10101110

Supernetting

CIDR

IP Subnetting Made Easy!

255.128.0.0

255.255.240.0

A guide to understanding IP subnetting that won't leave you pulling you hair out.

John J. Kowalski

Variable Length Subnet Mask

IP Subnetting Made Easy

By John J. Kowalski

First edition August 2007

ISBN 978-1-61539-174-5

Dedication

To Jesus Christ without whom I would have nothing.

and

To my wife, Leslie without whom I would be nothing.

Acknowledgements

Many, many thanks to those who assisted me in producing this book as reviewers. Gary Roesler, Greg Rinaldi, Rob Richardson, Luke Acha and Bill Pilkey. Thanks guys.

About the author…..

Well, where do I start? I've been in IT since 1986 when I got my start in database programming in the U.S Air Force. After my tour was up, I bounced around from small company to small company building computers and setting up simple networks (Novell, ARCnet and the like, back in the good old days of thin-net). Eventually I settled into a mega-corporation with a worldwide reach. I started out on the help desk and moved up. I have been fortunate to have supported huge data centers with thousands of servers and hundreds of routers, switches and firewalls. I've supported corporations that have facilities world wide with literally millions of customers. I've worked for customers in the manufacturing, banking, healthcare and government sector among others. In my "spare" time I also teach networking courses at Saint Clair County Community College in Port Huron, Michigan where I can be reached at **jkowalski@sc4.edu**.

This is my first book.

Introduction

First off, thanks for reading my book!

I've been in IT for a couple of decades now and have been working for one of the bigger IT providers for over half of that time. I also do some side jobs out of my house for a small number of customers that I have. Apparently that is still not enough to keep me busy so I teach night courses at my local community college. The method of instruction I use for teaching subnetting has been tested on a few hundred (un-suspecting) students with good results, so I hope if helps you as well. Originally I intended this tome only for my students; but many of them encouraged me to make it available to the masses… so here goes!

Few words go better together than "subnetting" and "ugh", except perhaps "painful" and "toothache". It's just one of those maddening subjects that you use, relatively rarely, but are hammered on relatively often by prospective employers, certification tests and vile, wretched network instructors (such as myself).

Fear not. I was in your shoes for years; I *sort* of got it, but was never anything more than a subnetting novice. This is because I was only taught one way of subnetting. The method I call the bit-method. In the bit method we start right from scratch, right from the bit level. Don't get me wrong; this is the ONLY way to truly <u>understand</u> the intricacies and the sheer beauty of this art form that we call subnetting (OK, I obviously lead a dull life if I use words like this to describe subnetting). Let me put it this way, without understanding *why* subnetting works you will never truly understand *how* it works.

My technique is to start with the bit method. This is a time honored method of learning how subnetting works.

The problem is it is cumbersome. Once you know the nuts and bolts of how it works however, I will present you with a quick reference method (a cheat sheet) that I use that will make the task of figuring out subnets simple. If you know the nuts and bolts and want to skip to the end, fine. I can't stop you; it's your book! However, I would recommend you at least peruse the basics starting with the first chapter. I cannot tell you how many people I've run into that learned subnetting wrong the first time around and nine times out of ten this is the primary reason subnetting is so difficult for so many people in the first place. Many of the instructors I learned under did not start at the basics, so I never "un-learned" my bad habits. In my classes I always start with the assumption that the people in my class just landed on a spaceship and know nothing about the topic.

That said, clear your minds. Forget all you know (or think you know) about subnetting. Let's start with a clean slate and get cracking. By the end of the book you'll be a subnetting professional…..hey; have I ever steered you wrong before??

Table of Contents

Chapter 1

Why is subnetting so dang hard?

You're flying through your certification test when suddenly it hits you; the subnetting portion of the test. Your heart sinks. You've studied the concepts, you've flipped bits and you've converted binary to decimal and decimal to binary until you are blue in the face and you still don't get it. You give it your best guess and skip on to the next question.

If you've been in IT for any length of time, this is a brick wall that you've likely hit time and again. If you are at work this means it is time to whip out the subnet calculator. Don't get me wrong; the subnet calculator is a fine crutch. I encourage my students to use it to verify and validate their work. But even the best crutch is still a crutch. What do you do however, in the middle of a job interview when your (hopefully) future boss asks you "what mask will I need to divide a class C address into four even pieces"? Or when you are in a test where there isn't a subnet calculator in sight?

Why is subnetting so dang hard? I've been teaching IT for years now and have never run into a networking concept as enigmatic as subnetting. I've seen it taught a dozen different ways. I've flipped bits, I've converted to and fro, I started from the bit level and worked my way up and from the decimal level and worked my way down

and still it just didn't "click". Oh sure, I understood the concepts while I read it and it was fresh in my mind but when it came down to actually subnetting something – I simply could not complete the task. Of all of the methods I'd had taught to me, none of them took me from concept to practice. Worse than that, math was never my strongest subject and subnetting is all about math? Right? (Wrong!).

One day while driving to work I had an epiphany. I was struck by a blinding light. ***"Subnetting is a piece of cake once you find the patterns"*** a voice said.

Well, ok, it wasn't anything that dramatic, but one day as I was driving to work thinking about subnetting (when you network for a living you day dream about stuff like this) I did, in fact, notice a pattern. Now patterns I can handle; abstract binary concepts are another story. As soon as I got to work I wrote out the pattern that popped into my head and I developed a simple cheat sheet. It's been all down hill for me since then and subnetting no longer scares me.

Now, I could just show you the pattern, but then that would make for an extremely short book. Since you have to learn how to walk before you learn how to run, we will begin by starting with the basics of subnetting. If you are absolutely sure you know the basics, then go ahead and skip to the end. Let me warn you however, that most of the problems I see with people who don't understand subnetting are due to the bad habits and wrong ideas they learned somewhere along the way, so I highly recommend that you read the beginning, middle and end portions of the book. Besides, a review isn't a bad thing, is it?

Chapter 2

What the heck is a network address?

So what is a network address?

A network address is a binary representation of a special numbering system used to find devices on the internet.

Machines do not speak our language and therefore do not know how to handle decimal numbers. Likewise we don't speak binary. Since the computer has no desire to understand our language, we need to understand how computers and networking equipment handle binary numbers.

To review; the numbering system humans use is called the decimal numbering system. The numbering system computers use is called binary numbering. Binary is the language of computers. I've revealed so far that computers use a combination of 8 "switches" to handle IP addressing. Actually, these are not called switches (that would be too easy) they are called bits. Now a bit is a simple little creature that, like a switch, can only assume one of two states; on or off. Recalling that a "1" is on and a "0" is off, when we put eight bits together in their various on or off settings they represent a number (remember one hand represents five and two hands represent ten?)

How many combinations of numbers are possible with 8 bits? Well, to figure this out we use a simple mathematical formula. If there is a single bit and it can assume one of two positions that can be represented by 2x1 or 2^1 (this is spoken as two to the first power). Two bits, each able to represent two positions can be expressed as 2x2 or 2^2. "2^2" is spoken as "two to the second power". Need three bits? No problem. Three bits is 2x2x2 or 2^3. You may have guessed that "2^3" is spoken as "two to the third power".

Here are some of the powers of two written out for you:

2^1	**2**
2^2	**4**
2^3	**8**
2^4	**16**
2^5	**32**
2^6	**64**
2^7	**128**
2^8	**256**
2^9	**512**
2^{10}	**1024**
2^{11}	**2048**
2^{12}	**4096**

... and on and on

So in our original question we had eight bits. Since each can assume one of two positions that would work out to be 2x2x2x2x2x2x2x2 or, put more simply, 2^8. Regardless of how you write it, the answer is 256. This means that 8 bits can be organized in 256 unique combinations or, to better suit our purposes, eight bits can represent up to 256 different numbers. Cool! Now we understand subnetting, right?

Chapter 3

IP addresses rules

Let's lay down some rules for IP addresses. Don't panic, there are only 4 of them.

All IP addresses are:

1. Divided into 4 sections called *octets*
2. Written in dotted decimal format
3. 32 bits long
4. Divided into a network portion and a host portion via their subnet mask

Octets (Latin for a group of eight) are groups of eight bits. We mentioned this earlier, so no surprise here. Remember that **each** octet can range in value from 0 to 255.

Written in dotted decimal format - this means is that in between each octet we stick in a decimal point just to break up the monotony.

Thirty two bits long means that an IP address is comprised of 32 ones or zeros (four octets, eight bits each 4 X 8 = 32).

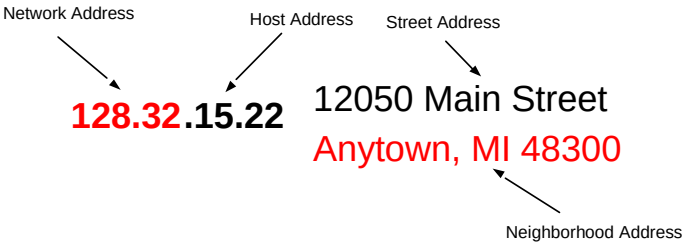
Divided into a network portion and a host portion - more on this in a later chapter.

Chapter 5

IP address class warfare

Not all IP addresses are created equal. In fact there are five different classes of IP addresses. We, however, are going to concentrate on the three most important classes called class A, B and C.

An IP address can be thought of as being similar to your own home address. Your address has two parts, one part that designates your city and a part that designates your individual home. One part gets you into the neighborhood while the other gets you to your house. Network addresses are similar to this example. You have a portion that gets you to your neighborhood (your network) and one that gets you to your PC (your node).



How do we determine what part is host and what part is node? Simple, we apply a "filter" to the IP address called the "subnet mask" - - but more on this later.

Recall that there are three classes that we are going to focus on; Class A, B and C. How do we tell them apart? Well, fortunately there is a fixed standard as shown in the table below.

Class	Decimal Range	Binary Representation
Class A	0-127	00000000 - 01111111
Class B	128-191	10000000 - 10111111
Class C	192-224	11000000 - 11011111

The table above amplifies the left most octets. In fact, this is all we have to consider when determining the class of an IP address. Looking at the table we notice that if the left most bit of the left most octet of an address is "0" then the address has to be a class A address. This is because with the left most bit as zero, the highest we can count to in binary is 127 (01111111 = 64+32+16+8+4+2+1). Therefore a Class A address encompasses addresses beginning with zero thru 127. If the left most 2 bits of the left most octet are "10" then the address must be a class B. Class B addresses range from 128 (the next number where a class A leaves off) to 191. Why? Look at the bits. With the left most two bits being a one and a zero, the least we can make out of it is 128 (10000000) while the most we can make out of it is 191 (10111111 – or if you prefer – 128+32+16+8+4+2+1). This leaves us with a class C address which begins with the three left most bits of the left most octet being "110". Because of this class C address cover from 192 – 224.

So what does all of this mean? How about another analogy? Consider this; worldwide there are relatively few big cities in the world. Big cities, such as New York,

Mexico City and the like have millions of homes in them. Conversely there are millions of small towns in the world with relatively few homes in them. Network addresses are no different. A class A networks are like a large city. There aren't a whole lot of them, but there are a lot of hosts (houses) in each network (city). Conversely, there are many more class C networks but each has only a few addresses per subnet. B addresses, by the way, are "medium-sized-towns" nestled in between the two.

Address Class	1 octet (8 bits)	1 octet (8 bits)	1 octet (8 bits)	1 octet (8 bits)
Class A	Network	Host	Host	Host
Class B	Network	Network	Host	Host
Class C	Network	Network	Network	Host

If a network is a class A network, then the 1st octet represents the network (neighborhood) portion and the last three octets represent the node (house) portion. Class B addresses are evenly split while class C addresses are the inverse of a class A. Remember that each octet is 8 bits long. This means that if we could use all eight bits in the first octet of a class A address there would be a total of 256 (2^8) possible networks, EACH with 16,777,216 (2^{24}) individual addresses (think: few big cities with lots and lots of homes). Class B addresses would likewise be split down the middle with 65,536 networks (2^{16}) each with 65,536 hosts (2^{16}) while a class C would be the inverse of a class A and have 16,777,216 (2^{24}) networks EACH with 256 (2^8) hosts[1] (think: millions of small towns with relatively few homes). Not everyone in the world lives in a big city;

[1] There are actually fewer networks than this for a number of reasons; I chose to oversimplify the point for clarity.

most live in small towns. Likewise not all IP addresses are class A addresses. The majority are class C addresses.

Why do we have different classes? Well, originally it was to give flexibility to IP addressing. It's not a one-size-fits-all world after all. The original intent was to create three sizes of networks to allow some flexibility for network addressing. Today with IP subnetting we can dice and slice IP address ranges into whatever size we want; so while the original intent was good, with subnetting we've gone far beyond that. We can, for instance, purchase a class B address and split it up anyway we want to. We can carve off half of the address for our corporate headquarters, 5,000 addresses for the East coast office, 12 addresses for the Elbonian office and 25,000 addresses for the European offices…..or we can leave it whole. As network administrators, it's our choice (and 'tis better to have choices than not). Instead of being confined to three different types of networks we have many times that capability, thanks to subnetting.

Let's throw one more ingredient into this mix. The group that created the IP address schema was visionary indeed. In addition to creating the classes in an attempt to add flexibility, they also had the foresight to set aside some addresses within each group for special purposes. They realized that if we could only have one unique IP address for each network device that we would run out of IP addresses rather quickly. What to do? Well, they came up with the idea of creating two separate types of IP addresses within each of the A, B and C classes; one for public use and one for the private use. What is public and what is private? Public addresses, simply put, are those that are reachable from other public addresses in the world. Private addresses were set aside so that companies could freely set up networks to their liking

and yet still utilize the power and flexibility of IP addressing. Private addresses, however, are not reachable outside of their private network.

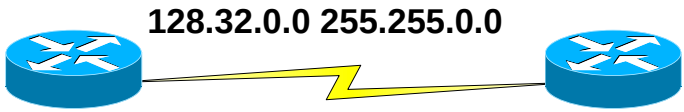
Here's an example. A local veterinarian has 3 computers in her office. They need to talk to each other to log the animal's medical records, print bills and schedule appointments. Across town is a small flower shop. The flower shop has computers also. They have a server in back, a point-of-sale PC in front with a cash register attached and a printer. Each business uses IP addressing to talk within their network. The two businesses, however cannot talk to each other. Since they cannot/will not/do not need to talk to each other, why should they use separate and unique IP addresses? The answer is they don't. In fact, they can use the exact same IP addresses without any issues precisely because they are not connected to each other. Who cares if the IP address of the veterinarian's front office PC is 192.168.0.1 and the IP address of the point of sale of PC at the flower shop is also 192.168.0.1.? Since they do not exchange information and do not know of each other's existence, it does not matter. This is how private addressing works.

Chapter 9

Variable Length Subnetting

Ready for a new concept? As flexible as it is having a class A, B and C addresses, it still isn't enough these days. What we've covered thus far is known as *"classful subnetting"* and if that were all there is to it I could end this book right here. Classful subnetting however is not adequate for all situations. Take for instance, an example wherein you have a corporate headquarters and a regional sales office. You will need three networks to connect each of these (one for the HQ, one for the regional sales office and one for the link between the two). If you used classful addressing you would only have one network available. What now? Buy two more class B's? If so, you would have to use an entire class B address just for the link between the two sites.

Why do I need **65,536** host addresses on a point-to-point link?!?



Recall that each class B address has 65,536 addresses in it. This means that since you would only need two addresses, one for the router at each site, you would waste the rest of the addresses. Imagine; over 65,000 addresses gone to waste. It's no wonder we were running out of addresses!

There is a solution of course. It's called….well it's called a lot of things. Classless subnetting, CIDR (Classless inter-domain routing – pronounced "cedar" or "cider"), VLSM (Variable Length Subnet Masking) or just plain subnetting. We'll stick with the generic term "subnetting" since that always seemed the most descriptive to me.

Recall previously that we had three options (A, B or C) for subnet masks. With subnetting we many times those options. In classful subnetting we set our masks on the hard boundary of the octet. Like this:

Class A: 11111111.00000000. 00000000.00000000
Class B: 11111111. 11111111. 00000000.00000000
Class C: 11111111. 11111111. 11111111.00000000

Let's look at our original example. 128.32.15.22. We know that this is a class B address and looks like this in binary:

128.32.15.22 = 10000000.0010000.00001111.00010110

We therefore know that it's subnet mask looks like this in binary:

255.255.0.0 = 11111111. 11111111. 00000000.00000000

In short, what we have it this:

255.255.0.0 = 11111111. 11111111.00000000. 00000000
Network . Host

Nice... but somewhat restrictive. What if we moved the subnet mask boundary from where it is to a new place…say for instance one bit to the right; like this:

255.255.128.0 = 11111111. 11111111.10000000. 00000000
Network . Host

Now we have something completely new. We have a *classless* subnet mask. While we're moving network boundaries, what if we moved them some more? What if, for instance, we moved the network boundary three bits?

11111111. 11111111.11100000. 00000000
Network . Host

Now we've got flexibility! If however three is good, fourteen would be even better…right?

11111111. 11111111. 11111111. 11111100
Network . Host

Can we really do this? Sure! But what exactly have we done? Well, remember that in order to figure out the number of networks and hosts we count up the bits and use the power of two. With this we learned that a class A address (8 network bits and 24 hosts bits) yields 256 networks (2^8) each with 16,777,216 (2^{24}) hosts. Let's apply the same to the above examples:

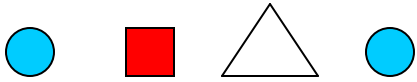
11111111. 11111111.10000000. 00000000 EQUALS
17 Network bits/15 Host bits
2^{17} = 131,072 2^{15} = 32,768

So what have we done? We've taken our class B network and carved it up into 131,072 *individual* networks each with 32,768 hosts.

Chapter 10

Supernetting

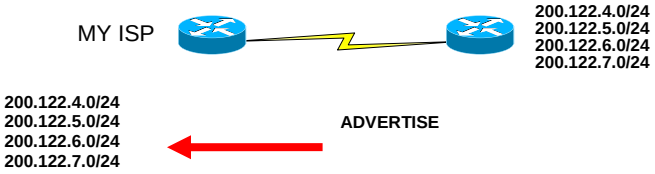
Great! Now that you've got subnetting down it's time to throw something new at you. Are you ready for supernetting? I have a simple test to see if you are. See if you pick the two patterns below that are identical:



If you picked the first and the last, you can supernet. You see, all supernetting consists of is pattern recognition. Just look for identical patterns and you can't go wrong. Lots of people flip out when they see a supernetting question on a test. There's really no need to if you can spot a simple pattern.

Remember how we learned that we can write down subnet masks using a shorthand method that only counts the subnet bits? We went from saying "the 128.32.0.0 network with a 255.255.0.0 mask" to saying "the 128.32.0.0/16 network". Supernetting uses a similar concept. In supernetting we also pay close attention to the network bits in the mask. Supernetting is also called ***aggregation***. This is because when creating a subnet, it means that we have broken up a classful network into pieces. When we create a supernet, however, we combine classful networks into a single "uber-network".

Here's a scenario. I have four class C addresses. I need to advertise these to my Internet Service Provider (ISP) so they can, in turn, advertise them to the world.



If we use what we've learned thus far, my router would have to advertise each of these networks individually to my ISP. This isn't really too bad because it's only 4 networks. But what if I'm a huge company or the US Government and have dozens, or even hundreds of networks? Wouldn't it be neat if there was a way to advertise all of this in a consolidated manner? This is precisely what subnetting accomplishes.

Here are the rules for supernetting

1) Supernets are used to combine two or more classful networks
2) Supernets only work on contiguous networks

Remember that when we created subnets all we did was move the network-host boundary from its classful position to the right? Well, with supernetting we are doing the same thing, but in the opposite direction.

Moving the network/host boundary to the left of the "classful" boundary is called supernetting

Let's look at our previous example. To supernet these addresses our first step is to write out all four of them in binary.

Chapter 11

The Cheat Sheet

OK. You know the nuts and bolts of how this stuff works. This is the part of the book that I said in the beginning that you could turn to if you already knew how subnetting works. This is a method that I came up with that seems to work fairly well for solving most subnet problems.

When you take a certification test, be it for Cisco, Microsoft or others it typically works like this; you show them your photo ID, empty your pockets and are given some scratch paper and a pen or pencil. That's it. No pager, no cell phone, nothing. Nothing but the clothes on your back are allowed in the exam room. This cheat sheet, therefore is something the can be recreated from memory. It goes like this:

1) Write a column of numbers from 1 thru 8
2) Write "Network" above the column
3) Next to the numbers, write down all of the possible subnet masks combinations
4) To the right of that column write down the number 7 thru 0 and write "Hosts" at the top.

That's it. It should look like this when you are done:

NW	Host
1	128 – 7
2	192 – 6
3	224 – 5
4	240 – 4
5	248 – 3
6	252 – 2
7	254 – 1
8	255 – 0

The hardest thing about recreating this is probably the center portion of the column, but remembers to just start with 128 and then divide by half from there. For example; 128, 64, 32, 16, 8, 4, 2, 1. Take the 128 and add 64 (192). Then add 192 and 32 (224) then 224 and 16 (240) until you get to 255. Notice also that the "NW" column plus the "Host" column equal eight when added horizontally (remember; eight bits per octet). The other part of the cheat sheet is the powers of two that we saw earlier:

2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024
2^{11}	2048
2^{12}	4096

That's it. That is all you need from now on. Ready to use our newly created cheat sheet? Let's get at it!