Global Knowledge®

Expert Reference Series of White Papers

Solving the Mysteries of Subnetting

Solving the Mysteries of Subnetting

Raj Tolani, Global Knowledge Instructor

Introduction

Subnetting is not a complicated topic, but it has confused students for a very long time. However, subnetting is an important topic for many different certifications, including those from Cisco®. In the real world, people just punch in the numbers in many of the free subnet calculators that are readily available on the Internet. For exam purposes, you have to do this in a very fast manner since many exams are time-based, and you don't have the luxury of spending precious time on any one question. This whitepaper will solve some of those age-old and complicated subnetting puzzles.

Why Do We Subnet?

Why can't you take the numbers provided by the service providers? The answer is simple – because YOU know your network. Your provider doesn't have any idea about what you will be doing with the numbers internally in your network. Only you know the details, such as how many users you would want in each network. Only you know how many of these networks (sub-networks/broadcast domains) you need. Various design classes will give you different specs on how big the broadcast domain should be (answer depending on the application needs you have). We also do subnetting because you might want to create a manageable, doable, and realistic set of numbers rather than the defaults, which might not work for your needs or could be overkill.

Note: Each router interface is its own broadcast domain, since routers, by default, terminate broadcasts. We plan subnets for each broadcast domain. A Cisco switch by default has all ports in VLAN 1. But the moment you start creating VLANs you need an IP subnet for each VLAN since a VLAN is also a broadcast domain. Remember you need a L3 device, i.e. a router, to interconnect your broadcast domains/VLANs/subnets. Yes subnet = broadcast domain = vlan. We use these terms interchangeably, but subnet is an L3 concept and VLAN is an L2 concept.

There are three classes of addresses that we use for unicast (one-to-one communication) purposes in our IPv4 network (32 bit addresses): Class A; Class B; and Class C. There are some default rules for each class, indicating which octets (8 bits) are network and which octets are host. The defaults are:

Class A	Network	Host	Host	Host	First 8 bits for network; last 24 bits for host
Class B	Network	Network	Host	Host	First 16 bits for network; last 16 bits for host
Class C	Network	Network	Network	Host	First 24 bits for network; last 8 bits for host

Since we are working with binary here, we can easily calculate the number of hosts possible in each class of addresses.

Class A has 24 bits of host possibilities, which comes to 224 - 2 number of hosts. This comes to 16,777,214 hosts in each class A network.

With the Class B address, we have 16 bits of host possibilities. 216 - 2 = 65,534 hosts in each class B network.

With the Class C address, we have 8 bits of host possibilities. 28 - 2 = 254 hosts in each class C network.

The reason we subtracted two from each of these ranges is that the first and the last number in the range has a special meaning to the system. The very first number in the range (all bits in binary OFF) indicates the network/subnetwork you are on. The last number in the range (all bits in binary ON) indicates the directed broadcast on that network/subnetwork. Everything between is a valid host address ready to be assigned.

I don't know of any network that has the capability to support in a flat segment 16,777,214 users with Class A (or 65,534 hosts for Class B). Since these numbers are not realistic, you need to create some realistic doable sub-networks (subnets) with them, based on your needs. With class C, you get 254 hosts in each segment, which is possible but can still be subnetted to accommodate smaller networks (like point-to-point WAN networks with two hosts or any other possibilities).

With all subnetting questions, first you have to check to see what class of address you are working with. This always tells you which octets are network octets so you can take it from there and manipulate the host bits based upon your needs.

Remember your ranges of addresses:

- Class A address range is from 1 – 127 (127 reserved for special purposes)
- Class B address range is from 128 – 191
- Class C address range is from 192 – 223

You also need to see what the default subnet mask is for the given class of addresses. The default subnet mask (meaning not subnetting—one big flat network) for the three unicast class of addresses are:

Class A: 255.0.0.0 -> meaning first 8 bits are ON, can also call it /8

Class B: 255.255.0.0 -> meaning first 16 bits are ON, can also call it /16

Class C: 255.255.255.0 -> meaning first 24 bits are ON, can also call it /24

Note: The slash notation just means how many bits are binary "1" from left to right out of the total 32 bits. For example /24 means that the first 24 bits are ON which also implies the last eight bits are OFF (binary "0"). So, 255.255.255.0

Let’s dig into an example.

Let’s use RFC 1918 Class B address 172.16.0.0. In this scenario we are going to work on this to get eight subnets.

Note: RFC 1918 addresses are private addresses meaning for your private internal networks only. The private range of addresses are:

Class A: 10.0.0.0	–	10.255.255.255
Class B: 172.16.0.0	–	172.31.255.255
Class C: 192.168.0.0	–	192.168.255.255

(How did I get eight subnets; This is a random number I picked, based on eight different VLANs I might want to configure in my network).

The default subnet mask for class B is 255.255.0.0

In binary, the subnet mask is 11111111.11111111.00000000.00000000

Note that there are 16 consecutive zeros, which indicate the number of hosts (as previously discussed).

Remember that we have 65,534 hosts possible, but in ONE big, flat network. We don’t want that for this scenario; we need eight different subnets. So let’s do it.

Ask yourself how many bits you need to get 8 subnets. The formula for this is
2^n = # of subnets where n is the number of bits to use for subnets

In this scenario:
2^n=8 so using basic math n=3 (three occurrences of 2 to get 8 -> 2 x 2 x 2)

Since we said earlier that zeros identify the number of hosts you have (from right to left in the 32-bit subnet mask), the same works for subnets. Subnets are indicated by the number of contiguous ones we have (from left to right in the 32 bit subnet mask).We just determined that we need 3 bits for this example where we need eight subnets. Remember now that subnets are indicated by contiguous ones from left to right.

The subnet mask we had for class B is
11111111. 11111111. 00000000. 00000000

Now that we need to borrow three additional bits, let's write this down. (Remember, this will be from left to right in what was the host portion of the mask)

11111111. 11111111. 11100000. 00000000 (now thirteen bits left for host, indicated by 13 zeroes, so $2^{13} – 2$ would give us 8,190 hosts in each subnet).

This gives us 255.255.224.0 for our subnet mask. The way I got 224 here is the addition of the first three bits that are ON now in the third octet – remember the binary value of first three bits are 128, 64, and 32 – adding up to 224.

Once you determine the mask, you need to write that down in binary. In this case, the number is

255.255.224.0

255. 255. 224. 0

11111111. 11111111. 11100000. 00000000

Now, here is a tricky part. After the first subnet (subnet zero), the decimal value of the lowest active bit is your second subnet (after subnet zero). Then you increment that number by itself until you reach the mask. This will give you all the subnets. Another trick that some people use is to subtract 224 (our subnet mask number) from 256 which will also give us 32 (our increment for subnets).

Let's see it step-by-step:

Decimal value of the lowest active bit.

Active means 1, not active means 0.

Remember, the binary place holders for all eight bits in the octet.

In this example, only the first three bits are on. The decimal values of those bits are 128, 64, and 32; 32 is the lowest active bit versus 64 or 128.

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

Let's increment 32 by itself until we get the mask (224).

32 – Second subnet

64 – Third subnet

96 – Fourth subnet

128 – Fifth subnet

160 – Sixth subnet

192 – Seventh subnet

224 – Eighth subnet

This just gave us all of our eight subnets (with subnet 0 being the first subnet). Let's write these numbers down for clarity and get our range of host addresses within each subnet. (Range is the smallest number possible to the biggest number possible).

172.16.0.0 -> 172.16.31.255 First subnet all possible numbers in the range

172.16.32.0 -> 172.16.63.255 Second subnet all possible numbers in the range

172/16.64.0 -> 172.16.95.255 Third subnet all possible numbers in the range

172.16.96.0 -> 172.16.127.255 Fourth subnet all possible numbers in the range

172.16.128.0 -> 172.16.159.255 Fifth subnet all possible numbers in the range

172.16.160.0 -> 172.16.191.255 Sixth subnet all possible numbers in the range

172.16.192.0 -> 172.16.223.255 Seventh subnet all possible numbers in the range

172.16.224.0 -> 172.16.255.255 Eighth subnet all possible numbers in the range

Do not forget. When we were calculating the number of hosts, we kept subtracting 2 from our host range (the first and the last one). The first and the last one have special meanings. The first number in the range is the subnet itself (the wire that the people are plugged into). The last number is the directed broadcast for all the hosts on that subnet.

So, the valid list of hosts is every number in the range except for the first and the last number.

For the first subnet, it will be 172.16.0.1 to 172.16.31.254.

For the second subnet, it will be 172.16.32.1 to 172.16.63.254 and so on for all subnets.

Also note that we got the end number in the range very fast, because it is one number less than the next subnet. Since the first subnet (third octet here) was 0 and the next subnet was 32 it was easy to see that the first subnet will end at one less than the next number; 31 in our example. We applied the same logic for the next range of subnets. Take a look at the third subnet which is 64 which means that the previous subnet (32) will end at one less than 64; which is true and it ends at 63. Some do point out that those numbers also are increments of 32 so if you can do the additions in your head and not mess up – more power to you.

Most of my clients implement 255.255.255.0 in their network (/24), so they are used to seeing the third octet increment by one. That is not the case here. We saw that the second subnet for example goes from 172.16.32.1 to 172.16.63.254 (valid host ranges). This means the first valid host address is 172.16.32.1, followed by 172.16.32.2 and then 172.16.32.3 and so on. What do you think the number will be when you get to 172.16.32.254? If you said 172.16.32.255 then you are correct and the host after that will be 172.16.33.0. These numbers are perfectly valid for host addresses. I don't want you to just panic because there is a 0 or a 255 in the address and call them subnetwork or broadcast addresses. Always keep in mind that the subnetwork is the first number possible in the range and the directed broadcast is the last number possible in the range. Everything in between is a valid host address, regardless of how weird it looks.

Verify Your Work

This is how to verify your work in binary, if you didn't like the easy "trick" method approach. How do you verify that the three bits we got will give you eight subnets? If you write down all possibilities of 0s and 1s, you will see that there are only eight possibilities:

000 – First subnet	100 – Fifth subnet
001 – Second subnet	101 – Sixth subnet
010 – Third subnet	110 – Seventh subnet
011 – Fourth subnet	111 – Eighth subnet

Don't forget that these are all the possibilities with the first three bits we have borrowed. We still have 13 bits left after using these first three bits (in the third octet so we have five bits in the third octet and 8 bits in the fourth octet left, which is a total of 13 bits).

Pick any of the above three bit combinations and write them down twice (lets say we pick 011 for our example).

172.16. 011 __ __ __ __ __. __ __ __ __ __ __ __ __

172.16. 011 __ __ __ __ __. __ __ __ __ __ __ __ __

Now fill in the first one with all zeros in the 13 bits left blank and the next one with ones in the last 13 bits (remember the first number in the range is all zeros and the last number in the range is all ones).

172.16. 011 0 0 0 0 0. 0 0 0 0 0 0 0 0 which in decimal is 172.16.96.0

172.16. 011 1 1 1 1 1. 1 1 1 1 1 1 1 1 which in decimal is 172.16.127.255

WOW! This gave us the same set of numbers that we got with our short, trick method. This second verification method is not bad either, but we were lucky that we only had to write down all possibilities of 0s and 1s of only three bits. If we had a big number, we will be spending some quality time writing down ones and zeroes, which we might not appreciate.

This is end of subnetting for now, but ask yourself one question. Consider the possibility if one of these subnets was meant for a point-to-point circuit. Remember from our previous discussion that there are only two points in a point-to-point circuit, so why do we have 13 bits for hosts? With 13 bits left for host (all zeros) we get 8190 hosts per subnet. But this is a point-to-point subnet, and we will NEVER need 8190 addresses. This is a waste of addresses and not what you would want to do in a properly designed network. This is especially true when we are complaining that we are running out of IPv4 addresses and are coming up with alternate solutions like IPv6 to fix the problem of not enough addresses. To solve this problem of point-to-point interface addressing, we take the subnetting to the next step called VLSM (variable-length subnet masking). What we have done so far is FLSM (Fixed-length subnett masking). Since our host needs are not the same across the organization, we will be doing VLSM all the time, which is not a stretch from the FLSM case study we have done here and we explore VLSM in our ICND2 and the CCNA Boot Camp curricula at Global Knowledge. I can't wait to see you there to go into more detail about VLSM.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge Check out the following Global Knowledge courses:

Understanding Networking Fundamentals

TCP/IP Networking

ICND1 – Interconnecting Cisco Network Devices 1

ICND2 – Interconnecting Cisco Network Devices 2

CCNA Boot Camp v2

For more information or to register, visit www.globalknowledge.com or call 1-800-COURSES to speak with a sales representative.

Our courses and enhanced, hands-on labs offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 1,200 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and business training needs.

About the Author

Dheeraj (Raj) Tolani has been working with Global Knowledge as a contract instructor teaching various networking courses including CCNP track. He has been in the industry for over 18 years working with various technologies, including Cisco, Banyan Vines, Microsoft, and Novell. Dheeraj has worked as a consultant for various medical, financial, legal, government, and publishing companies. He runs a consulting company based in NYC providing IP integration solutions (www.rajtolani.com).