Understanding IP Addresses, Subnets, and CIDR Notation for Networking

by Justin Ellingwood

Introduction

Understanding networking is a fundamental part of configuring complex environments on the internet. This has implications when trying to communicate between servers efficiently, developing secure network policies, and keeping your nodes organized.

In a previous guide, we went over some basic networking terminology. You should look through that guide to make sure you are familiar with the concepts presented there.

In this article, we will discuss some more specific concepts that are involved with designing or interacting with networked computers. Specifically, we will be covering network classes, subnets, and CIDR notation for grouping IP addresses.

Understanding IP addresses

Every location or device on a network must be addressable. This means that it can be reached by referencing its designation under a predefined system of addresses. In the normal TCP/IP model of network layering, this is handled on a few different layers, but usually when we refer to an address on a network we are talking about an IP address.

IP addresses allow network resources to be reached through a network interface. If one computer wants to communicate with another computer, it can address the information to the remote computer's IP address. Assuming that the two computers are on the same network, or that the different computers and devices in between can translate requests across networks, the computers should be able to reach each other and send information.

Each IP address must be unique on its own network. Networks can be isolated from one another, and they can be bridged and translated to provide access between distinct networks. A system called Network Address Translation, allows the addresses to be rewritten when packets traverse network borders to allow them to continue on to their correct destination. This allows the same IP address to be used on multiple, isolated networks while still allowing these to communicate with each other if configured correctly.

The difference between IPv4 and IPv6

There are two revisions of the IP protocol that are widely implemented on systems today: IPv4 and IPv6. IPv6 is slowly replacing IPv4 due to improvements in the protocol and the limitations of IPv4 address space. Simply put, the world now has too many internet-connected devices for the amount of addresses available through IPv4.

IPv4 addresses are 32-bit addresses. Each byte, or 8-bit segment of the address, is divided by a period and typically expressed as a number **0–255**. Even though these numbers are typically expressed in decimal to aid in human comprehension, each segment is usually referred to as an octet to express the fact that it is a representation of 8 bits.

A typical IPv4 address looks something like this:
192.168.0.5

The lowest value in each octet is a **0**, and the highest value is **255**.

We can also express this in binary to get a better idea of how the four octets will look. We will separate each 4 bits by a space for readability and replace the dots with dashes:
1100 0000 - 1010 1000 - 0000 0000 - 0000 0101

Recognizing that these two formats represent the same number will be important for understanding concepts later on.

Although there are some other differences in the protocol and background functionality of IPv4 and IPv6, the most noticeable difference is the address space. IPv6 expresses addresses as an 128-bit number. To put that into perspective, this means that IPv6 has space for more than 7.9×10^28 times the amount of addresses as IPv4.

To express this extended address range, IPv6 is generally written out as eight segments of four hexadecimal digits. Hexadecimal numbers represent the numbers 0–15 by using the digits 0–9, as well as the numbers a–f to express the higher values. A typical IPv6 address might look something like this:
1203:8fe0:fe80:b897:8990:8a7c:99bf:323d

You may also see these addresses written in a compact format. The rules of IPv6 allow you to remove any leading zeros from each octet, and to replace a single range of zeroed groups with a double colon (::).

For instance, if you have one group in an IPv6 address that looks like this:
...:00bc:...

You could instead just type:
...:bc:...

To demonstrate the second case, if you have a range in an IPv6 address with multiple groups as zeroes, like this:
...:18bc:0000:0000:0000:00ff:...

You could compact this like so (also removing the leading zeros of the group like we did above):
...:18bc::ff...

You can do this only once per address, or else the full address will be unable to be reconstructed.

While IPv6 is becoming more common every day, in this guide, we will be exploring the remaining concepts using IPv4 addresses because it is easier to discuss with a smaller address space.

IPv4 Addresses Classes and Reserved Ranges

IP addresses are typically made of two separate components. The first part of the address is used to identify the network that the address is a part of. The part that comes afterwards is used to specify a specific host within that network.

Where the network specification ends and the host specification begins depends on how the network is configured. We will discuss this more thoroughly momentarily.

IPv4 addresses were traditionally divided into five different "classes", named A through E, meant to differentiate segments of the available addressable IPv4 space. These are defined by the first four bits of each address. You can identify what class an IP address belongs to by looking at these bits.

Here is a translation table that defines the addresses based on their leading bits:

Class A
0--- : If the first bit of an IPv4 address is "0", this means that the address is part of class A. This means that any address from **0.0.0.0 to 127.255.255.255** is in class A.

Class B
10-- : Class B includes any address from **128.0.0.0 to 191.255.255.255**. This represents the addresses that have a "1" for their first bit, but don't have a "1" for their second bit.

Class C
110- : Class C is defined as the addresses ranging from **192.0.0.0 to 223.255.255.255**. This represents all of the addresses with a "1" for their first two bits, but without a "1" for their third bit.

Class D
1110 : This class includes addresses that have "111" as their first three bits, but a "0" for the next bit. This address range includes addresses from **224.0.0.0 to 239.255.255.255**.

Class E
1111 : This class defines addresses between **240.0.0.0 and 255.255.255.255**. Any address that begins with four "1" bits is included in this class.

Class D addresses are reserved for multi-casting protocols, which allow a packet to be sent to a group of hosts in one movement. Class E addresses are reserved for future and experimental use, and are largely not used.

Traditionally, each of the regular classes (A–C) divided the networking and host portions of the address differently to accommodate different sized networks. Class A addresses used the remainder of the first octet to represent the network and the rest of the address to define hosts. This was good for defining a few networks with a lot of hosts each.

The class B addresses used the first two octets (the remainder of the first, and the entire second) to define the network and the rest to define the hosts on each network. The class C addresses used the first three octets to define the network and the last octet to define hosts within that network.

The division of large portions of IP space into classes is now almost a legacy concept. Originally, this was implemented as a stop-gap for the problem of rapidly depleting IPv4 addresses (you can have multiple computers with the same host if they are in separate networks). This was replaced largely by later schemes that we will discuss below.

Reserved Private Ranges

There are also some portions of the IPv4 space that are reserved for specific uses.

One of the most useful reserved ranges is the loopback range specified by addresses from **127.0.0.0 to 127.255.255.255**. This range is used by each host to test networking to itself. Typically, this is expressed by the first address in this range: **127.0.0.1**.

Each of the normal classes also have a range within them that is used to designate private network addresses. For instance, for class A addresses, the addresses from **10.0.0.0 to 10.255.255.255** are reserved for private network assignment. For class B, this range is **172.16.0.0 to 172.31.255.255**. For class C, the range of **192.168.0.0 to 192.168.255.255** is reserved for private usage.

Any computer that is not hooked up to the internet directly (any computer that goes through a router or other NAT system) can use these addresses at will.
There are additional address ranges reserved for specific use-cases. You can find a summary of reserved addresses here.
https://en.wikipedia.org/wiki/Reserved_IP_addresses

Netmasks and Subnets
The process of dividing a network into smaller network sections is called subnetting. This can be useful for many different purposes and helps isolate groups of hosts from each other to deal with them more easily.

As we discussed above, each address space is divided into a network portion and a host portion. The amount of the address that each of these take up is dependent on the class that the address belongs to. For instance, for class C addresses, the first 3 octets are used to describe the network. For the address **192.168.0.15**, the **192.168.0** portion describes the network and the **15** describes the host.

By default, each network has only one subnet, which contains all of the host addresses defined within. A netmask is basically a specification of the amount of address bits that are used for the network portion. A subnet mask is another netmask within used to further divide the network.

Each bit of the address that is considered significant for describing the network should be represented as a “1” in the netmask.

For instance, the address we discussed above, **192.168.0.15** can be expressed like this, in binary:
1100 0000 - 1010 1000 - 0000 0000 - 0000 1111

As we described above, the network portion for class C addresses is the first 3 octets, or the first 24 bits. Since these are the significant bits that we want to preserve, the netmask would be:
1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

This can be written in the normal IPv4 format as **255.255.255.0**. Any bit that is a “0” in the binary representation of the netmask is considered part of the host portion of the address and can be variable. The bits that are “1” are static, however, for the network or subnetwork that is being discussed.

We determine the network portion of the address by applying a bitwise AND operation to between the address and the netmask. A bitwise AND operation will save the networking portion of the address and discard the host portion. The result of this on our above example that represents our network is:
1100 0000 - 1010 1000 - 0000 0000 - 0000 0000

This can be expressed as **192.168.0.0**. The host specification is then the difference between these original value and the host portion. In our case, the host is **0000 1111** or **15**.

The idea of subnetting is to take a portion of the host space of an address, and use it as an additional networking specification to divide the address space again.

For instance, a netmask of **255.255.255.0** as we saw above leaves us with 254 hosts in the network (you cannot end in 0 or 255 because these are reserved). If we wanted to divide this into two subnetworks, we could use one bit of the conventional host portion of the address as the subnet mask.

So, continuing with our example, the networking portion is:
1100 0000 - 1010 1000 - 0000 0000

The host portion is:
0000 1111

We can use the first bit of our host to designate a subnetwork. We can do this by adjusting the subnet mask from this:
1111 1111 - 1111 1111 - 1111 1111 - 0000 0000

To this:
1111 1111 - 1111 1111 - 1111 1111 - 1000 0000

In traditional IPv4 notation, this would be expressed as **192.168.0.128**. What we have done here is to designate the first bit of the last octet as significant in addressing the network. This effectively produces two subnetworks. The first subnetwork is from **192.168.0.1 to 192.168.0.127**. The second subnetwork contains the hosts **192.168.0.129 to 192.168.0.255**. Traditionally, the subnet itself must not be used as an address.

If we use more bits out of the host space for networking, we can get more and more subnetworks.

CIDR Notation

A system called Classless Inter-Domain Routing, or CIDR, was developed as an alternative to traditional subnetting. The idea is that you can add a specification in the IP address itself as to the number of significant bits that make up the routing or networking portion.

For example, we could express the idea that the IP address **192.168.0.15** is associated with the netmask **255.255.255.0** by using the CIDR notation of **192.168.0.15/24**. This means that the first 24 bits of the IP address given are considered significant for the network routing.

This allows us some interesting possibilities. We can use these to reference "supernets". In this case, we mean a more inclusive address range that is not possible with a traditional subnet mask. For instance, in a class C network, like above, we could not combine the addresses from the networks 1**92.168.0.0** and **192.168.1.0** because the netmask for class C addresses is **255.255.255.0**.

However, using CIDR notation, we can combine these blocks by referencing this chunk as **192.168.0.0/23**. This specifies that there are 23 bits used for the network portion that we are referring to.

So the first network (**192.168.0.0**) could be represented like this in binary:
1100 0000 - 1010 1000 - 0000 0000 - 0000 0000

While the second network (**192.168.1.0**) would be like this:
1100 0000 - 1010 1000 - 0000 0001 - 0000 0000

The CIDR address we specified indicates that the first 23 bits are used for the network block we are referencing. This is equivalent to a netmask of **255.255.254.0**, or:
1111 1111 - 1111 1111 - 1111 1110 - 0000 0000

As you can see, with this block the 24th bit can be either 0 or 1 and it will still match, because the network block only cares about the first 23 digits.

CIDR allows us more control over addressing continuous blocks of IP addresses. This is much more useful than the subnetting we talked about originally.

Conclusion

Hopefully by now, you should have a working understanding of some of the networking implications of the IP protocol. While dealing with this type of networking is not always intuitive, and may be difficult to work with at times, it is important to understand what is going on in order to configure your software and components correctly.

There are various calculators and tools online that will help you understand some of these concepts and get the correct addresses and ranges that you need by typing in certain information. ***https://CIDR.xyz*** provides a translation from decimal-based IP addresses to octets, and lets you visualize different CIDR netmasks.