and large, or tall, grande, or venti for you caffeine aficionados and addicts. The IANA block sizes are called Class A, which is large, Class B, which is medium, and Class C, which is small. Note that Class A, B, and C addresses are now often referred to as legacy address classes, as the current approach to IP addressing is based on a different scheme called Classless Inter-Domain Routing (CIDR).

	Start	End	Subnet Mask	# of Networks	# of Hosts
Class A	1.0.0.0	127.0.0.0	255.0.0.0	126	16,777,214
Class B	128.0.0.0	191.255.0.0	255.255.0.0	16,382	65,534
Class C	192.0.0.0	223.255.255.0	255.255.255.0	2,097,150	254

Table showing Class A, B, and C network blocks, and the associated network addresses.

If you are assigned a Class A block of addresses, the network number will be between 1.0.0.0 and 127.0.0.0. For example, any network that starts with 3.0.0.0 or 67.0.0.0 would be part of a Class A block of IP addresses. Notice that there are only 126 Class A blocks.

The organization that's assigned the Class A block can subdivide it using netmasks any way they wish using a process called subnetting. And anyone that receives a Class A block will certainly use subnets, because if they use the default subnet mask of 255.0.0.0, they will have a single network segment with ~16.7 million devices on the same segment and all these devices will be in the same collision domain.

As an example of subnetting, say you are assigned 67.0.0.0. You can use the subnet mask 255.255.0.0 and hand out network numbers 67.0.0.0 through 67.255.0.0 to the LAN administrators in your organization. This would allow each LAN administrator to either build a single network segment with 65534 hosts, or the LAN administrator could also use netmasks to further subdivide the network address. For example, say you've been assigned the network number 67.143.0.0. You could hand out

¹¹ <https://www.meridianoutpost.com/resources/articles/IP-classes.php>

network numbers 67.143.0.0. through 67.143.255.0. In other words, you could build 255 networks with 254 hosts on each network, which is much more practical solution.

If you are assigned a Class B block of addresses, the network number will be between 128.0.0.0 and 191.255.0.0. For example, any network that starts with 132.86.0.0 or 177.1.0.0 would be part of a Class B block of IP addresses. Notice that there are 16,382 Class B blocks, and each block can hold a maximum of 65534 hosts. Like Class A blocks, Class B blocks are typically subdivided into multiple network numbers. For example, if you used a netmask of 255.255.255.0 with a Class B block of address, it would result in 255 network numbers, with a max of 254 host per network segment, which is once again a much more practical use of the block addresses.

If you are assigned a Class C block of addresses, the network number will be between 192.0.0.0 and 223.255.255.0. For example, any network that starts with 197.0.14.0 or 223.12.10.0 would be part of a Class C block of IP addresses. Notice that there are 2,097,150 Class C blocks, and each block can hold a maximum of 254 hosts. You can also subdivide a Class C block using netmasks, but you'll have to use advanced netmasks which you'll learn about below.

You may have noticed that there are some network addresses that aren't assigned in any of the blocks. There are certain network numbers in each block that are called private network numbers or non-routable network numbers. These network numbers are called private or non-routable because if a router is sent a packet with one of these addresses it will drop it. This means the private addresses can be used on a local network segment, but they'll never be able to reach another network segment or the Internet. This might seem a little weird, but it's actually a great feature that's made it possible to extend the life of IPv4. You'll learn all about this in a minute as it's a key feature of modern-day networking.

Here are the three blocks of private IP addresses in IPv4:

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

The IP address 127.0.0.1 is also not assigned as part of a block, because as you learned this is known as the loopback address, and it is reserved for the internal loopback function of a computer's network interface.

In the olden days, when the Internet was first starting, it was relatively easy to get a Class B or Class C network number. When I worked at Hanford in the 1980s and 1990s, I was able to get two Class B networks, and PNNL also had two class B networks. Since then, Hanford and PNNL returned at least one of the Class B network numbers each, and they may have returned more. You'll learn how they were able to do this and still connect computers to the Internet in a minute when you learn about Private IP Addresses.

Today, it's nearly impossible to get a block of addresses from IANA and RIRs, especially a Class A or Class B block. But even though you can't go directly to IANA or an RIR and get a block of addresses, you can get an IP address or set of addresses. You'll just get them from your ISP or upstream network provider.

IP Address Details – Private or Non-Routable IP Addresses

When the Internet first started the designers planned for growth and thought that having over 4 billion IP addresses would be more than enough. But, as the popularity of the Internet surged, and more and more devices needed an IP address it seemed like 4 billion wasn't going to be nearly enough. One of the problems was the sheer number of devices that wanted to connect and needed IP addresses, and another problem was that since IP addresses are handed out in large blocks there were many addresses that ended up being "wasted". That is, if an organization received a Class B block of 65535 addresses, but only needed 30,000, the remaining 35,000 would not be used, which was a huge waste of numbers.

The number of available addresses started to shrink at the same time that the demand was rapidly growing, forcing the IETF to look for solutions. They came up with two, the first of which was create a new version of IP and increase the size of the IP address, from 4 numbers to 6 numbers. This new address version was introduced in 1998 and is called IPv6¹². While some systems use IPv6, at this point in time IPv4 is still more commonly used. Even though IPv6 was released decades ago there's been a

¹² <http://www.steves-internet-guide.com/ipv6-guide/>

huge lag in adoption. The reason for the delay is that another process, the use of private or non-routable IP addresses, freed up a huge percentage of the IPv4 address pool.

Here are a couple of last notes about IPv6 before we start on non-routable addresses. The first thing is that at this point if someone is talking about IP addresses and doesn't specify a version, they're probably talking about IPv4 because it is still by far the most commonly used. If someone is talking about IPv6, they will spell it out, but if they just say IP address, they are most likely referring to IPv4. The second is that IPv4 was designed in the 1970s¹³, and while the design has proven to be great, by the 1990s it was possible to see areas that could be improved. So, in addition to creating a larger address pool, IPv6 makes some other technical improvements¹⁴ to increase the overall performance. You'll learn more about IPv6 later in the class, but for now we want to concentrate on non-routable IP addresses as they are used everywhere, including your home network.

The first thing we need to do is define what a non-routable IP address is. As the name states, a non-routable address is an IP address that will not pass through a router. When a router sees a network packet where the destination IP address is a non-routable IP address, it will drop the packet. Non-routable IP addresses can be used to send packets to other devices on the same network segment, including the default gateway/router, but they'll never be passed outside of the local network segment. It's important to note that you can still send packets to the default gateway using non-routable IP addresses. You'll see why this is important in a minute.

Non-routable IP addresses are used for building private networks. By private network, we mean a network that may be connected to the Internet, but the devices on the network can't be seen by any devices outside the network. And by outside the network we mean any devices on the non-LAN side of the router. For your home network and most small networks, outside the network means any device on the Internet.

You can think of a private network as something like a secret military base. The base has streets and buildings, and the buildings have street addresses like normal buildings. People inside the base can send

¹³ <https://www.geeksforgeeks.org/history-of-tcp-ip/>

¹⁴ https://www.tutorialspoint.com/ipv6/ipv6_address_types.htm

mail to each other by using the street addresses, but no one outside the base knows anything about the base layout. They don't know the street names or building addresses, they don't even know what streets or buildings exist.

This is what happens with most home networks, and many intranets for larger businesses and organizations. If you look at the IP addresses assigned to the devices on your home network or almost any home network, you'll see that they almost certainly start with 192.168. You'll also see IP addresses that start with 192.168 on almost all the computers at CBC, and IP addresses that start with 172.16 or 192.168 on most computers and devices connected to a network that's connected to the Internet.

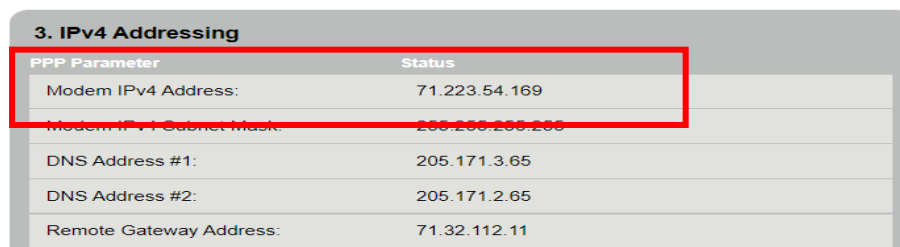
But how is this possible? After all, isn't it a requirement for every computer and device connected to the Internet to have a unique IP address? Here's how the non-routable addresses have been implemented and how most network connections are made today. This is made possible through a system called Network Address Translation or NAT, which runs on the LAN routers or your home router.

The NAT system acts like the mailroom workers at the secret military base. Anytime someone wants to send mail to the outside world, they first send it to the mailroom. The workers in the mailroom write down some data about your message, mainly your address as the original sender, and the address you're sending the message to. They then take the message out the envelope you used and place it in a new envelope. When they address the new envelope they use the original recipient address, but they use the address for the base mail room as the sender's address.

They send your message in their new envelope to your original recipient. The recipient receives your message, and when they address their reply, they look at the envelope they got the message in and see that it came from the base mail room, so they send their message back to the base mail room address. In fact, they have no idea what your real address is, since it's nowhere on the envelope the message came in. They send their reply, and since it's sent to the base mail room when it arrives it's delivered to the mail room. The mail room inspects their list and sees that it's a response to a message you sent. They take the message out its envelope and place it in a new envelope with your address on the secret base as the recipient address. The process that the mail room uses, keeping lists of outbound mail, changing envelopes and addresses, and using their lists to forward incoming mail is known as address translation.













This system has a couple of benefits. The first is increased privacy, since any mail that is sent out from the base looks like it's coming from a single address and no one on the outside world can gather any new information about the buildings on the base by looking at the sender's address. The second is that addresses inside the base can be exact duplicates of addresses on other bases as these private addresses are never seen by anyone outside of the base.

On a network the address translation works the same way as the mail room process, with the LAN router doing the translation. Let's use a home network for a walk through, but the process will be the same for a LAN run by any organization. When you connect your home network to your ISP one of the first things that happens is your router sends a signal to the ISP saying it's ready to go to work by sending out a DHCP request. Your ISP will then assign an IP address to your router from its pool of available addresses. The following figure shows the IP configuration for a wireless router which is connected to the ISP CenturyLink, and CenturyLink assigned the wireless router the address 71.223.54.169.




PPP Parameter	Status
Modem IPv4 Address:	71.223.54.169
Modem IPv4 Subnet Mask:	255.255.255.255
DNS Address #1:	205.171.3.65
DNS Address #2:	205.171.2.65
Remote Gateway Address:	71.32.112.11

If we look at the devices connected to the wireless router, we can see that they've all been assigned addresses that start with 192.168.0 which is the Class C non-routable network number. Almost all home routers also act as DHCP servers, which is how each device on this network got its IP address. That is, when the device connected to the wireless network, the device also sent a DHCP request, and the router, acting as a DHCP server assigned the device their IP address, netmask, default gateway address, and DNS server addresses.

DEVICE NAME	IP ADDRESS	MAC ADDRESS	CONNECTION TYPE
 Unknown	192.168.0.5	00:f3:61:a7:64:f8	 5 GHz CenturyLink3272
 amazon-9412c0d28	192.168.0.86	44:d5:cc:cd:f1:0b	 5 GHz CenturyLink3272
 amazon-987f40404	192.168.0.87	dc:91:bf:d0:09:17	 5 GHz CenturyLink3272
 TONY-s-S21-Ultra	192.168.0.104	d6:e4:a8:d8:80:85	 5 GHz CenturyLink3272
 CVD-Q2827	192.168.0.103	84:5c:f3:ca:82:b7	 5 GHz CenturyLink3272
 YSPC	192.168.0.105	14:18:c3:73:e0:79	 5 GHz CenturyLink3272

Whenever a device on the home LAN sends a network packet to the Internet, the packet will be grabbed by the router. For example, let's say the computer with the IP address 192.168.0.103 wants to view a web page at google.com. The computer builds the network packet asking for the web page, using Google's IP address as the destination IP and 192.168.0.103 as the sender's IP address.

The computer then uses the netmask and determines that google.com is on a different network segment, so it builds an ethernet frame, places the IP packet in the data section of the ethernet frame, and addresses the frame to the default gateway's MAC address, which in this case the wireless router. The home router unpacks the ethernet frame, sees the network packet, and knows it needs to route it upstream to get it to google.com. But before the router forwards the network packet it will run the NAT process and record some data about your network packet. The router will record your device's IP address as the original sender, and the IP address of the device the packets are being sent to. The router will then build a new network packet sending it to the original recipient IP address but changing the sender's address to the "outside" address used by the router, in this case 71.223.54.169. The following figure shows an example NAT table, used by the router to keep track of outgoing network packets.



NAT Status

Protocol	Timeout	Source IP	Source Port	Destination IP	Destination Port
tcp	0	192.168.0.1	443	192.168.0.103	62568
tcp	0	192.168.0.1	443	192.168.0.103	62569
tcp	0	71.223.54.169	44958	184.72.149.114	443
tcp	0	127.0.0.1	47732	127.0.0.1	12345
tcp	0	127.0.0.1	12345	127.0.0.1	47732
tcp	0	192.168.0.1	443	192.168.0.103	62572
tcp	0	192.168.0.1	443	192.168.0.103	62571
tcp	0	192.168.0.1	443	192.168.0.103	62562
tcp	0	192.168.0.1	443	192.168.0.103	62570
udp	0	71.223.54.169	50191	34.211.237.22	6363
tcp	7370	192.168.0.104	51942	35.82.251.20	443
tcp	7308	192.168.0.104	41044	172.217.14.67	443
tcp	7307	192.168.0.104	56810	142.250.141.188	5228

When Google gets the network packet, it will build a response and send it back to 71.223.54.169 which is the IP address of your router. When router receives any network packets from its external interface, it inspects its NAT list to see which device made the original request. In this case the router sees that this packet is a response to the request you sent to Google. To get the network packets back to your computer, when the router receives the packet it extracts the data, then builds a new network packet,

placing the data from Google inside the new packet, and using your computer's IP address as the recipient address.

Since your IP address isn't in the IP packet sent to google, google.com and any other routers that forwarded the network packets won't know that the packet actually came from your computer or your computer's IP address. This keeps the IP addresses of all the devices on your network private and hidden from the outside world.

Besides keeping your internal IP addresses private, using non-routable addresses has another huge benefit. Most devices on local network segments can use private addresses, and the only device on a network segment that needs a "real" IP address is the default gateway. This frees up almost all of the IPv4 addresses that were previously being used by devices on local network segments, which has greatly extended the life and usefulness of IPv4.

IP Address Details – Advanced Subnetting & Netmasks

Now let's turn our attention to netmasks and advanced subnetting. Let's start by doing a quick review of why a network administrator would choose to sub-divide a network number. This is done to make network numbers more flexible, make more efficient use of the IP addresses available for a given network number, and at the same time reduce the number of devices in a collision domain, especially for organizations that received large blocks of IP addresses. That is, say you were an organization and receive a Class B block of addresses that use the network number 131.27.0.0. This means that you have all the IP addresses between 131.27.0.1 and 131.27.255.254, or 65534 IP addresses you can assign to devices on your network. But you certainly don't want all 65534 devices to be connected to the same LAN because this would require that all the devices be connected to the same network segment or plugged into the same router because this would be physically impossible. And all the devices would be in the same ethernet collision domain, which means there would be a lot of contention for the network media, lots of collisions, which would cause lots of delay for network traffic.

So instead of having one large network, you'll probably want to subdivide your network using the netmask. There are three types of netmasks, that I'm going to call simple netmasks, binary netmasks, and complex binary netmasks.

The simple netmasks are the ones that you've already learned about, where the bits in an octet are either set to all 1s or all 0s. These consist of 255.0.0.0, 255.255.0.0 and 255.255.255.0. These netmasks make it really easy find the network portion of an IP address because it will consist of an entire octet.

The binary netmasks and complex binary netmasks are netmasks that require using binary numbers to determine the network and host portions of each address. They're used like super simple netmasks, where the 1 bits in the netmask designate the network, but the numbers in the octets won't be 255. These types of netmasks aren't as easy for humans to use because you have to write the netmask in binary, but they provided added versatility and more efficient use of IP addresses. For example, even a class C subnet mask of 255.255.255.0 is not very efficient as it only allows 1 network with 254 hosts, and this is a large number of devices to connect to a single network segment and a large number of devices in a single collision domain. With a binary netmask you could subdivide this into two networks, or 4 networks, and with a complex binary netmask you could subdivide it into 5 networks.

Here are some examples of binary netmasks that will illustrate what they look like and how they're used.

Binary Netmask - Example 1

Let's start with an example of subnetting the class C address 212.13.177.0, dividing it into two subnets. To do this we'll use a netmask of 255.255.255.128. The first step to understanding what's going on with this netmask is to write it out in binary. In this case the binary netmask is:

```
11111111.11111111.11111111.10000000
```

To use this to divide an IP address into the network portion and the host portion, find all the bits that are set to 1. The bits that are set to 1 represent the network portion, and the bits that are set to 0 represent the host portion.

Now let's look at using this netmask to decide if two IP addresses, 212.13.177.14 and 212.13.177.134, are on the same network segment or not. The first step is to write each address in binary, then we'll compare the network portion of the addresses to determine if they're on the same subnet or not.

212.13.177.14 = 11010100.00001101.10110001.00001110

212.13.177.134 = 11010100.00001101.10110001.10000110

When we apply the netmask to each number, we can skip the first three octets because they're obviously the same. I'm going to write the last octet of binary numbers for the netmask and both IP addresses on subsequent lines to make it easy to see the network bits, and if they match or not.

Netmask: 10000000

IP 1: 00001110

IP 2: 10000110

In this case the two IP addresses are on different networks as IP address 1 has a 0 in the network portion and IP address 2 has a 1.

Now let's check the two IP addresses 212.13.177.14 and 212.13.177.68. These numbers in binary are:

212.13.177.14 = 11010100.00001101.10110001.00001110

212.13.177.68 = 11010100.00001101.10110001.01000100

You might be able to see the answer already, but let's write out the last octet of the netmask and the two IP addresses to make it crystal clear.

Netmask: 10000000

IP 1: 00001110

IP 2: 00000100

In this case the two IP addresses are on the same network as IP address 1 has a 0 in the network portion and IP address 2 also has a 0.

The next thing to see is how many networks and how many hosts per network we can achieve with this netmask. In this case, the network bit can be set to 0 or 1, which means that we can only have two different (sub) networks. For the number of hosts per network, we have 7 bits to work with. This means that we can have 2^7 or 128 total addresses per network. But we don't use 0 because this represents the network, and we don't use all 1's, or 127 because this is the broadcast address. Subtracting these two possible host values leaves 128-2 or 126 possible hosts.

Binary Netmask - Example 2

Let's do another example of subnetting the class C address 212.13.177.0, this time dividing it into four subnets. To do this we'll use a netmask of 255.255.255.192. In this case the binary netmask is:

11111111.11111111.11111111.11000000

Notice that this is just like the 255.255.255.168 netmask, it just has one more 1 bit. Once again, to use this to divide an IP address into the network portion and the host portion, find all the bits that are set to 1. The bits that are set to 1 represent the network portion, and the bits that are set to 0 represent the host portion.

Now let's look at using this netmask to decide if two IP addresses, 212.13.177.10 and 212.13.177.234, are on the same network segment or not. The first step is to write each address in binary, then we'll compare the network portion of the addresses to determine if they're on the same subnet or not.

212.13.177.10 = 11010100.00001101.10110001.00001010

212.13.177.134 = 11010100.00001101.10110001.11101010

Once again, once you write the IP addresses in binary the answer is probably clear, but let's complete the process by writing the last octet of the netmask and the two IP addresses to check. I'm going to skip

the first three octets because they're still the same, and only write the last octet of binary numbers for the netmask and both IP addresses on subsequent lines to make it easy to compare the network bits.

Netmask: 10000000
IP 1: 00001010
IP 2: 11101010

In this case the two IP addresses are on different networks as IP address 1 has a 00 in the network portion while the network bits in IP address 2 are 11.

Now let's check the two IP addresses 212.13.177.14 and 212.13.177.33. These numbers in binary are:

212.13.177.14 = 11010100.00001101.10110001.00001110

212.13.177.68 = 11010100.00001101.10110001.00100001

You might be able to see the answer already, but let's write out the last octet of the netmask and the two IP addresses to make it crystal clear.

Netmask: 10000000
IP 1: 00001110
IP 2: 00100001

In this case the two IP addresses are on the same network as both IP addresses have a 00 in the network portion.

The next thing to see is how many networks and how many hosts per network we can achieve with this netmask. In this case, the network bit can be set from 00 to 11, which means that we can only have four different (sub) networks. For the number of hosts per network, we have 6 bits to work with. This means that we can have 2^6 or 64 total addresses per network. But we don't use 0 because this represents the network, and we don't use all 1's, or 63 because this is the broadcast address. Subtracting these two possible host values leaves $64-2$ or 62 possible hosts.

As you can see, each time we add a bit to the network portion of the netmask, we subtract a bit from the host portion. Adding bits to the network portion allows for more possible networks, but fewer hosts on each network. The following table shows the possible values for the binary netmasks for subnetting a Class C network, along with the number of network addresses and the possible number of hosts and number of usable hosts per network. Note that you'd never use the last two netmasks as they don't have any usable host numbers.

Subnet Mask	Binary Netmask (Last Octet)	Networks	Hosts	Usable Hosts	Broadcast Address	CIDR
255.255.255.0	00000000	1	256	254	0.0.0.255	/24
255.255.255.128	10000000	2	128	126	0.0.0.127	/25
255.255.255.192	11000000	4	64	62	0.0.0.63	/26
255.255.255.224	11100000	8	32	30	0.0.0.31	/27
255.255.255.240	11110000	16	16	14	0.0.0.15	/28
255.255.255.248	11111000	32	8	6	0.0.0.7	/29
255.255.255.252	11111100	64	4	2	0.0.0.3	/30
255.255.255.254	11111110	128	2	0	0.0.0.1	/31
255.255.255.255	11111111	256	0	0	0.0.0.0	/32

You could also use this process to subnet a Class B or Class A network using a binary netmask. I'm not going to cover the process here as the idea is impractical. That is, there are good reasons to make subnet to create more networks, even if they have fewer hosts per network. But there's not a good reason to take a Class B or Class A network and create subnets with 500 or more hosts per network. We'll go through one example of using a binary netmask on a Class B address, but we're not going to look at every possible netmask. The only time I suggest you go through the exercise of using binary subnet masks on Class B or Class A networks is if you're planning on taking a certification exam on networking like the Cisco CCNA or CompTIA Network+.

Binary Netmask - Example 3

Here's an example of subnetting the class B address 155.13.0.0 using a netmask of 255.255.254.0. In this case the binary netmask is:

11111111.11111111.11111110.00000000

Notice that this is much like the 255.255.255.0 netmask, it just has one less 1 bit. Like always, to use this to divide an IP address into the network portion and the host portion, we need to first find all the bits that are set to 1 as they will represent the network portion, while the bits that are set to 0 represent the host portion.

Now let's apply this netmask to decide if two IP addresses, 155.13.8.10 and 155.13.177.199, are on the same network segment or not. The first step is to write each address in binary, then we'll compare the network portion of the addresses to determine if they're on the same subnet or not.

155.13.8.10 = 10011011.00001101.00001000.00001010

155.13.177.199 = 10011011.00001101.10110001.11000111

Once again, once you write the IP addresses in binary the answer is probably clear, but let's complete the process by writing the third octet of the netmask and the two IP addresses to check. I'm going to skip the first two octets because they're obviously the same for both IP addresses, and also skip the last octet since it's all host bits. This means we'll only write the third octet of binary numbers for the netmask and both IP addresses on subsequent lines to make it easy to compare the network bits.

Netmask: 11111110

IP 1: 00001000

IP 2: 10110001

In this case the two IP addresses are on different networks as the network portions of the IP addresses are different.

The next thing to see is how many networks and how many hosts per network we can achieve with this netmask. In this case, the network bit can be set with 7 bits which means that we have 2^7 or 128 (sub) networks. For the number of hosts per network, we now have 9 bits to work with, 1 from the third octet plus the 8 from the fourth octet. This means that we can have 2^9 or 512 total addresses per network. But we don't use 0 because this represents the network, and we don't use all 1's because this is the broadcast address. Subtracting these two possible host values leaves $512-2$ or 510 possible hosts.

Complex Binary Subnet Masks

If you're liking counting in binary, then you're now in for a treat as we look at complex subnet masks. If you're not digging binary, you might not like this part so much. But keep in mind you won't need to know this unless you work as a network administrator or want to try a certification exam. So far, the netmasks we've looked at, both the simple netmasks and the binary netmasks have had a solid row of either 255's or 1's if we're counting in binary. In technical terms we describe the bits in these numbers as being contiguous, which means all the 1's are adjacent without any 0's between any of the 1's. I'm not sure who invented the term contiguous, and sometimes wonder if it came to be when some tech writer misspelled continuous. In any case, a complex subnet mask is one that has bits set to 1 that are not contiguous. A complex subnet mask can be used to create a subnet that is not a power of two. For example, a complex subnet mask of 255.255.255.248 has 29 network bits and 3 host bits. This type of netmask is extremely versatile, and it can allow a network administrator to make the optimal use of their pool of IP addresses.

Binary Netmask - Example 4

Here's an example of subnetting the class C address 212.13.177.0 using the netmask 255.255.255.217. The first step to understanding what's going on with this netmask is to write it out in binary. In this case the binary netmask is:

11111111.11111111.11111111.11011001

To use this to divide an IP address into the network portion and the host portion, find all the bits that are set to 1. The bits that are set to 1 represent the network portion, and the bits that are set to 0 represent the host portion.

Now let's look at using this netmask to decide if two IP addresses, 212.13.177.14 and 212.13.177.134, are on the same network segment or not. The first step is to write each address in binary, then we'll compare the network portion of the addresses to determine if they're on the same subnet or not.

212.13.177.14 = 11010100.00001101.10110001.00001110

212.13.177.134 = 11010100.00001101.10110001.10000110

When we apply the netmask to each number, we can skip the first three octets because they're obviously the same. I'm going to write the last octet of binary numbers for the netmask and both IP addresses on subsequent lines to make it easy to see the network bits, and if they match or not.

Netmask: 11011001

IP 1: 00001110

IP 2: 10000110

In this case the two IP addresses are on different networks as the network bits in the IP addresses are different.

Now let's check the two IP addresses 212.13.177.78 and 212.13.177.108. These numbers in binary are:

212.13.177.14 = 11010100.00001101.10110001.01001110

212.13.177.68 = 11010100.00001101.10110001.01101100

You might be able to see the answer already, but let's write out the last octet of the netmask and the two IP addresses to make it crystal clear.

Netmask: 11011001
IP 1: 01001110
IP 2: 01101100

In this case the two IP addresses are on the same network as they have identical bits in their respective network portions.

DNS Basics

Another important component used by the Network Layer is DNS (Domain Name System). DNS is a protocol combined with a distributed system of servers that are used to translate human-friendly domain names such as google.com or columbia.edu into IP addresses. DNS provides humans with a way to use the names of web sites and network devices instead of trying to remember IP addresses when we want to send email or browse the web. This is just like remembering someone's name as opposed to remembering their phone number. I don't know about you, but I'd much rather ask my phone to call Saul than trying to remember that Saul's phone number is 505-503-4455. But, since the Network Layer must have IP addresses to build and send IP packets the DNS system is essential to the functioning of the Network Layer and the Internet.

DNS History

The Domain Name System (DNS) has its roots in the early days of the Internet. In the 1970s, the ARPANET (the precursor to the Internet) used a central file to map human-readable hostnames to numerical IP addresses. The file was named HOSTS.TXT and it was maintained by the Stanford Research Institute (SRI). This file had to be manually updated and distributed to all the computers on ARPANET. Whenever a new system was added, the computer owner would call SRI and ask them to add the new computer and the computer's IP address to the file. This system worked fine in the early days when there weren't many computers on the network, but as more and more systems joined ARPANET using a

single centralized file became increasingly difficult, and soon it became obvious that a better system was needed.

In the early 1980s, the need for a more scalable and automated system for resolving domain names to IP addresses led to the development of the Domain Name System or DNS. John Postel oversaw the ARPANET list, and he gave the task of selecting a new system to Paul Mockapetris who had to choose from 5 candidate systems. Mockapetris ended up developing his own system, creating the basic DNS architecture, and naming conventions that are still used today.

The first actual implementation of Mockapetris' DNS was created in 1984 by four students at the University of California, Berkeley, who wrote the Berkeley Internet Name Domain (BIND) software. BIND quickly became the de facto standard for DNS software and became formalized in 1987 with the publication of RFC 1035, which defined the structure and operation of the DNS system. In the 1990s BIND and DNS were also adopted by Microsoft. Since then, DNS has undergone numerous changes and improvements to address issues such as security and scalability. Today, the DNS system is an essential part of the Internet infrastructure, consisting of a highly distributed network of servers and caches that work together to efficiently resolve domain names to IP addresses, enabling users around the world to access online resources quickly and easily.

DNS Names

You've probably been using DNS names your entire life, but just in case you don't know the basics of their format here's a little background information. Each domain name consists of at least two strings separated by dots, with the rightmost string indicating the top-level domain or TLD. The top-level domain names are used to categorize domain names based on their purpose or geographic location. There are two main categories of TLDs, generic top-level domains (gTLDs) and country-code top-level domains (ccTLDs). The first TLDs were created in 1984 and included seven gTLDs: .com, .edu, .gov, .mil, .net, .org, and .arpa. Generic top-level domains (gTLDs) are TLDs that are not restricted to a particular country or region, and are intended for use by individuals, organizations, and businesses worldwide. In contrast, country-code top-level domains (ccTLDs) are two-letter codes that are assigned to specific countries or territories, such as .us for the United States, .uk for the United Kingdom, and .de for Germany.

A couple of interesting country codes are .fm which is assigned to the Federated States of Micronesia. This was a windfall for Micronesia, since they can sell DNS names in this domain to the many FM radio stations want their DNS name to end in .fm. Similarly .cc is assigned to the Cocos (Keeling) Islands, a territory of Australia, who can sell DNS names to community colleges, and .tv is assigned to Tuvalu, a small island nation, which can sell DNS names to television stations and shows that want a DNS name that end in .tv.

The distinction between gTLDs and ccTLDs is important because it affects how domain names are registered and used. While gTLDs are available for registration to anyone, ccTLDs may have specific registration requirements or restrictions based on the country or region they represent.

Up until 1998 the TLDs were managed by the Internet Assigned Numbers Authority (IANA), at which time management was transferred to the Internet Corporation for Assigned Names and Numbers (ICANN). ICANN is a non-profit organization that is responsible for managing and coordinating the DNS system, including the allocation of TLDs. Today, there are hundreds of TLDs, including many new gTLDs that were added as part of a major expansion of the DNS system in 2013. The expansion of TLDs was intended to provide greater choice and flexibility for domain name registration, and to support the growth of the Internet in new regions and industries. The expansion included the addition of hundreds of new gTLDs, such as .app, .blog, and .club, as well as many new ccTLDs.

How an Organization Gets a DNS Name

The process of obtaining a DNS name involves several steps, blending administrative, technical, and sometimes legal considerations. Here are the steps:

1. Choose the Domain Name. While this might seem simple it can take some thought as the name should be relevant to the organization's name, brand, or mission, easy to remember, spell, and pronounce, and most important is must be available. Quite often an organization's first choice for a domain name may already be taken. If it is already taken the organization may try and negotiate with the current owner to see if they're willing to sell the name, or the organization can try a different name.

Active Domain Names: 2024 Statistics

As of the end of the fourth quarter of 2024, there were approximately 364.3 million domain name registrations across all top-level domains (TLDs) worldwide. These figures provide insight into the scope and scale of the modern internet.

Breakdown by TLD Category

.com and .net TLDs:

- Combined, these two TLDs accounted for 169.0 million registrations.
- .com: 156.3 million
- .net: 12.7 million

Country-Code TLDs (ccTLDs):

- These include domains like .cn (China), .de (Germany), and .uk (United Kingdom).
- Totaling 140.8 million registrations.

New Generic TLDs (ngTLDs):

- Examples include .xyz, .app, and .online.
- With 35.4 million registrations.

Other Legacy Generic TLDs:

- Such as .org, .info, and .biz.
- Comprising 17.3 million registrations.

These figures are sourced from the Domain Name Industry Brief (DNIB) published by VeriSign, which provides comprehensive statistics on domain name registrations. For the full report, visit:

<https://www.dnib.com/articles/the-domain-name-industry-brief-q4-2024>

2. **Select a Domain Registrar.** A domain registrar is a company accredited to sell domain names. Registrars are accredited through organizations like ICANN (Internet Corporation for Assigned Names and Numbers) for generic domains or National agencies for country-code domains. Examples of registrars include GoDaddy, Google Domains, Hover, etc. Choosing a registrar depends on factors like pricing, customer support, and additional services (like hosting or security)
3. **Register the Domain.** Once an organization has selected a DNS name and a registrar, they work with the registrar to make their DNS name official and get it out on the DNS servers on the Internet. Most

of the registration process consists of filling out administrative information such as the contact information for the domain owner, a technical contact for the domain, billing info, etc.

4. Configure the DNS Settings. The registrar will then build the DNS records that join or associate the DNS name with the IP address(es) associated with the name. You can think of these records as being like database fields, but they're not in a database, they're just text entries in a file that's read by a DNS server. You can also think of the DNS records as being like phone book entries whose main purpose is to associate a name with a phone number. The main DNS record is called an A Record, where A stands for Address. This is the record that associates a DNS name with an IP address. There are several other types of records including CNAME Records, which are canonical names or nicknames for a site, and MX Records, which point to the mail server for a site.
5. Load the Records on a DNS server. Once the records are built, they need to be loaded onto a DNS server. The DNS system is distributed and there isn't one DNS server that holds all the DNS information for the entire system. In this distributed system larger organizations will run their own DNS servers, which would be like maintaining their own section of the phone book. While running a DNS server isn't difficult, it does take some work so most smaller organizations or individuals who use a service like GoDaddy or Google Sites to build their web site will contract with a commercial DNS server like Cloudflare or AWS Route 53 and have them host their DNS records. In the phone book analogy, contracting with a commercial DNS server is like adding a phone number to a larger phone book. Once the DNS Records are loaded on a DNS server it can take several hours for it to propagate through the DNS system on the Internet and become active.

The Domain Name Gold Rush: A Modern Digital Land Grab

In the early days of the internet, the domain name system (DNS)—essentially the address book of the web—became a new frontier of opportunity. By the mid-1990s, registering a domain cost as little as \$70, and early adopters realized that these virtual addresses could become valuable assets. Much like staking a gold claim in the 19th century, people began registering domain names they thought might someday fetch a profit. This speculative era quickly earned the nickname the domain name gold rush.

Some of the first big domain sales were completely legal and incredibly lucrative. For example, business.com was originally purchased in the mid-1990s and later sold in 1999 for \$7.5 million. It eventually became a business-to-business search engine and directory. Similarly, beer.com was sold for \$7 million, and vacationrentals.com was purchased for \$35 million—reportedly to keep it out of a competitor's hands (Kinsella).

These cases didn't involve trademark infringement because the names were either common nouns or descriptive terms. In this sense, domain names were treated like digital real estate: you could buy a good location (name) and sell it to the highest bidder.

As the internet grew, opportunists began registering names associated with companies, public figures, and major brands. The goal was often to sell the name back to the rightful owner for a profit—a practice that came to be known as cybersquatting.

One of the first and most important legal cases to address this was *Panavision Int'l, L.P. v. Toeppen*. Dennis Toeppen, an early cybersquatter, registered panavision.com and offered to sell it back to the camera company for \$13,000. The company refused and sued. In a landmark decision, the Ninth Circuit ruled that registering a domain name to extract money from a trademark holder diluted the brand and constituted bad-faith behavior under trademark law (*Panavision v. Toeppen*).

Toeppen had also registered domains like deltaairlines.com and neiman-marcus.com. This case became a model for others, demonstrating that U.S. courts were willing to treat domain names as trademark-sensitive property when used in bad faith.

The Domain Name Gold Rush: A Modern Digital Land Grab (Continued)

Another famous case involved Nissan Motor Co. v. Nissan Computer Corp. In this instance, Uzi Nissan had registered nissan.com for his small business years before the car company attempted to take control of the domain. The court ruled that while Nissan Motors could not take the domain away, Uzi Nissan was prohibited from monetizing it through auto-related advertisements that could confuse consumers (Nissan Motor Co. v. Nissan Computer Corp.).

To combat cybersquatting on a broader scale, two major legal tools were introduced around the same time:

1. The Anticybersquatting Consumer Protection Act (ACPA), enacted in 1999, allowed trademark owners to file lawsuits in U.S. courts to recover domain names registered in bad faith. The law specifically targeted those who registered domains “with a bad faith intent to profit” from someone else’s trademark (ACPA §1125(d)).
2. The Uniform Domain-Name Dispute-Resolution Policy (UDRP), developed by the Internet Corporation for Assigned Names and Numbers (ICANN), provided a faster and less expensive way to resolve disputes. Cases could be brought before arbitrators at organizations like the World Intellectual Property Organization (WIPO) without going to court. Thousands of domain names have been transferred through this mechanism.

It wasn’t just businesses that had to defend their names—celebrities also found themselves in disputes over their identities online. In 2000, the pop star Madonna won a UDRP case against Dan Parisi, who had registered madonna.com. WIPO ruled that the name was being used in bad faith and ordered the transfer of the domain to the singer (Madonna v. Parisi).

However, not every celebrity succeeded. Musician Sting, whose legal name is Gordon Sumner, filed a complaint against a registrant who had owned sting.com for use in online gaming. WIPO ruled against Sting, determining that the registrant had a legitimate interest in the domain and had not acted in bad faith (Sumner v. Urvan). This case highlighted the limits of the UDRP when there is no clear intent to exploit a famous name commercially.

The Domain Name Gold Rush: A Modern Digital Land Grab (Continued)

The domain name gold rush revealed something essential about the early internet: even in a virtual world, control over naming and identity matters deeply. The period produced both legitimate business stories and cautionary tales about overreach. Over time, legal frameworks like the ACPA and UDRP brought order to the chaos, allowing trademark owners to defend their brands and reclaim their names.

Today, most high-value domains are already owned, and many are protected by robust legal frameworks. Still, the legacy of those early digital pioneers and opportunists lives on—in the names we type every day and the court decisions that helped shape modern internet law.

Works Cited

Panavision Int'l, L.P. v. Toeppen, 141 F.3d 1316 (9th Cir. 1998).

Nissan Motor Co. v. Nissan Computer Corp., 378 F.3d 1002 (9th Cir. 2004).

Madonna v. Dan Parisi and 'Madonna.com', WIPO Case No. D2000-0847, 12 Oct. 2000. World Intellectual Property Organization.

<https://www.wipo.int/amc/en/domains/decisions/html/2000/d2000-0847.html>

Gordon Sumner p/k/a Sting v. Michael Urvan, WIPO Case No. D2000-0596, 20 July 2000. World Intellectual Property Organization.

<https://www.wipo.int/amc/en/domains/decisions/html/2000/d2000-0596.html>

United States Congress. Anticybersquatting Consumer Protection Act. 15 U.S.C. § 1125(d). Enacted 29 Nov. 1999.

Kinsella, Warren. "Domain Name Mania." Forbes, 3 Aug. 2000,

<https://www.forbes.com/2000/08/03/0803domains.html>

DNS Components

In previous descriptions of how DNS works, the steps in the process for resolving a DNS name to an IP address was grossly oversimplified. While this simple explanation may be accurate enough when you're first learning about the Network Layer, the process is much more involved and it's important that you

learn the steps in the actual process if you want to be able to configure and troubleshoot DNS connections. In this section you'll learn about the DNS process by first learning about the components of the system, then walking through the steps required to resolve a DNS name to an IP address.

The first thing to learn is that the DNS system is composed of several components including the client, a resolver, three levels of servers, and an application protocol. The two easiest to understand components are the client, which is any device that makes a request to resolve a DNS name to an IP address, and the DNS protocol which is a set of rules that defines the format for the DNS requests and responses¹⁵.

The DNS Resolver is the server on the client side, that directly interfaces with the clients. Anytime a client needs an IP address it asks the DNS Resolver for help. The DNS Resolver then uses the other DNS servers to track down the information, and once it has IP address, it returns it to the client. You'll learn how a DNS Resolver works with the other servers to find an IP address in bit, but for now the important point is that the clients only talk to the DNS Resolver, and the DNS Resolver is the device that does all the work in resolving the DNS name to its corresponding IP address. Another important thing to note is that the DNS Resolver is the device that Microsoft calls the DNS Server in the IP configuration. Calling this device a *DNS server* is a little unfortunate, as in the actual DNS system there are three types of servers, and the DNS Resolver is not one of these three servers.

On the server side of DNS there are three main components or three types of servers, DNS root servers, Top Level Domain (TLD) servers, and DNS servers. The reason that there are so many components to DNS is due to the large number of domain names, and the large volume of requests for name resolution. As of the end of the fourth quarter of 2024, there were approximately 364.3 million domain name registrations¹⁶ and in late 2022 the DNS system handled an average of over 1 million queries per second or 60 million per minute.¹⁷

While DNS could have used a single database and a single server when it was first implemented, the designers had the foresight to see that this design wouldn't be able to scale up and handle a large

¹⁵ <https://www.catchpoint.com/blog/how-dns-works>

¹⁶ <https://www.dnib.com/articles/the-domain-name-industry-brief-q4-2024>

¹⁷ <https://wgntv.com/business/press-releases/globenewswire/8789925/ns1s-global-dns-traffic-report-reveals-public-resolvers-dominate-the-internet/>

number of requests. They foresaw that trying to use a single system with a high volume of requests could cause the system to get overwhelmed, resulting in significant delays for tasks like web browsing or sending email. Their solution was to design DNS as a distributed system, giving each organization the ability to run their own DNS server to hold their DNS names and IP addresses. Or using the phone book analogy, each organization could create and manage their own section of the DNS phone book. But this distributed design also means that there will be thousands and thousands of DNS servers, or small phone books, scattered around all parts of the Internet. Using a distributed system, where the workload is spread out over thousands of DNS servers makes it possible to avoid the bottleneck and delays that result from using a single centralized server.

While using a single central server has its problems it also has one good feature. The great thing about using a system with one centralized server is that it's easy to tell all the clients the IP address of the server making configuration easy. The flipside of this is using a distributed system with thousands of DNS servers makes it much more difficult to tell all the clients which specific server out of the thousands of DNS servers holds the DNS records they want.

The DNS systems solves this problem by using three “levels” of servers, with the servers at each level narrowing down the path through all the DNS servers to the server that can resolve the DNS name to its IP address. The first level of servers are called DNS Root servers and they look at the Top Level Domain of the requested DNS name, the .com or .gov portion, and based on this return the IP address to the appropriate server in the next level. For example, if the request is for ebay.com the Root Server will return the IP address for the server in the next level that handles .com addresses. The servers in this second level are called Top Level Domain (TLD) servers.

The TLD servers use the next portion of the DNS name to decide which specific DNS server can resolve the DNS name to its IP address. For example, the .com TLD server will look up ebay.com in its table and return the IP address of the DNS server that knows how to resolve ebay.com to its IP address.

The DNS Server (with a capital S) is the third level of server in the DNS model, and its like the organizations portion of the phone book. That is, each DNS Server at this level holds the DNS records like the all-important A record, which associates the DNS name with its corresponding IP address. This is the

DNS Server that we really want to talk to, we just can't find it without talking to the DNS Root server and the TLD Server first.

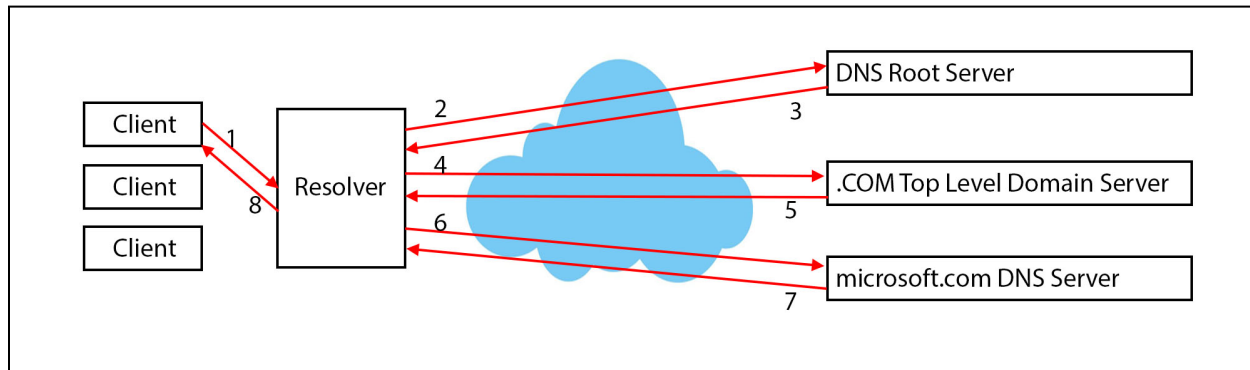
Here's a summary of the steps in the process for resolving a DNS name to an IP address:

1. A client needs the IP address associated with a DNS name, so it builds a DNS Request formatting the request with the rules set in the DNS protocol. The client sends the DNS request to the DNS Resolver for its network.
2. The DNS Resolver sends the request to a DNS Root Server.
3. The DNS Root server looks at the top level domain in the request, and returns the IP address of a TLD Server for this top level domain.
4. The DNS Resolver sends the request to the TLD Server.
5. The TLD Server looks up second part the domain name in its database and finds the IP address of the DNS Server that has the information for the DNS name we're looking for. The TLD Server returns the IP address of this DNS Server to the DNS Resolver.
6. The DNS Resolver sends the DNS request to the DNS Server.
7. The DNS Server sends the IP address associated with the DNS Name back to the DNS Resolver.
8. The DNS Resolver sends the IP address back to the client.

The Complete DNS Process

In our simple discussions of DNS we made it look like the client communicated directly with a DNS server, asking the server to resolve a DNS name to an IP address, but this simplified process isn't correct.

To help you understand how all the DNS components work together let's look at the process that's followed to resolve a name to an IP address. During this explanation we'll use the following diagram which shows the steps involved in the DNS process.



1. The DNS Client is any device or software that needs to translate a domain name into an IP address. For example, if you want to view a web page at msdn.microsoft.com the network stack on your computer will need to know the IP address of msdn.microsoft.com. The first step in this process consists of the Network Layer on your computer building a DNS Request using the DNS protocol and placing this DNS Request inside an IP packet. This IP packet will be sent to the DNS Resolver for your network. Remember that on a Windows computer the DNS Resolver(s) are referred to as the Primary DNS server and the Secondary DNS server. (To avoid confusion, we're going to call it by the correct technical name of DNS Resolver for the rest of this explanation.) The IP address of the DNS Resolver is one of the items that must be configured on every computer.

In the diagram this is labelled as Step 1, where the DNS Client sends the DNS Request to the DNS Resolver.

2. The DNS Resolver is the local part of the DNS system that has the responsibility for returning an IP address to the client. By local we mean that this computer is typically physically located on or near the client's network. The DNS Resolver handles the DNS requests for every device on an entire network. That is, the clients never ask the larger DNS system to resolve an IP address, they always ask the DNS Resolver to handle it for them. The DNS Resolver will handle the request, and once a response is received send the IP address back to the client.

When the DNS Resolver receives a new DNS request from a client the first thing it does is check its cache, which is a list of DNS names and IP addresses it's previously resolved. The Resolver builds the cache by storing the DNS names and IP addresses it receives any time it sends a DNS Request for a client and receives a response. If the DNS Resolver finds a copy in its cache, it returns the IP address to the client, which would be Step 8 in the diagram. This is done to speed up the entire process, because as you'll see, using a cached copy makes it possible to return the IP address using several fewer steps and sending several fewer network packets.

If the DNS Resolver doesn't have the requested data in its cache, it builds an IP packet with a DNS Request and sends it to one of the DNS Root Servers. Each DNS Resolver has a built-in list of IP addresses for the DNS Root Servers, so it knows the IP address of the DNS Root Server which it can use as the destination address in the IP packet. The following figure shows a list of DNS Root Servers.

HOSTNAME	IP ADDRESSES	OPERATOR
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	Verisign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California, Information Sciences Institute
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	Verisign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

List of DNS Root Servers from IANA.org.

It may look like there are only 13 Root Servers but there are actually dozens of Root Servers at each IP address. If there were only 13, they would easily be overwhelmed as they have to respond to hundreds of thousands of DNS Requests each minute. Each Root Server location has multiple servers and uses something called Round Robin DNS to balance the load of incoming requests between the servers. You'll learn about Round Robin DNS later, when you take a deep dive into the details of DNS.

The important thing to take away is that every DNS Resolver has this list of Root Servers and will build and send the packet shown as Step 2 in the diagram.

3. At this point in the request process, the DNS Root Server starts a process to determine which DNS server will ultimately handle the request. Each DNS Root Server has a list of the IP addresses for the Top Level Domain (TLD) Servers and will return the IP address of the appropriate TLD Server to the DNS Resolver. The Root Server determines the appropriate TLD server by looking at the last portion of any DNS name which will be .com, .blog, etc., or a country code. In our example, the client is looking for the IP address for msdn.microsoft.com, so the Root Server will return the IP address for the .com TLD server, which we'll say is: 192.5.6.30.

This IP address will be sent back to the DNS Resolver in the network packet shown as Step 3 in the diagram.

4. The DNS Resolver will receive this packet with the IP address for the appropriate TLD Server. In our example we now know the IP address of the TLD server for any DNS names ending in .com and the DNS Resolver will build another network packet using the .com TLD's IP address as the destination IP. This is Step 4 in the diagram.
5. Each TLD Server has a list of all the next level domains in its domain, along with their IP address. For example, the .com TLD servers will have lists of every DNS name that ends in .com, while the .edu TLD servers will have a list of every DNS name that ends in .edu.

In our example we're looking for msdn.microsoft.com, so the next portion of the DNS name is microsoft. When our example request gets to the .com TLD Server, the .com TLD server will find the IP address associated with microsoft. For our example let's assume that this IP address is: 20.81.111.85. The TLD server will build another response to send back to the DNS Resolver on our network, telling it that Microsoft's DNS Server's IP address is 20.81.111.85. This is shown as packet 5 in the diagram.

6. The DNS Resolver will receive this packet, and now knows the IP address of the DNS Server that holds the information it really wants. In the DNS ecosystem these servers are called

Authoritative servers because they have the authority to hand out IP addresses for any DNS names in their zone of authority, as opposed to the Root and TLD servers, which play a different role in the process. The DNS Resolver will build another network packet to send to this last authoritative DNS Server. In the case of our example the DNS Resolver will build a network packet to send to Microsoft's Authoritative DNS Server's at IP address 20.81.111.85. This is shown as step 6 in the diagram.

7. Each authoritative DNS Server has a list of DNS names and their corresponding IP addresses for the DNS names in its zone of authority. The zone of authority typically corresponds to a DNS name such microsoft.com. That is, any DNS name such as msdn.microsoft.com or xbox.microsoft.com would all fall into the microsoft.com zone of authority and be served by Microsoft's DNS server, while docs.google.com and sites.google.com will all fall under the google.com zone of authority and be served by Google's DNS server. However, sometimes DNS servers will cover multiple DNS names in their zone of authority. For example, if you register a DNS name for a web site you use with a web hosting service such as sites.google.com or GoDaddy, you won't run your own DNS server. Instead, you'll use one of Google's DNS Servers or GoDaddy's DNS Servers, and they will cover your respective zone of authority.

In the case of our example, the Microsoft Authoritative DNS Server will look up the IP address for msdn.microsoft.com and build a network packet to send it back to the DNS Resolver. For example, let's say the IP address for msdn.microsoft.com is 13.107.238.70. This will be placed in a DNS response packet and sent back to the DNS Resolver.

8. The DNS Resolver will receive the DNS response, put the IP address for msdn.microsoft.com in its cache, then forward the DNS response with msdn.microsoft.com's IP address back to the client. In addition to sending the IP address to the client, the DNS Resolver caches the IP addresses to speed up the process in case another client on its network makes a DNS request for the same DNS name. The amount of time an IP address remains in cache is set by the authoritative DNS server using a setting in the DNS response called the Time To Live or TTL. Typically, the TTL is set to ~3600 seconds, or 1 hour, but this can vary as each DNS administrator can change it.

9. The client will receive the DNS response from the DNS Resolver, and now have the IP address it needs to build the network packet it was originally working on.

This system of using multiple levels of DNS servers might seem a little complicated, but it was adopted for several important reasons including scalability, redundancy, localized control and access, and security.

Scalability - A centralized server model would be impractical for the size and complexity of the DNS system. With millions of domain names and billions of requests every day, a single server would not be able to handle the volume of traffic and would quickly become overwhelmed. By using a distributed system of servers, the load can be spread across multiple servers, making the system more scalable and resilient.

Redundancy- A distributed system of servers provides redundancy and fault tolerance. If a single server fails or is taken offline, the other servers in the system can still handle requests and provide the necessary information, and there are redundant servers at every level of the DNS eco system. This ensures that the DNS system remains available and functional even in the face of hardware failures or other disruptions.

Local Control - Because each network controls its own DNS authoritative server, they can change or add to the DNS names and addresses within their own domains at any time. If a centralized system were used instead, adding or changing names and IP addresses would require filling out forms and waiting for someone else to make the changes.

Localized Access – The DNS Resolvers are typically located close to the clients they serve. This means DNS queries can be resolved more quickly and efficiently, especially if the DNS name and IP address are already cached, reducing latency, and improving performance.

Security - A distributed system of servers can be more secure than a centralized server model. If a single server were used for all of DNS, and if it is compromised, it could potentially compromise every system connected to the Internet. By distributing the data and workload across multiple servers, the impact of a security breach can be minimized. Of course, the flip side of this is that each DNS Resolver and

authoritative server needs to be secured by someone with the experience to ensure the servers are properly secured.

The last thing to note about DNS is that the system performs amazingly well. Even though resolving a name to an IP address requires several network transactions, the entire process happens very quickly. In fact, in most cases the amount of time it takes to resolve a DNS to an IP address is measured in milliseconds, or thousandths of a second.

DHCP Basics

In this section you'll learn the details of DHCP including how the client can request, release or renew its IP information, the protocol used to exchange network configuration information between the client and server, and how to configure a DHCP server.

History

As networks grew in popularity one of the things that network administrators were looking for was a way to automatically configure the network settings for a computer or device. While manually configuring the network settings on a single device isn't difficult, it can be difficult to manually configure all the devices on a large network which may have hundreds or even thousands of connected devices. It can also be difficult to configure devices in dynamic networks, like in a coffee shop or college campus, where users want to connect wireless devices without the inconvenience of manually setting the IP address, netmask, default gateway/router IP, etc. And since necessity is the mother of invention, this led to the development of the DHCP protocol and service, which allow computers and devices to connect to a network and automatically receive their network settings. In this section you'll learn about the history and background of DHCP.

Dynamic Host Configuration Protocol (DHCP) is a protocol used to dynamically assign IP addresses and other network configuration parameters, such as subnet masks, default gateways, and DNS servers, to network devices. That is, DHCP uses the network to assign network settings to computers and other devices on the network. This may sound a little strange or counter intuitive, as it requires a device that isn't configured to use the network is somehow supposed to use the network to get its configuration information. But as you'll learn there is a process that allows this to happen. DHCP is an extremely

valuable tool as it allows network administrators to manage IP address allocation and network configuration easily, without having to manually assign IP addresses to each device.

DHCP evolved from a protocol known as the Bootstrap Protocol (BOOTP) which was introduced in the early 1980s during an effort by several UNIX vendors to provide something called diskless workstations. As the name implies, diskless workstations were computers that didn't have a local hard drive, but instead stored everything on a drive located on the network. The push behind this was that at the time hard drives were relatively expensive and networks were getting faster (although they were ridiculously slow by today's standards). Plus, it would make administration easier as both operating system software and files, and application programs, could be administered from a central location instead of having to install and manage the OS and applications on each separate computer. It turned out to be a good idea, but it was ahead of its time, as at the time networks weren't fast enough to make it practical. But even though the diskless workstations weren't adopted in their entirety, the protocol they used to boot, bootp, was found to be useful and eventually evolved into the DHCP we know and love today.

In 1993, the Internet Engineering Task Force (IETF) released the first version of DHCP as an extension to BOOTP, which added features such as automatic IP address allocation, lease management, and support for multiple vendor-specific options. The current version of DHCP, DHCPv6, was released in 2003 and provides support for IPv6 networks. The specifications for DHCP are documented in a series of IETF RFCs (Request for Comments), including RFC 2131 for DHCPv4 and RFC 3315 for DHCPv6. These documents outline the message formats, options, and procedures used by DHCP clients and servers to communicate and exchange configuration information.

DHCP has become a widely adopted protocol and is supported by most operating systems and network devices. DHCP servers are commonly deployed in enterprise networks, Internet service providers, and home networks to manage IP address allocation. DHCP is built into Windows, Linux, macOS, iPhones, Android phones, and almost every operating system or device that require network connectivity. It works so well and so transparently that most people have no idea that their devices are using DHCP.

Overview

Before we jump into the details, let's start with an overview of the DHCP process which will show the main components and terminology. Remember that DHCP (Dynamic Host Configuration Protocol) is a network protocol used to automate the process of assigning IP addresses and configuring network parameters for devices on a network. Its purpose is to simplify the management of IP addresses and network settings by dynamically allocating and renewing them as needed. DHCP eliminates the manual configuration of IP addresses, subnet masks, default gateways, DNS server addresses, and other network parameters on individual devices. By centralizing the IP address management, DHCP enables efficient utilization of available IP addresses and facilitates easy addition, removal, and movement of devices within a network.

Here are the main components of DHCP:

1. **DHCP Server:** A DHCP server is a network device or software application responsible for assigning IP addresses and other configuration parameters to DHCP clients on a network. It manages a pool of available IP addresses and leases them to clients upon request.
2. **DHCP Client:** A DHCP client is a device, such as a computer, smartphone, or network printer, that requests network configuration information from a DHCP server. It is responsible for starting the DHCP process when it needs to obtain an IP address, subnet mask, default gateway, and DNS server addresses.
3. **DHCP Relay Agent:** A DHCP relay agent is a network device that forwards DHCP messages between DHCP clients and DHCP servers that are on different subnets or networks. It relays the DHCP requests and responses to ensure communication between the clients and servers across network boundaries.

In the DHCP process a client obtains its network configuration information from a DHCP server using the following handshake process:

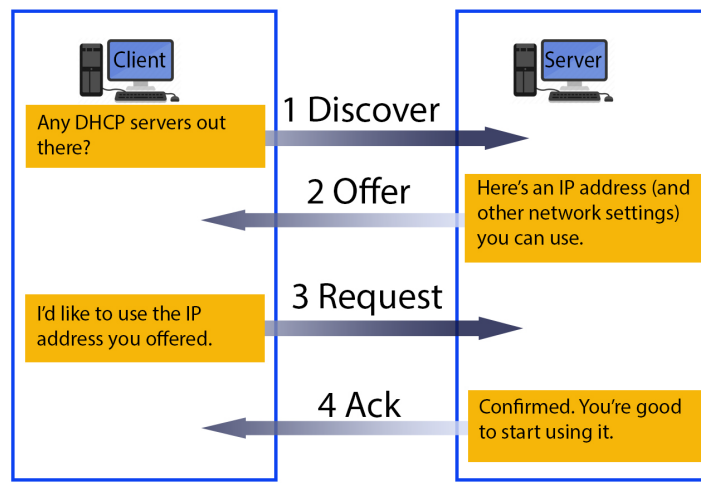


Figure XXX - The DHCP handshake.

1. **DHCP Discover Message:** The DHCP Discover message is a broadcast message sent by a DHCP client to discover available DHCP servers on the network. It is used to initiate the IP address assignment process. The Discover message contains information about the client's hardware address, network segment, and DHCP options it supports. The goal of this message is to find a DHCP server, so it's addressed as both an IP broadcast and a MAC broadcast.
2. **DHCP Offer Message:** When a DHCP server receives a DHCP Discover message, it looks through its pool of available IP addresses, and if one is available the DHCP server responds with a DHCP Offer message. The goal of this message is to propose a set of network settings to the client, so this message includes all the network configuration information including an IP address lease offer, subnet mask, default gateway, DNS server addresses, and any other configuration options. The client doesn't have an IP address yet, so this message is sent as an IP broadcast. But the DHCP server does know the client's MAC address, so the ethernet frame is addressed to the client's MAC address.
3. **DHCP Request Message:** After receiving a DHCP Offer message, the DHCP client sends a DHCP Request message. This message formally confirms the acceptance of the offered IP address and configuration parameters from the DHCP server. The client now knows the DHCP server's IP and

MAC addresses, so the IP packet and ethernet frame can be addressed directly to the DHCP server.

4. **DHCP Acknowledgment Message:** Upon receiving the DHCP Request message, the DHCP server marks the IP address as in use, so it won't be offered to any other devices. The formal name for this process is binding because the DHCP server binds the clients MAC address with the leased IP address. After the binding is complete the DHCP server sends a DHCP Acknowledgment message to the client. When the client receives this message, it knows the handshake is complete and it can start to use the IP address.

Lease, Renewal, and Rebinding Times

During the DHCP process there are several values exchanged that deal with lease times, the Lease Time, Lease Rebinding Time, and Lease Renewal Time. In this section you'll learn what the meaning of these values and how they're used. Let's start with the Lease Time. When a client leases an IP address from a DHCP server it doesn't get to use it forever, as the lease will expire at some point, which is specified by the Lease Time. This time is tracked by the DHCP server and starts when the server sends the DHCP Acknowledgement. If the client doesn't renew the address before the lease runs out, the server removes the binding from its database and returns the IP address to the pool of available addresses.

If a client wants to continue using a leased IP address, it needs to send a DHCP Renewal message to the DHCP server before the Lease Time expires. Renewing a lease is a simpler process than creating the original lease. And renewing a lease is preferable to losing the current IP address, stopping all network traffic on the client, and obtaining a new IP address. Plus, with a lease renewal the client can keep using the same IP address, as opposed to possibly getting a completely different IP address, which might happen if the DHCP lease process is started from the beginning. The amount of time the client waits before sending the DHCP Renewal message is specified by the Lease Renewal Time. The Lease Renewal Time should be less than the Lease Time, which will give the client some time to renew before the lease expires.

The Lease Rebinding Time is used by the client in case the DHCP server doesn't respond to a renewal message. For example, if the DHCP server that issued the original lease was replaced or taken offline,

the client won't be able to renew the lease. If the DHCP server doesn't respond and the Lease Rebinding Time is reached, the client will broadcast a Lease Rebinding message. This is sent as a broadcast so that any DHCP server will respond. The Rebinding message also includes the client's current IP address, in the hope that a new DHCP server can be found, and that the new DHCP server will be able to issue a new lease for the same IP address. The Lease Rebinding Time should be greater than the Lease Renewal Time, but less than the Lease Time.

If you're a network administrator and need to set the lease times, you need to take a couple of things into consideration to find the best times for a specific situation. The first is how static or dynamic will the devices connecting to the network be. That is, will devices be needing addresses for days or weeks at a time, like in a business office, or will the devices only need an address for an hour or a few hours, like in a coffee shop or airport, or will be something in between like in your home where some devices are permanent, but others may come and go. The second factor is how many IP addresses you have to work with and how many devices will need an address. In most cases you'll be working with non-routable IP addresses, so you'll typically be working with ~250 IP addresses from the 192.168.1.0 network. But, if you become a network admin, you may have situations where you have smaller or larger pools of addresses to work with.

Here are a couple of scenarios and suggestions for values for the lease times. In the first scenario assume you're setting up a wireless network in a location like a coffee shop where the expectation is that there will be devices connecting and disconnecting from the network frequently, with the average connection lasting an hour or less. The coffee shop has 15 tables and a few other places to sit and is licensed for a maximum occupancy of 100 people. In this case shorter times are probably better because most customers aren't going to stay for an extended period, and even if they do, they can renew an IP address with little interruption. You also want to reclaim unused IP addresses frequently, because when a customer leaves the shop they won't need the address, and you may need the IP address for the constant stream of new customers. That is, if you set the lease time to something like 24 hours, your DHCP server will most likely run out of addresses because it won't reclaim unused addresses quickly enough. For this scenario a good lease time would be an hour or 3600 seconds. For the renewal time and rebinding time, the general rule of thumb is that the renewal time should be 50% of the lease time, or 30 minutes (1800 seconds) in this case, and the rebinding time should be 87.5% of the lease time or

52.5 minutes (3150 seconds). Keep in mind that these percentages are just guidelines and not a strict rule, but it does give you some place to start when you're selecting times.

If you're setting up DHCP on a network in an office, or some location where the connected devices will remain fixed for weeks and months at a time, you can afford to make the lease time longer. For example, at the college we have several computer labs where the IP addresses and IP configuration settings for the computers in the labs are obtained through DHCP. In this case the lease time is set to 8 days or 691,200 seconds. This might seem like a long time, but the computers in these labs are rarely changed. The renewal times are 7.5 days or 561,600 seconds, and the rebinding time is set to 6 days and 22 hours or 684,000 seconds. The renewal and rebinding times aren't set to the typical 50% and 82.5%, but they still leave plenty of time for the devices to renew, or rebind if necessary.

The last scenario to think about is in your home where your home router is most likely acting as your DHCP server. This situation falls somewhere between the coffee shop and computer lab scenarios. That is, some of your home devices will be static, while others such as your laptops and phones may be connected and disconnected frequently. Plus, you may have friends and family members that drop by and want to connect their devices and need IP addresses on a more dynamic basis. But even if you allow your friends and family members to connect to your network you most likely won't be concerned about running out of addresses like you would be with the coffee shop scenario. Home wireless routers will come set with a default DHCP lease time, with most being set to either 24 hours or 72 hours (3 days) depending on the router brand and model.

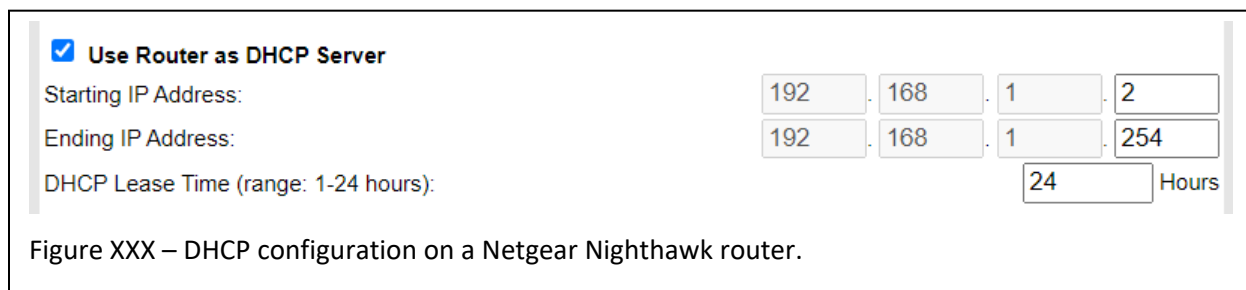
Router Brand	Default DHCP Lease Time
TP-Link	24 hours
Netgear	24 or 72 hours
Linksys	24 hours or 7 days
ASUS	24 hours
D-Link	24 or 72 hours

Table XXX – Typical DHCP Lease times for major home router manufacturers.

The amount of control you have over the DHCP lease times also varies by manufacturer and model. Some won't have any way to change the default values, some will let you change all three values, and

others will allow you to set the DHCP Lease Time but not the renewal and rebinding times. In the last case, the renewal and rebinding times will typically be set to the 50% and 82.5% of the lease time.

If you want to check or control the settings on your home router you'll have to login to the router as the administrator, and then find the configuration settings. The process for logging in as administrator and finding the DHCP settings varies by router brand and model, so you'll have to search the Internet for instructions for your router. Note that this is just something you can do if you're curious. I don't suggest mucking around with these values, or any of the settings on your home router, unless you're having specific issues that you're trying to correct. Your home router is definitely a case where the saying of "If it ain't broke, don't fix it" applies.



☒ **Use Router as DHCP Server**

Starting IP Address: 192 . 168 . 1 . 2

Ending IP Address: 192 . 168 . 1 . 254

DHCP Lease Time (range: 1-24 hours): 24 Hours

Figure XXX – DHCP configuration on a Netgear Nighthawk router.

The following figure shows the DHCP configuration settings for a Netgear Nighthawk router. As you can see, the only configurable setting is for the DHCP Lease Time, and the default value is 24 hours.

You can also use Wireshark to view the DHCP settings by starting Wireshark, setting the filter to only view DHCP packets, then connecting a new device to your home network. You can then drill down into the DHCP Offer and DHCP Acknowledge messages to see the lease, renewal, and rebinding values.

DHCP Process with DHCP Relay

The "normal" DHCP process works if the client and server are on the same network segment, but it will fail if they're on different segments with a router between them. It will fail because most of the process relies on ethernet broadcasts, and routers won't pass ethernet broadcasts from one network segment to another. If you want to use DHCP in this situation, it can still be done by using something called a DHCP Relay. The DHCP relay is a device and service that are used to forward DHCP messages between

clients and servers on different networks, ensuring that the clients can still obtain IP addresses and configuration information.

The DHCP Relay is typically a router that has NICs connected to two different network segments. One of the network segments will have a DHCP server and the other will not. The DHCP Relay will route the DHCP Discover message from the network segment without the DHCP server to the network segment where the DHCP server lives. The DHCP Relay knows to do this routing because it's been loaded with a program that's been configured to pass the DHCP Discover messages between network segments. The DHCP Relay will also handle the returning the DHCP offer to the DHCP client.

Here's what the DHCP process looks like when a DHCP Relay is involved. The packets sent in the handshake are pretty much the same. The big difference is that the DHCP relay sits in the middle of the handshake, moving the packets back and forth between the two network segments.

1. **DHCP Discover:** The process starts the same as the DHCP process without a DHCP Relay, because the client will have no idea if a DHCP Relay is involved or not. This means that the client will create a DHCP Discover message which will be broadcast on the network segment.
2. **DHCP Relay forwards DHCP Discover:** The DHCP Relay will see the ethernet broadcast and pass the DHCP Discover message up the network stack to the DHCP Relay application code. This code will note the MAC address of the original message, then create another message to send to the DHCP server. This message will be sent out as another ethernet broadcast on the NIC connected to the "other" network segment, the network segment with the DHCP server.
3. **DHCP Offer:** The DHCP server receives the DHCP Discover packet and does its thing, finding an available IP address and sending the DHCP Offer.
4. **DHCP Relay forwards DHCP Offer:** The DHCP Relay receives the DHCP Offer. It then looks in its list to find the MAC address of the device that started the DHCP process, and forwards the DHCP offer to this device.

5. **DHCP Request:** The client receives the forwarded DHCP Offer and builds a DHCP Request saying it would like to use the IP address. To the client, it looks like the DHCP Offer came from the DHCP Relay, so it addresses the DHCP Request to the relay agent.
6. **DHCP Relay forwards DHCP Request:** The DHCP Relay receives the DHCP Request and forwards it to the DHCP server.
7. **DHCP Acknowledgement:** The DHCP Server receives the DHCP Request, performs the binding, builds the DHCP Acknowledgement, and sends the packet to the DHCP Relay.
8. **DHCP Relay forwards DHCP Acknowledgement:** The DHCP Relay receives the DHCP Acknowledgement and forwards it to the client.
9. **Client receives DHCP Acknowledgement:** The client receives the DHCP Acknowledgement, begins to use the network, and everyone lives happily ever after.

Labsim Section 4 Notes

To complete the Labsim homework for this section do the following labs:

2.6.13

4.2.9

4.3.7 – Remember that you need to r-click on any device to open up Windows and run any commands.

4.4.5

4.4.6

4.6.4

4.6.6

4.6.7

4.6.8

6.2.10 – Do this lab from section 6.2 before doing the other labs. This lab has to do with client configuration and is relatively easy.

6.2.5 – This lab, and all the remaining labs in section 6.2 require knowledge of setting up a DHCP server. You'll need to read and watch the material in 6.2.1 – 6.2.4 to gain this knowledge and complete these labs. While it's not crucial that you know how to configure a DHCP server, both the material and the labs should provide you with greater clarity on what DHCP is and how it works.

6.2.6

6.2.7

6.2.8

6.3.4 – This lab, and the next lab deal with APIPA. I haven't seen APIPA used at all in the real world, but it is something you should know about. This lab also has you deal with a DHCP server which is running in a Virtual Machine. If you haven't dealt with VMs before this can seem a little strange, but if you follow the lab instructions you should be able complete the steps. The main thing is to try and not get lost in the commands, and try to understand what's going on with DHCP and APIPA.

6.3.5

6.5.11

The following labs have you work on a DNS server, adding records, and troubleshooting records. These labs can be a little difficult because you need to start a VM, then use Microsoft's interface to edit the DNS records. If you start the lab, then end it, and score it, you can see the steps for starting the VM and accessing the DNS server. The important concepts for you to take away from these labs are what a zone is, what an A record contains, and what a CNAME record contains.

6.5.12

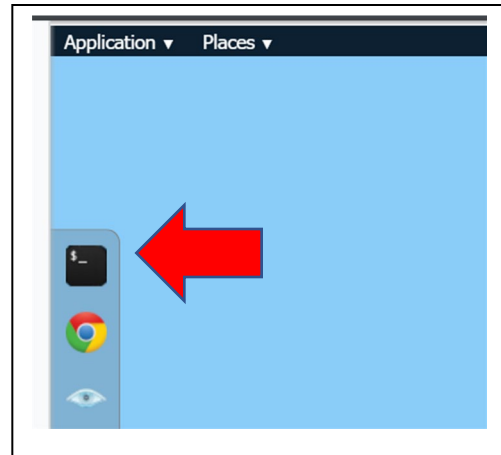
6.5.13

6.5.14

6.5.15

6.6.5

6.6.6 – This lab is done in Linux, but the nslookup command is the same as it is in Windows. To run the command(s), start the lab and then click the **terminal** app.



Extra Credit – The following labs are optional and will be counted as extra credit. You're not expected to know how to complete these labs, and Labsim doesn't even provide much instruction in their videos and notes. But they provide good practice in the real world problem of figuring out how to configure devices you haven't worked with before. And Labsim will almost always tell you the exact steps you need to perform if you start the lab and then tell Labsim you're done.

4.2.10 – This lab requires knowing which buttons to push to configure the networking on an iPad. If you're like me and don't know what to do, you can start the lab and try to figure it out. If you can't find the way to change the settings, you can end the lab and Labsim will tell you where to go and what buttons to push. Or, you can always ask Dr. Internet how to configure the network settings on an iPad.

4.2.11 – This lab requires some knowledge of Linux to configure the network settings. If you're not already familiar with Linux most of this lab will seem like gibberish as you'll be typing commands to change directories and edit a file. If you exit the lab, Labsim will tell you exactly what to type, but at that point this because more of a typing exercise than a networking exercise.

4.4.6 - This is another Linux Lab. To answer the first question, you won't run any Linux commands, you'll need to click on the Exhibits button at the top right of the Lab Window.

4.5.9 – This uses IPv6, which can be good to know, but is a little more complicated than IPv4. While it's good to know about IPv6, the use of non-routable IP addresses has extended the life of IPv4. And, most of networking is still built around IPv4, so it would be better to learn as much as you can about IPv4 first.

Once you have a firm grasp of how IPv4 works you'll be able to quickly pick up IPv6. I suggest looking at everything in Labsim Section 4.5 if you want to do and understand this lab.

4.6.5 – More Linux!

6.4.4 -6.4.10 – These labs have to do with setting up a DHCP Relay. While this isn't hard in concept, to complete you need to learn the commands and processes to accomplish this. And once again, the steps are complicated enough that completing them may add confusion and make it more difficult to understand the concept.

Ways to Check Your Comprehension

The material in this section will be included in Test 2 in Labsim, which isn't due for a few weeks. If you want a couple of other ways to check your comprehension before Test 2 you can use the Practice Questions in Labsim, which you have to complete as part of your homework, or you can use the Practice Test Questions that I've set up for you.

The Practice Test Questions are completely optional, but they will provide several more hands-on labs and questions that will be similar to those on the actual test. This is a great way to get more hands-on experience, as well as a good way to check your comprehension and prepare for the real test. The Practice Test is completely optional, and you can take it as few or as many times as you want. If you do decide to take the Practice Test you should note that it may have questions over items that were not assigned in class or questions that have to do with Linux systems.