A gentle guide to subnetting

Part 1: Understanding Binary

Introduction

Here we have it, folks. The guide you've all been waiting for! The guide on subnetting! This is a three-part guide with the first part giving you a brief introduction to binary and finishing with some basic easy subnetting. The second guide will be more advanced subnetting and VLSM. So take a deep breath and let's get to it!

People are always advertising fancy tricks, tips, magic, etc. to make subnetting "easier". The fact is, subnetting is only as difficult as you make it. The best way to learn subnetting is just to do it! Before long, it will become second nature.

Let's start with binary. As you know, the entire internet communicates with electrical, light, or radio signals. These signals are nothing more than a series of waves, pulses, or lights (depending on the medium) that indicate either "on" or "off". For instance, if the medium is fiber optics, these are essentially blinking lights. They are blinking MUCH faster than we can percieve, but they are blinking. Each time the light is "on", it signifies a 1 and when the light is "off", it signifies a "0". Each 1 or 0 is called a "bit". These bits stream together to form binary code. "Binary" stands for "2", which represents the 2 possible digits, 0 and 1. These signals are grouped together in predetermined lengths to indicate when the message stops and starts.

======|11000001010100000000000000001|========
Fiber Fiber

An IP address is 4 Bytes (4B) or 4 blocks of 8 bits (8b). Keep in mind that when abbreviating bits and Bytes, a Capital "B" signifies "Bytes" and a lowercase "b" signifies "bits." I will capitalize Bytes throughout this guide to remind you of that fact. An example of the IP address 192.168.0.1 in bit form is:

11000000.10101000.00000000.00000001

Each Byte in an IP address is also called an "octet". The "oct" in octet refers to 8. Think octagon, octave, etc.

Still with me? Good! Think of this as a much more advanced Morse Code, where the dots are 0s and the dashes are 1s.

Creating an IP address out of bits

Alright, now you know what bits and Bytes are, let's discuss how to put them together to get larger numbers out of a series of 0s and 1s. In each Byte, each of the 8 bits represents a placeholder for a power of 2. It starts on the left, in the 8^{th} position, at 2^7 and continues to the right to the last bit which is 2^0. Remember, everything in networking starts with "0". So, the full byte looks like this:

Byte							
1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

In this table, you can see that each bit represents a power of 2. So the first bit on the left is 7 2s multipled together: 2*2*2*2*2*2*2 = 128 Or 2^7. Here is an example of all bits being "on" or set to 1 to equal 255:

255								
1	1	1	1	1	1	1	1	11111111
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$
128	64	32	16	8	4	2	1	128 + 64 + 32 + 16 + 8 +4 + 2 + 1 = 255

As you can see, we add all of the values together to form 255. That is the highest number that can be used in an IPv4 address. Here is an example of all bits turned "off" to form 0:

0								
0	0	0	0	0	0	0	0	00000000
~~2^7~~	~~2^6~~	~~2^5~~	~~2^4~~	~~2^3~~	~~2^2~~	~~2^1~~	~~2^0~~	0
~~128~~	~~64~~	~~32~~	~~16~~	~~8~~	~~4~~	~~2~~	~~1~~	0

Since every bit is a 0, we don't add any numbers and just leave it at 0.

Now that we've done that, let's break the IP address 192.168.0.15 into binary!

First, we have the 192. Only the first 2 bits are on, which means we add 2^7(128) and 2^6(64) which gives us 192.

192								
1	1	0	0	0	0	0	0	10101000
2^7	2^6	~~2^5~~	~~2^4~~	~~2^3~~	~~2^2~~	~~2^1~~	~~2^0~~	$2^7 + 2^6$
128	64	~~32~~	~~16~~	~~8~~	~~4~~	~~2~~	~~1~~	128 + 64 = 192

168 has a few more bits that are on. For 168, we add $2^7 + 2^5 + 2^3$.

168								
1	0	1	0	1	0	0	0	10101000
2^7	~~2^6~~	2^5	~~2^4~~	2^3	~~2^2~~	~~2^1~~	~~2^0~~	$2^7 + 2^5 + 2^3$
128	~~64~~	32	~~16~~	8	~~4~~	~~2~~	~~1~~	128 + 32 + 8 = 168

Here is our 0, which has all bits set to 0, or off. There is nothing to add here.

0								
0	0	0	0	0	0	0	0	00000000
~~2^7~~	~~2^6~~	~~2^5~~	~~2^4~~	~~2^3~~	~~2^2~~	~~2^1~~	~~2^0~~	0
~~128~~	~~64~~	~~32~~	~~16~~	~~8~~	~~4~~	~~2~~	~~1~~	0

And finally, our 15. For 15, we need to add $2^3 + 2^2 + 2^1 + 2^0$ like so:

15								
0	0	0	0	1	1	1	1	00001111
~~2^7~~	~~2^6~~	~~2^5~~	~~2^4~~	2^3	2^2	2^1	2^0	$2^1 + 2^0$
~~128~~	~~64~~	~~32~~	~~16~~	8	4	2	1	8 + 4 + 2 + 1 = 15

These four bytes of 0s and 1s come together to form the 4 octets that make up our IP address:

192	168	0	15
11000000	10101000	00000000	00001111

Try a few of your own until you get the hang of it! You will definitely want to be able to do this easily for the next section. There are binary calculators available as well as charts, but try your best to do this without referencing these things as it will help you in the long run. the guide.)

192.168.1.12
10.1.0.254
172.16.127.5
128.254.64.32

Conclusion

It might look difficult at first, but with lots of practice, binary will make perfect sense! Join me in my next guide and we will start subnetting!

IP Conversion Solutions

192.168.1.12
11000000.10101000.00000001.00001100

10.1.0.254
00001010.00000001.0000000.11111110

172.16.127.5
10101100.00010000.01111111.00000101

128.255.64.32
10000000.11111111.01000000.00100000

A gentle guide to subnetting

Part 2: Networks and basic subnetting

What is a Network?

I'm sure you have all heard of networks and understand that they exist and that they are comprised of your, or your company's, devices. You may have also noticed that they typically all have the same IP addressing scheme, at least the first numbers. Most smaller networks (less than 254 hosts; there's a reason for this that I'll explain later) only change the last octet of their IP address. For instance, in your home network, you may have 192.168.1.1 as your router's IP. Your computer might be 192.168.1.2 and your phone might be 192.168.1.99. If you are at home now, go ahead and find your IP addresses on your devices, or login to your router to obtain them from there. Unfortunately, I won't be able to assist you with this as I don't know what devices you have, but look around online to see if you can figure out how to find them.

As you can see by the above example, each IP address has 3 octets that match: 192.168.1 and one that doesn't: 1, 2, and 99 respectively in this example. The first 3 octets are known as the "network" portion of the address and the last octet is known as the "host" portion:

192	168	1	99
Network	Network	Network	Host

Remember that an octet can only be a number from 0 – 255, so that leaves 256 possibilities for that last number. I mentioned earlier that there are only 254 possibilities for a host, so what happened to the other 2 addresses?! The addresses 192.168.1.0 and 192.168.1.255 are reserved for the network. The first address of any network, in this case 192.168.1.0, is reserved for the "network address". This address is used to identify your network for other routers or devices to use. If you need to setup a VPN to your company, that company's network will need to know how to get to yours, so it will use 192.168.1.0 as the address to route to. The last address of the network, 192.168.1.255, is the "broadcast" address. This address is used to send "broadcasts" to every host on the network. For instance, if a computer on a network doesn't know the IP address of a computer named "Bob", it might send a message to 192.168.1.255 in order to "broadcast" the message to every other device on the network asking "what IP address does "Bob" have?" If the router on the network knows "Bob", it will

reply with Bob's address. These concepts are outside the scope of this document, but the primary takeaway you need to know is that the first and last IP addresses of a network are reserved and cannot be given to hosts. So you will almost always subtract 2 from the available hosts on your network.

Classful Networking

We are finally starting to get closer to full subnetting! Now that we understand the network and host portion of a network, let's discuss how we can determine how many hosts we can have in a particular network. In our previous example, we used the network 192.168.1.0 which has 1 network and a possibility of 254 hosts. How do we determine this for other networks? In the past, there was a concept of "network classes". These classes made it very easy to tell how many hosts and networks could be created from one address. The primary Network classes are as follows:

Class A: 0.0.0.0 – 127.255.255.255

Class B: 128.0.0.0 – 191.255.255.255

Class C: 192.0.0.0 – 223.255.255.255

But what about the addresses from 224.0.0.0 and above? These are reserved for special cases and are beyond the scope of this guide.

The purpose of these "classes" was to make it easier to determine what address to use based on how many hosts and networks you require. Class A networks can support up to 16 MILLION hosts, so that would be used for very large networks. Class B networks can support about 65,000, and a Class C (which most of you currently use at home) can support 254 hosts. This was a very easy way of doing things, but as networking became more complicated, so did the requirements for creating networks. This led to "Classless Inter-Domain Routing" or CIDR. With CIDR, the range the IP address falls into doesn't matter anymore. Any IP address can be in any "class".

Subnetworks

So how do we determine how many networks and hosts are available now? And why do we care? Let's say you have three departments in your organization. These three departments need to be separated from eachother in a way that allows you to control who has access to each department's respective resources. The easiest way is to put them on different networks that can't talk to eachother by design (unless you allow it.) These networks are called "subnetworks" or "subnets" for short.

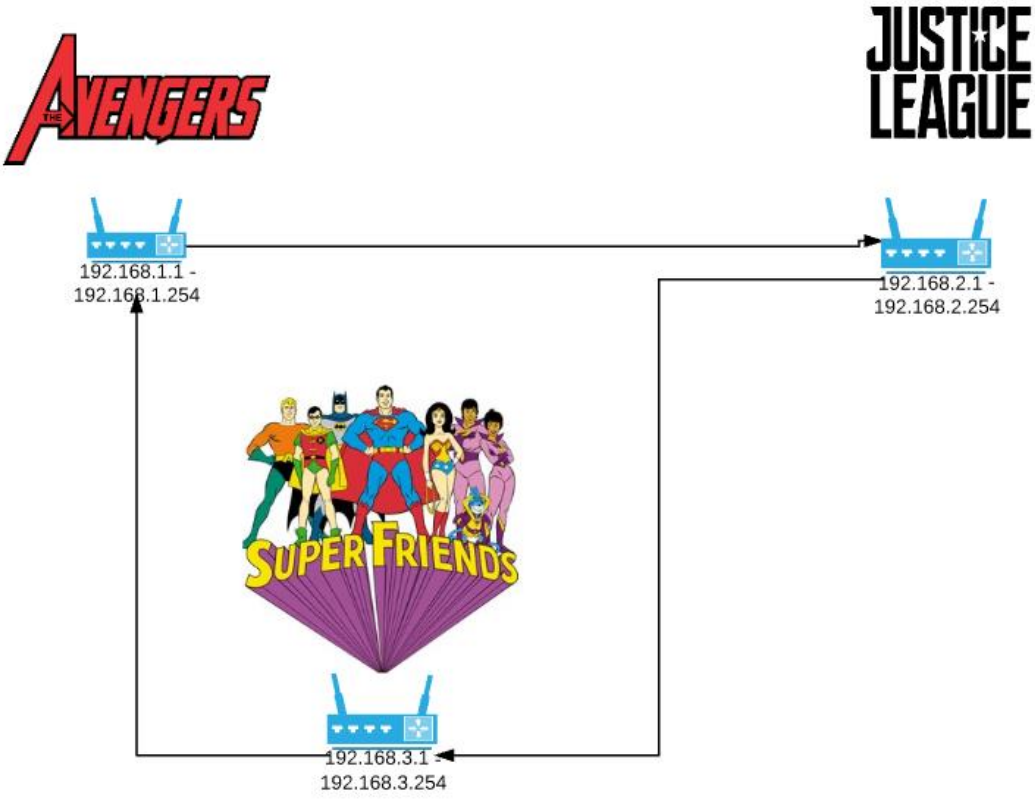
A subnet allows us to divide networks into smaller networks to more efficiently allocate IP addresses to logical business units. Ok, that explanation might have been a little long, so let's use an example:

Most guides would use Finance, Management, and Marketing, but I think we can do something more entertaining than that. Let's use The Avengers, The Superfriends, and The Justice League.

Let's say the three departments all use a 192.168.1.0 network. This makes things easy, but they can only have 254 hosts! So what happens if The Avengers adds 150 employees and The Superfriends adds 150? That isn't going to work! So that's one reason why this network won't work. Assuming all three

organizations don't add too many hosts, the next issue is the ability to access eachother's resources. If they are all on the same network, it is easy for Clark Kent in the Justice League to be able to access a computer that Tony Stark at The Avengers office wants to keep secure! You may also want to ensure The Superfriends aren't able to access Youtube while still allowing The Justice League to look up all the cat videos they want. These are all reasons to "subnet" your network!

Now, let's say The Avengers gets addresses 192.168.1.1 – 192.168.1.254, the Justice League gets 192.168.2.1 – 192.168.2.254, and The Superfriends gets 192.168.3.1 – 192.168.3.254. That gives all three of them 254 addresses to use and puts them on different networks (Remember, the network in this case is the first 3 octets.) This allows for security, scalability, and administration!



So how do we know how large this network is? Why can't this network just start at 192.168.0.0 – 192.168.255.255 like a class B network? This is where "Classless Inter-Domain Routing" or "CIDR notation" comes into play!

If you have been in IT for any amount of time, you have probably seen the "/" slash following an IP address with a number after it. Such as: 192.168.1.1/24. This /24 shows us that 24 bits are dedicated to the network. Remember bits and Bytes? We are back to those again! So, if 24 bits (3 Bytes) are dedicated to the network, we get 8 bits (or 1 Byte) dedicated for hosts. If we have 8 bits to use for hosts,

that means in the last octet, all of the bits are set to 1. So, if we have 8 bits set to 1, that is 255. Here is the table for a refresher:

255								
1	1	1	1	1	1	1	1	11111111
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$
128	64	32	16	8	4	2	1	128 + 64 + 32 + 16 + 8 +4 + 2 + 1 = 255

In an IP address, you have a portion dedicated to the network and a portion of the address dedicated to the hosts. In a /24 network, that means the first 3 octets are for the network. In this example, those three octets are 192.168.1. If you write that out in binary, that is 24 bits (3 * 8). If the subnet mask is 24, that means the first 3 octets can't change, but the last octet (the 8 that are left) can!

192.168.1.0 is the following in binary:
11000000.10101000.00000001.00000000

A /24 subnet mask in binary looks like this:
11111111.11111111.11111111.00000000

In decimal, this is: 255.255.255.0. This should look familiar if you have ever setup a static IP address.

This allows us to use an address range of 192.168.1.0 – 192.168.1.255, which is the network we are used to. To find the number of hosts available, we just take the number of 0s and raise 2 to that power and subtract 2. The equation is as follows:

$2^{\text{number-of-0s}} - 2$ = Number of possible hosts. In this case: $2^8 - 2 = 254$.

See how the first 24 bits are set to 1 and the last 8 are set to 0? This means that the first 3 octets cannot be used for hosts. The "subnet mask" tells the host "you can only change the last number if you want to be on my network." If you were to set your address to 192.168.2.0, you would not be able to communicate on the network since you have changed the 3rd octet. If you wanted to be able to use the 3rd octet, you would have to use a /16 mask, or 255.255.0.0, which can be expressed as:

11111111.11111111.00000000.00000000 in binary. This allows a range of IP addresses from
192.168.0.0 – 192.168.255.255 or
1100000.10101000.00000000.00000000 – 11000000.10101000.11111111.11111111 in binary.

This allows you to change the 3rd and 4th octets, which means the number of hosts we can have is 2^{16} -2 = 65,534 hosts!

Another common example is a /8, this means only the first octet is reserved for the network and the last 3 octets are for hosts, which gives us a lot of hosts!

11111111.00000000.00000000.00000000 is the mask, which means we can have
192.0.0.0 – 192.255.255.255 or
11000000.00000000.00000000.00000000 – 11000000.11111111.11111111.11111111

in use for our hosts. So 2^{24} -2 = 16,777,214. That's a lot of hosts!

Summary of CIDR

Wow, that's a lot of information! The best way to really understand basic subnetting is to practice, practice, practice! Let's go over a few of the concepts:

192.0.0.0/8 means that the last 3 octets can be used for IP addresses (hosts). The first octet cannot change, or you will be on a different network. This can also be expressed as 192.0.0.0 with a subnet mask of 255.0.0.0

192.168.0.0/16 means that the last 2 octets can be used for hosts. The first 2 octets cannot be changed as they are used for the network. This can also be written as 192.168.0.0 subnet mask 255.255.0.0

192.168.1.0/24 means that the last octet can be used for hosts. The first 3 octets cannot be changed as they are used for the network. This can also be written as 192.168.1.0 subnet mask 255.255.255.0.

192.168.1.1/32 means that NO octets can be used for hosts! This address is the ONLY address allowed. This is used for specifying an individual IP address for a policy. This can be written as 192.168.1.1 subnet mask 255.255.255.255. You will never use this to create networks, only to specify hosts for use in networking policies such as firewall whitelists.

Conclusion

Join me for the next guide where I will walk you through more advanced subnetting concepts!

A gentle guide to subnetting

Part 3: More CIDRs and Route Summarization

Intro

Welcome back to the gentle guide to subnetting! This is the third and final part of this guide and will cover more advanced subnetting topics including "IP summarization". Please make sure you fully understand the first two guides before continuing with this one as these are more advanced topics.

More CIDRs!

In the previous guide, we went over the subnet masks /8, /16, /24, and /32. These are some of the most common subnet masks as they represent the now deprecated class system and are easy to calculate. But a well designed network will rarely use those subnets as it's very rare that those are the exact requirements. Unless your organization is going to never have more than 254 devices, a /24 is probably not the best choice. A /16 provides 65,536 hosts, so that wouldn't make much sense for a department with only 300 employees. This means we need to start using more specific subnets masks! Let's take a look at the /24 in binary again. Remember, a /24 has 3 octets for the network and 1 octet for hosts:

1111111	11111111	11111111	00000000
Network	Network	Network	Hosts

As you recall from the previous guide, to find the number of hosts available, you count the number of 0s in the mask and you raise 2 to that power. In this case, we have 8 0s, so we do 2^8 to get 256. We then subtract the network and broadcast address which gives us 254 usable addresses. So the equation is $2^8 – 2 = 254$. Another piece of information you will want to know is how to determine the number of subnets we can have with a particular mask. We do this by looking at the number of 1s, or subnet bits, in the hosts section and raising 2 to the power of that number. In the above table, there are not any 1s in the hosts section, so let's look at a new subnet mask that does have some!

Calculating number of subnets

Let's say you have a business with 15 departments. You need a scheme that will allow you to create 15 subnets with 14 host addresses in each subnet. How can you tell how many subnets you can create? First, find the number of 1s in the hosts section. We will call these bits "subnet bits" then we raise 2 to that power. It is similar to the hosts equation, only you are using the number of 1s instead of the number of 0s and you are not subtracting 2 from this number. Let's look at an example with a /27:

/28			
1111111	11111111	11111111	11110000
255	255	255	240
nnnnnnnn	nnnnnnnn	nnnnnnnn	sssshhhh

n = network bit; s = subnet bit; h = host bit

As you can see, there are 28 network bits set to "on". So there are 28 1s and 4 0s. This allows for a total of 14 hosts, so let's see how many subnets this gives us. The last octet is still considered the host octet, but 4 of those 0s have been changed to 1s. These 1s in the host section allow us to create subnetworks to further divide the hosts. In a /24, you can only create 1 subnet, or 2^0, but in a /28, you can get 16, or 2^4, subnets since you borrow some of the host bits to use as subnet bits.

I know this is a lot of information, but bear with me! Let's do another example:

Let's say you are the network manager of a very large organization. We will call it Stark Industries. You have a lot of employees and a lot of devices. You also have a lot of departments. Your boss, Mr. Stark, instructs you to design a new network since your current one is very poorly designed. He asks you to create separate subnets for 64 departments. You are also required to create enough IP space for around 1000 hosts per department. That should allow for all of the devices and gadgets that the employees are using to connect to the network without a problem. Let's do this step by step:

1. you need to create 64 subnets. 64 is 2^6. So you need 6 subnet bits (bits set to 1 in the host section).
2. So, let's take a look and see what our possibilities are:
    I. You could use a /26, which looks like this:

/26			
1111111	11111111	11111111	11111100
255	255	255	252
nnnnnnnn	nnnnnnnn	nnnnnnnn	ssssssh

n = network bit; s = subnet bit; h = host bit

II. This is great for the networks requirement, that's 2^6 which gives us 64 subnets! But, wait a minute! What about the number of hosts per subnet? A /26 provides only 2 host bits! That's only 4 hosts! That is not even close to our 1000 host requirement! So we need to rethink this. We have 6 subnet bits, but we also need at least 10 host bits to get our 1000 hosts: $2^{10} – 2 = 1022$. So let's try again, this time, let's change bits right right to left from 1 to 0 until we get enough host bits. So we change all of the 1s to 0s until we have 10 0s:

/22			
1111111	11111111	11111100	00000000
255	255	252	0
nnnnnnnn	nnnnnnnn	ssssssh h	hhhhhhhh

n = network bit; s = subnet bit; h = host bit

III. This looks better! Here, we have 22 bits masked for the network and 6 of those are in the host octet, so those are used for the subnets. So we have $2^6 = 64$ subnet bits and $(2^{10} -2) = 1022$ for the host bits! That meets our requirements!

Ok, let's try another one! This time, you're the IT manager for LexCorp. LexCorp has become a worldwide operation with thousands of workers all over the world. These workers all need to be able to connect to the VPN to perform their jobs securely. This means that they must use 30 very large VPN subnets. Each VPN supports a different region that you cover, so you will need at least 30 networks with 3,000 hosts for each network. So let's see how we would create that.

1. First, you need to find out how many host bits you need. For 3,000 hosts that's going to be $2^{12} -2 = 4094$. So, let's create our chart with 12 bits set to 0 for the hosts:

/20			
1111111	11111111	11110000	00000000
255	255	240	0
nnnnnnnn	nnnnnnnn	sssshhhh	hhhhhhhh

n = network bit; s = subnet bit; h = host bit

2. Ok, we have figured out how many host bits we need, but that's only $2^4 = 16$ subnets! If we move another bit to the subnet portion, we can't meet our 3,000 host goal. We need to find a way to to get $2^5 = 32$ subnets. So let's try by working with the 2nd octet instead of the 3rd:

/13			
1111111	11111000	00000000	00000000
255	248	0	0
nnnnnnnn	ssssshhh	hhhhhhhh	hhhhhhhh

n = network bit; s = subnet bit; h = host bit

3. Let's do the math here:
    a. Hosts: $2^{19} - 2 = 524{,}286$
    b. Subnets: $2^5 = 32$
4. We got it! As you can see, you can't always have a very efficient network. Unfortunately in this case, you have to provide 262,142 addresses for subnets that only need to support 3,000 hosts! But it was the only way we can meet the subnet requirement of 32.

Ok, now we know how to arrange the bits in the mask to create networks of a particular size, what exactly do these networks look like? Let's do a network starting with 192.168.1.0 that requires 4 subnets with 62 hosts in each and see what that looks like:

1. For our 62 hosts, we will need 6 host bits since $(2^6 - 2) = 62$.
2. For our subnets, we need only 2 subnet bits since $2^2 = 4$.
3. Let's create this below in our table:

/26			
1111111	11111111	11111111	11000000
255	255	255	192
nnnnnnnn	hhhhhhhh	hhhhhhhh	sshhhhhh

n = network bit; s = subnet bit; h = host bit

4. Great! So our network is 192.168.1.0/26 (255.255.255.192). So what subnets does this actually give us?
    a. Subnet 1: 192.168.1.0 – 192.168.1.63
    b. Subnet 2: 192.168.1.64 – 192.168.1.127
    c. Subnet 3: 192.168.1.128 – 192.168.1.191
    d. Subnet 4: 192.168.1.192 – 192.168.1.255
5. As you can see, each subnet is 64 hosts, so this network can be divided up 4 times to create 4 subnets!

Let's do one more that's a little more complicated. Let's do 32 subnets with 2000 hosts per subnet:

1. We have a network 172.16.0.0 that needs 2000 hosts, so we will do $(2^{11} - 2) = 2046$. So we need 11 bits for the host.
2. We need 32 subnets, so we need room for 5 subnet bits.

/21			
1111111	11111111	11111000	00000000
255	255	248	0
nnnnnnnn	nnnnnnnn	sssssshh	hhhhhhhh

n = network bit; s = subnet bit; h = host bit

3. Ok, so we have our 11 host bits and our 5 subnet bits. Let's take a look at some of the subnets we can create with this network.

    Subnet 1: 172.16.0.0 – 172.16.7.255
    Subnet 2: 172.16.8.0 – 172.16.15.255
    Subnet 3: 172.16.16.0 – 172.16.23.255
    …
    Subnet 32: 172.16.248.0 – 172.16.255.255

Whoa! How did I find that? Well, we have 32 subnets of 2048 hosts that are possible. So we need to figure out how large our subnets are to find the range. There are several equations out there that might help, The easiest way to do this is to take the octet containing subnet bits, the third octet in this case, and subtract its decimal form from 256. The 3rd octet is 248, so we will subtract that from 256 and we will get 8 addresses available for hosts in that octet. The last octet is all 0s, so that will have 0 – 255 available. So, this means our first usable address is 172.16.0.1, followed by 172.16.0.2, all the way to 172.16.7.255. There are 8 blocks of 255 addresses using this subnet mask. 8 * 256 = 2048, which is the number of hosts we have specified!

Let's try a few more examples with the network provided:

10.0.0.0/11

/11			
1111111	11100000	00000000	00000000
255	224	0	0
nnnnnnnn	ssshhhhh	hhhhhhhh	hhhhhhhh

n = network bit; s = subnet bit; h = host bit

Alright, so the octet containing the subnet bits is the 2nd octet. There are 3 subnet bits and 5 host bits. So let's find the number by which we will increment the 2nd octet:

(256 – 224) = increments of 32 with 2^{21} = 2,097,152 addresses in each. Let's list some of them:

10.0.0.0 – 10.31.255.255

Addresses in this subnet would increment like this:
10.0.0.0, 10.0.0.1, 10.0.0.2…10.0.1.0, 10.0.1.1, 10.0.1.2…10.1.0.0, 10.1.0.1…and so on. As you can see, this is A LOT of hosts available.

10.32.0.0 – 10.63.255.255

Addresses in this subnet would increment like this:
10.32.0.1, 10.32.0.2…10.33.0.1, 10.33.0.2…and so on.

10.64.0.0 – 10.95.255.255

…

10.224.0.0 – 10.255.255.255 is the last subnet available in this network.

This is another concept that you must practice, practice, and practice some more until it becomes easy. Invent your own scenarios. Perhaps create a network that allows 5 hosts and 10,000 subnets and see how you do!

Another common scenario you may encounter is finding the network boundaries for a particular address. This comes in handy when dealing with large address pools that are required to be on the same network.

For instance, I was once given an address of 64.47.228.30/28 and was not told what other addresses were usable. I knew I needed another 3 addresses that were part of that network, but which ones could I use? Let's take a look at this subnet mask:

/27			
1111111	11111111	11111111	11100000
255	255	255	240
nnnnnnnn	nnnnnnnn	nnnnnnnn	sssshhhh

Alright, so the octet that contains the host bits is the last octet. It is a 224, so 256 – 224 = 32. Each subnet will have 32 addresses (30 for hosts after you disregard the network and broadcast addresses.)

So, let's take a look at what networks those would be:

64.47.228.0 - 64.47.228.31 is the first subnet. As you recall, I cannot use 64.47.228.31 because it is the broadcast address. I cannot use 64.47.228.0 because it is the network address. This leaves me with 64.47.228.1 – 64.47.228.30 as usable addresses! The next subnet is 64.47.228.32 – 64.47.228.63, so if I use any addresses from that range, they will be on a different network and won't work. That was easy! Let's try one that's a little bit harder.

What is the address range of the network containing 10.4.34.57/14?

Let's chart it:

/14			
1111111	11111100	00000000	00000000
255	252	0	0
nnnnnnnn	ssssssh	hhhhhhhh	hhhhhhhh

Alright, so we take 256 – 252 to get the block size of our 3rd octet, which is 4, and we create our networks:

10.0.0.0 – 10.3.255.255 Is our first subnet. As you can see, there is no 10.4.34.57 in that network as it only goes to 10.3.255.255.

10.4.0.0 – 10.7.255.255 is our second network. Great! That one contains 10.4.34.57! So, if you wish to address hosts that can communicate with 10.4.34.57/14, you can use any address between 10.4.0.0 – 10.7.255.255!

Let's try one more:

172.45.135.48/17

/17			
1111111	11111100	10000000	00000000
255	255	128	0
nnnnnnnn	nnnnnnnn	shhhhhhh	hhhhhhhh

Ok, so our blocks in the 3rd octet will be 256 – 128 = 128. So let's see what we have:

172.45.0.0 – 172.45.127.255
172.45.128.0 – 172.45.255.255

Which one of these two networks contains 172.45.135.48? If you guessed the second network, you are right! We can use any address from 172.45.128.1 – 172.45.255.254 to address hosts on that network. 172.45.128.0 is the network address and 172.45.255.255 is the broadcast address.

Route Summarization

Ok, now that we have gone through the harder parts of subnetting, it's time to discuss another topic: "supernetting". Supernetting, also called "Route Summarization", allows us to take multiple subnets and consolidate them into one larger subnet. This enables us to create routing tables and policies with fewer networks listed which helps keep them cleaner and easier to manage. Let's say you have 2 networks: 192.168.1.0/24 and 192.168.2.0/24. You wish to create a route that will forward traffic from your router to both of these networks, but you don't want to create two separate routes. Let's break these masks down to binary and see how we can do this:

192	168	1	0
11000000	10101000	00000001	00000000

192	168	2	0
11000000	10101000	00000010	00000000

The goal of route summarization is to create a network using the bits the two addresses share in common. In this case, the first 22 bits are shared by these two networks!

11000000101010000000000100000000

11000000101010000000001000000000

As you can see, the first bit that differs is the 22nd bit. So if we create a new mask to mask out the first 22 bits, we can create a network that contains both networks. This network is:

192.168.0.0/22

In this case, the 3rd octet is changed to a 0 and the mask is changed to accommodate both networks. Let's do one that's a little more difficult:

We will use 3 networks this time:

10.1.32.0/10
10.2.45.0/10
10.3.0.0 /10

00001010000000010010000000000000
00001010000000100010110100000000
00001010000000110000000000000000

In this example, the first bit that differs is the 14th bit. This makes the supernet 10.0.0.0/14, which encompasses all 3 networks!

This is a fairly simple, but very important concept. You will find questions like this on many exams and it will also allow you to have the cleanest network infrastructure in your company!

Conclusion

Alright! You did it! You made it through subnetting! We have been solving real world problems and, with practice, you can be a networking master! Keep practicing and let me know if you have any questions!