IP Subnetting: Practical Subnet Design and Address Determination Example

When educators ask students what they consider to be the most confusing aspect in learning about networking, many say that it is IP address subnetting. While subnetting isn't all that difficult in concept, it can be a bit mind-boggling in part due to the manipulations of binary numbers required. Many people understand the ideas behind subnetting but find it hard to follow the actual steps required to subnet a network.

For this reason, even though I explained the concepts behind subnetting in detail in the previous section, I felt it would be valuable to have another section that provides a step-by-step look at how to perform custom subnetting. This section divides subnetting into five relatively straight-forward stages that cover determining requirements, making the design decision of how many bits to use for subnet ID and host ID, and then determining important numbers such as the subnet mask, subnet addresses and host addresses.

My focus in this section is on showing the practical "how" of subnetting. The topics here work through two examples using a Class B and a Class C sample network to show you how subnetting is done, and I am explicit in showing how everything is calculated. This means the section is a bit "number-heavy". Also, I try not to duplicate conceptual issues covered in the previous section, though a certain amount of overlap does occur. Overall, if you are not familiar with how subnetting works at all, you will want to read that section first. I do refer to topics in that section where appropriate, especially the summary tables. Incidentally, I only cover conventional subnetting here, not VLSM.

This section may serve as a useful refresher or summary of subnetting for someone who is already familiar with the basics but just wants to review the steps performed in subnetting. Again, always bear in mind that subnetting is based on the older "classful" IP addressing scheme, and today's Internet is classless, using CIDR.

Background Information: If you are not familiar with binary numbers, binary-to-decimal conversion and masking, and you didn't take my advice in preceding sections to brush up on these concepts using the background explanation of computational math, you ***really*** want to do that now.

Note: If in reading this section you find yourself wanting to do binary-to-decimal conversions or binary math, remember that most versions of Windows (and many other operating systems) have a calculator program that incorporates scientific functions.

IP Subnetting Step #1: Requirements Analysis

When you are building or upgrading a network as a whole, the first step isn't buying hardware, or figuring out protocols, or even design. It's *requirements analysis*, the process of determining what it is the network needs to do. Without this foundation, you risk implementing a network that may perfectly match your design—but not meet the needs of your organization. The exact same rule applies to subnetting as well. Before we look at the gory details of host addresses and subnet masks, we must decide how to subnet the network. To do that, we must understand the requirements of the network.

Key Subnetting Requirements

Analyzing the requirements of the network for subnetting isn't difficult, because there are only a few issues that we need to consider. Since requirements analysis is usually done by asking questions, here's a list of the most important questions in analyzing subnetting requirements:

- What class is our IP address block?
- How many physical subnets are on the network today? (A “physical subnet” generally refers to a broadcast domain on a LAN; a set of hosts on a physical network bounded by routers.)
- Do we anticipate adding any more physical networks in the near future, and if so, how many?
- How many hosts do we have in the largest of our subnets today?
- How many hosts do we anticipate having in the largest subnet in the near future?

The first question is important because everything in subnetting is based around dividing up a Class A, Class B or Class C network, so we need to know which we are dealing with. If we are in the process of designing a network from scratch and don't have a Class A, B or C block yet, then we will determine which we need based on the approximate size of the organization. After that, we need to determine two key numbers: how many physical subnets we have, and the maximum number of hosts per subnet.

Assessing Future Needs During Requirements Analysis

We need to analyze the requirements above not only for the present network, but for the *near future* as well. The current values for these two numbers represent how the network needs to be designed today. However, designing only for the present is not a good idea.

Suppose we have exactly four subnetworks in our network now. In theory, we could use only two bits for the subnet ID, since 2^2 is 4. However, if our company is growing rapidly, this would be a poor choice. When we need to add a fifth subnet we'd have a problem!

Similarly, consider the growth in the number of hosts in a subnet. If the current largest subnet has 60 hosts, you don't want 6a bits for the host ID, because that limits you to 62 hosts. You can divide large subnets into smaller ones, but this may just mean unnecessarily additional work.

So, what is the “near future”? The term is necessarily vague, because it depends on how far into the future the organization wants to look. On the one hand, planning for several years' growth can make sense, if you have enough IP addresses to do it. On the other, you don't want to plan too far out, since changes in the short term may cause you to completely redesign your network anyway.

Key Concept: To successfully subnet a network, you must begin by learning what the requirements of the network will be. The most important parameters to determine are the number of subnets required and the maximum number of hosts needed per subnet. Numbers should be based not just on present needs but requirements in the near future.

IP Subnetting Step #2: The Key Design Trade-off: Partitioning Network Address Host Bits

After we complete our brief requirements analysis, we should know the two critical parameters that we must have in order to subnet our network: the number of subnets required for the network, and the maximum number of hosts per subnetwork. In using these figures to design our subnetted network, we will be faced with the key design decision in subnetting: how to divide the 8, 16 or 24 bits in the “classful” host ID into subnet ID and host ID.

Deciding How Many Bits to Use for the Subnet ID and Host ID

Put another way, we need to decide how many bits to “steal” from the host ID to use for the subnet ID. As I introduced in the topic on custom subnet masks, the fundamental trade-off in choosing this number is as follows:

- Each bit taken from the host ID for the subnet ID doubles the number of subnets that are possible in the network.
- Each bit taken from the host ID for the subnet ID (approximately) halves the number of hosts that are possible within each subnet on the network.

There are six possible ways this decision can be made for a Class C network, as illustrated in Figure 73.

The relationship between the bits and the number of subnets and hosts is as follows:

- The number of subnets allowed in the network is two to the power of the number of subnet ID bits.
- The number of hosts allowed per subnet is two to the power of the number of host ID bits, less two.

We subtract two from the number of hosts in each subnet to exclude the “special meaning” cases where the host ID is all zeroes or all ones. As I explained in the topic on custom subnetting, this exclusion was originally also applied to the subnet ID, but is no longer in newer systems.

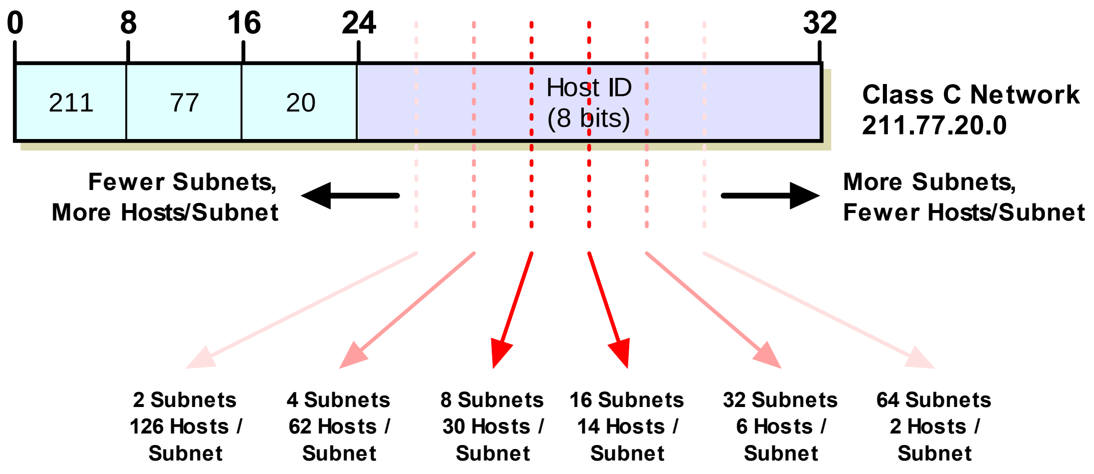

Figure 73: Subnetting Design Trade-Off For Class C Networks

Now, to choose how many bits to use for the subnet we could use *trial and error*. By this I mean we could try to first calculate the number of subnets and hosts when we use one bit for the subnet ID and leave the rest for the host ID. We could then try with two bits for the subnet ID, and then try with three and so on. This would be silly, however; it's time consuming and makes it hard for us to choose the best option. There's an easier method: we can use the subnetting summary tables. They let us look at all our options and usually see immediately the best one for us.

Class C Subnetting Design Example

Let's take an example. Suppose we have a Class C network, base address 211.77.20.0, with a total of 7 subnets. The maximum number of hosts per subnet is 25. Looking at the subnetting summary table for Class C, the answer is instantly clear: we need 3 bits for the subnet ID. Why? This allows us 8 subnets and 30 hosts per subnet. If we try to choose 2 bits, we can't define enough subnets (only 4). As Figure 74 shows, if we choose 4 bits for the subnet ID, then we can only have 14 hosts per subnet.

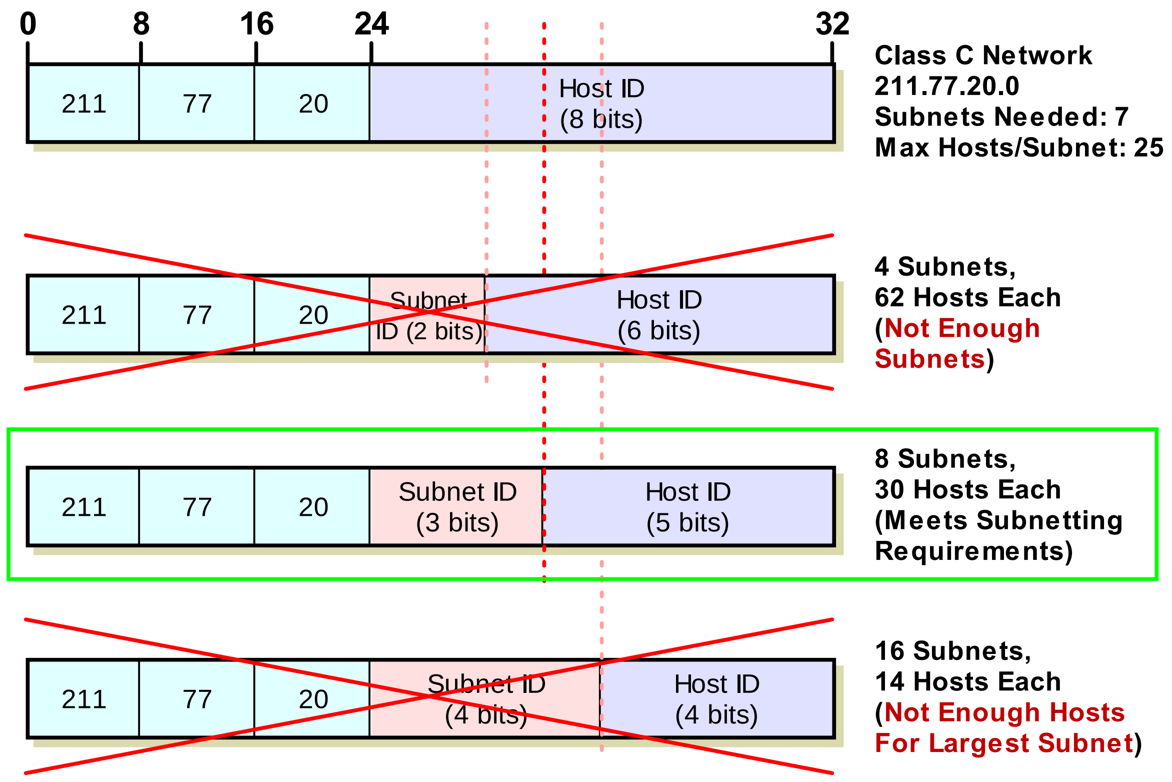

Figure 74: Example Class C Subnetting: An "Easy Decision"

In this particular example, where 7 subnets are needed and 25 hosts are needed for the largest subnet, there is only one choice of subnet ID size that meets the requirements.

Class B Subnetting Design Example

In some cases, especially with larger networks, we may have multiple choices. Consider a more interesting example, the larger Class B network 166.113.0.0, where we have a total of 15 subnets and the largest has 450 hosts. Examining the subnet summary table for Class B suggests four acceptable options, as shown in Figure 75.

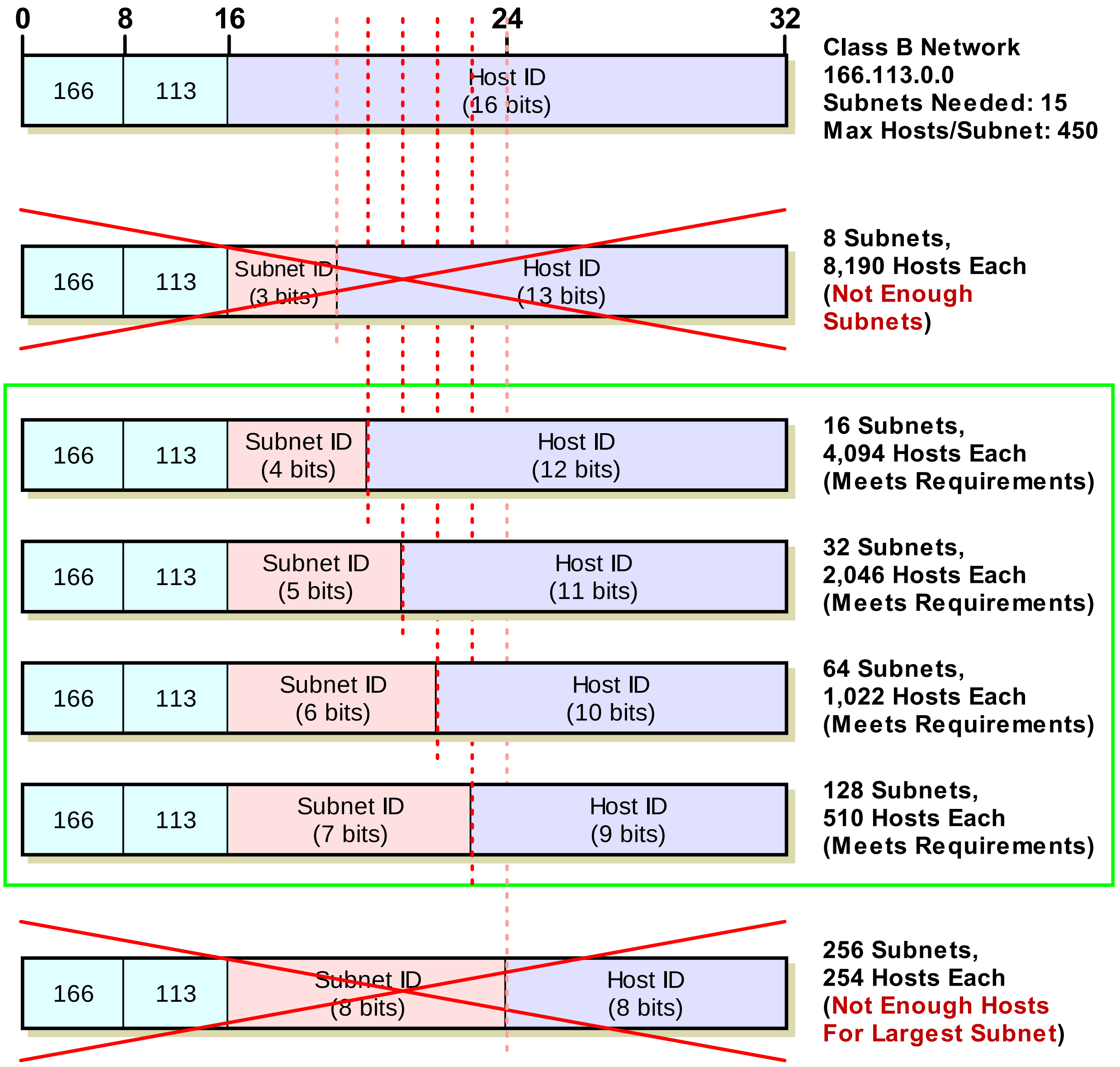
0
8
16
24
32
166
113
Host ID (16 bits)
Class B Network 166.113.0.0
Subnets Needed: 15
Max Hosts/Subnet: 450
Subnet ID (3 bits)
Host ID (13 bits)
8 Subnets, 8,190 Hosts Each (Not Enough Subnets)
Subnet ID (4 bits)
Host ID (12 bits)
16 Subnets, 4,094 Hosts Each (Meets Requirements)
Subnet ID (5 bits)
Host ID (11 bits)
32 Subnets, 2,046 Hosts Each (Meets Requirements)
Subnet ID (6 bits)
Host ID (10 bits)
64 Subnets, 1,022 Hosts Each (Meets Requirements)
Subnet ID (7 bits)
Host ID (9 bits)
128 Subnets, 510 Hosts Each (Meets Requirements)
Subnet ID (8 bits)
Host ID (8 bits)
256 Subnets, 254 Hosts Each (Not Enough Hosts For Largest Subnet)

Figure 75: Example Class B Subnetting: A More Difficult Decision

This Class B network needs at least 15 subnets and must allow up to 450 host per subnet. Three subnet ID bits is too few, and 8 means only 254 hosts per subnet, which is insufficient, but there are four acceptable options, so we must choose wisely. ☺

In all four of these, the number of subnets is equal to 15 or greater, and the number of hosts per subnet is over 450. So, which option should we choose? Usually, we want to pick something *in the middle*. If we use 4 bits for the subnet ID, this gives us only a maximum of 16 subnets, which limits growth in the number of subnets, since we already have 15. The same applies to the choice of 7 bits for the subnet ID, since we already have 450 hosts in one subnet now, and that limits us to 510. Thus, we probably want either 5 or 6 bits here. If we expect more growth in the number of hosts in the largest subnet, we'd choose 5 bits; if we expect more growth in the number of subnets, we'd choose 6 bits. If unsure, it's probably best to assume more growth in the number of hosts per subnet, so here we would choose 5 bits.

The converse problem may also occur: you may be in a position where there are no rows in the table that will match. For example, if our Class C example has 35 hosts in the largest subnet instead of 25, we are out of luck: there is no combination of subnet ID and host ID size that works. The same is true in our Class B example if we had 4,500 hosts in that big subnet instead of 450. In this situation we must either divide the large subnet into a smaller one, use more than one IP address block, or upgrade to a larger block.

Key Concept: If there is more than one combination of subnet ID and host ID sizes that will meet requirements, try to choose a “middle-of-the-road” option that best anticipates future growth requirements. If no combination meets the requirements, the requirements have to change!

IP Subnetting Step #3: Determining The Custom Subnet Mask

Once we have decided how many bits to use for the subnet ID and how many to leave for the host ID, we can determine the custom subnet mask for our network. Now, don't go running for cover on me. ☺ A lot of people's eyes glaze over at mention of the subnet mask, but it's really quite simple to figure out once we have done our homework in making the design decision we did in Step #2. In fact, there are two ways of doing this; one is less work than the other, but they're both quite easy. I was going to call them the “hard” way and the “easy” way, but instead, I'll call them “easy” and “easier”.

Calculating The Custom Subnet Mask

Let's start with the “easy” method, in which we determine the subnet mask in binary form from the information we already have about our network, and then convert the mask to decimal. To refresh your memory and guide the process, remember this: the subnet mask is a 32-bit binary number where a 1 represents each bit that is part of the network ID or subnet ID, and a 0 represents each bit of the host ID.

Class C Custom Subnet Mask Calculation Example

Refer back to the Class C example in the previous topic. We decided to use 3 bits for the subnet ID, leaving 5 bits for the host ID. Here are the steps we will follow to determine the custom subnet mask for this network (illustrated in Figure 76):

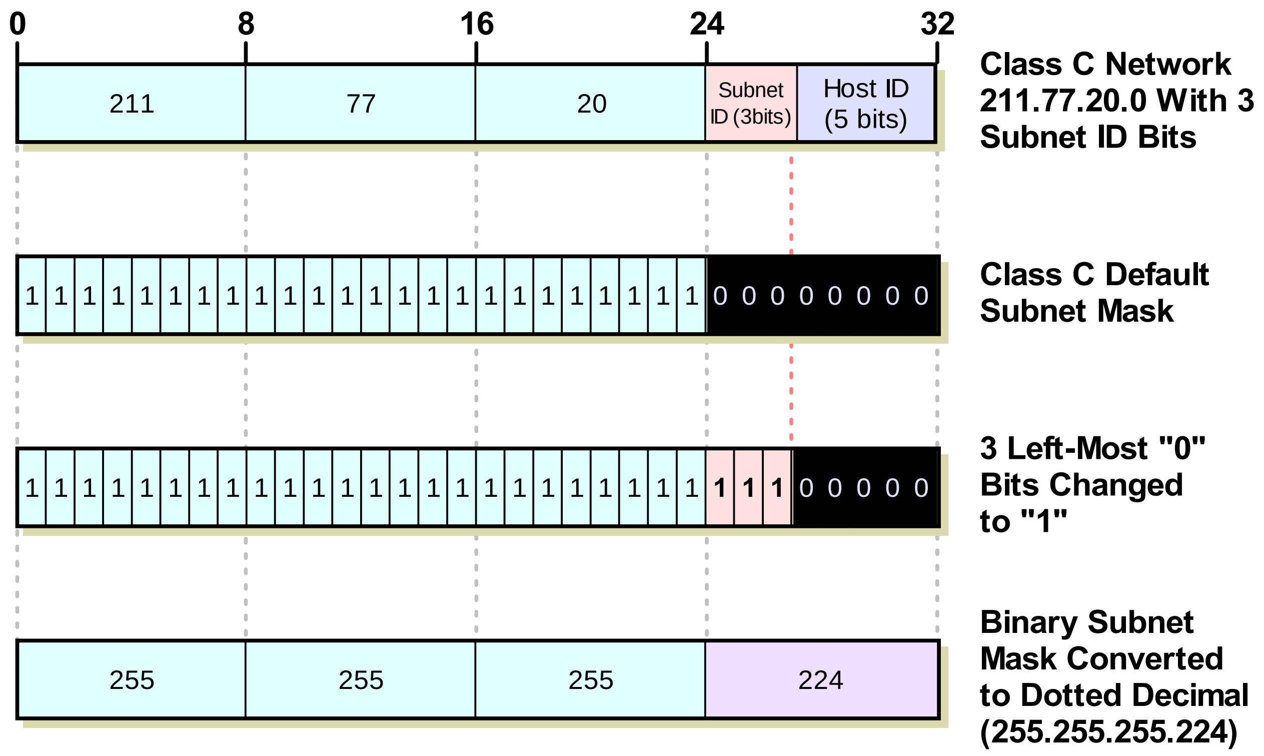

Figure 76: Determining The Custom Subnet Mask for A Class C Network

1. **Determine Default Subnet Mask:** Each of Classes A, B and C has a default subnet mask, which is the subnet mask for the network prior to subnetting. It has a 1 for each network ID bit and a 0 for each host ID bit. For Class C, the subnet mask is 255.255.255.0. In binary, this is:

   11111111 11111111 11111111 00000000

2. **Change Left-Most Zeroes To Ones For Subnet Bits:** We have decided to use 3 bits for the subnet ID. The subnet mask has to have a 1 for each of the network ID or subnet ID bits. The network ID bits are already 1 from the default subnet mask, so, we change the 3 ***left-most*** 0 bits in the default subnet mask from a 0 to 1, shown highlighted below. This results in the following custom subnet mask for our network:

   11111111 11111111 11111111 **111**00000

3. **Convert Subnet Mask To Dotted Decimal Notation:** We take each of the octets in the subnet mask and convert it to decimal. The result is our custom subnet mask in the form we usually see it: 255.255.255.224.

4. **Express Subnet Mask In “Slash Notation”:** Alternately, we can express the subnet mask in “slash notation”. This is just a slash followed by the number of ones in the subnet mask. 255.255.255.224 is equivalent to “/27”.

Class B Custom Subnet Mask Calculation Example

Now, let's do the same example with our Class B network (166.113.0.0) with 5 bits for the subnet ID (with a bit less narration this time; see Figure 77):

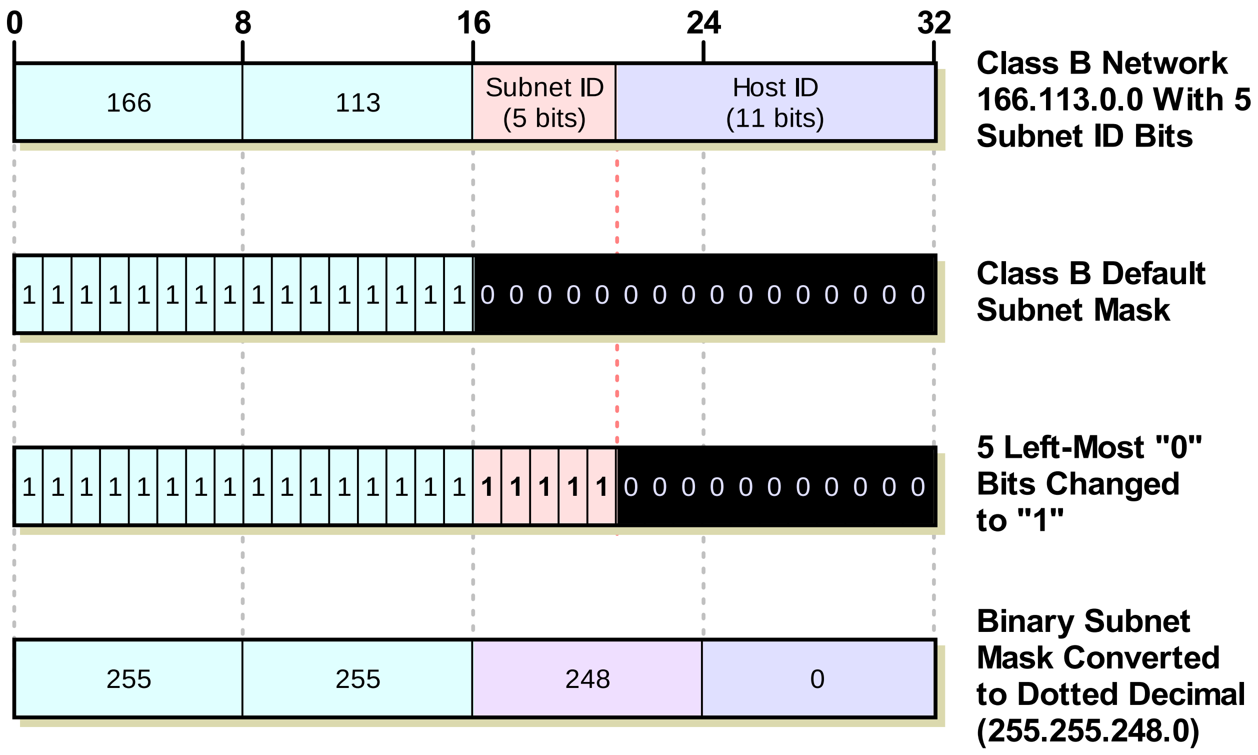

Figure 77: Determining The Custom Subnet Mask for A Class B Network

1. **Determine Default Subnet Mask:** For Class B, the subnet mask is 255.255.0.0. In binary, this is:

   11111111 11111111 00000000 00000000

2. **Change Left-Most Zeroes To Ones For Subnet Bits:** We have decided to use 5 bits for the subnet ID, so, we change the 5 left-most 0 bits from a 0 to 1, shown highlighted below, to give us our binary custom subnet mask:

   11111111 11111111 **11111**000 00000000

3. **Convert Subnet Mask To Dotted Decimal Notation:** We take each of the octets in the subnet mask and convert it to decimal, to give us a custom subnet mask of 255.255.**248**.0

4. **Express Subnet Mask In “Slash Notation”:** We can express the subnet mask 255.255.248.0 as “/21”, since it is 21 ones followed by 11 zeroes. In other words, its prefix length is 21.

Determining The Custom Subnet Mask Using Subnetting Tables

Now, what could be easier than that? Well, you could simply refer to the subnetting summary tables. Find the table for the appropriate class, and then find the row that you selected in the previous step that matches the number of subnet ID bits you want to use. You can see the matching subnet mask right there.

(Hey, it's good to know how to do it yourself! You may not always have tables to refer to!)

IP Subnetting Step #4: Determining Subnet Identifiers and Subnet Addresses

The network ID assigned to our network applies to the entire network. This includes all subnets and all hosts in all subnets. Each subnet, however, needs to be identified with a unique *subnet identifier* or *subnet ID*, so it can be differentiated from the other subnets in the network. This is of course the purpose of the subnet ID bits that we took from the host ID bits in subnetting. After we have identified each subnet we need to determine the address of each subnet, so we can use this in assigning hosts specific IP addresses.

This is another step in subnetting that is not really hard to understand or do. The key to understanding how to determine subnet IDs and subnet addresses is to always work in binary form, and then convert to decimal later. We will also look at a “shortcut” for determining addresses in decimal directly, which is faster but less conceptually simple.

Let's go directly to our examples to see how subnet IDs and addresses are determined. We number the subnets starting with 0, and then going to 1, 2, 3 and so on, up to the highest subnet ID that we need.

Note: I assume in this description that we will be using the all-zeroes and all-ones subnet numbers. In the original RFC 950 subnetting system, those two subnets are not used, which changes most of the calculations below. See here for an explanation.

We determine the subnet IDs and addresses as follows

1. **Subnet ID:** This is just the subnet number, and can be expressed in either binary or decimal form.
2. **Subnet Address:** This is the address formed by taking the address of the network as a whole, and substituting the (binary) subnet ID in for the subnet ID bits. We need to do this in binary, but only for the octets where there are subnet ID bits; the ones where there are only network ID bits or only host ID bits are left alone.

Seem complicated? Let's go back to our examples and we’ll see that it’s really not.

Class C Subnet ID and Address Determination Example

Recall our Class C network, 211.77.20.0. The network address in binary is:

    11010011 01001101 00010100 00000000

We are subnetting using 3 bits for the subnet ID, leaving 5 bits for the host ID. Now let's see the network address with the subnet bits in bold:

11010011 01001101 00010100 **000**00000

These are the bits we substitute with the subnet ID for each subnet. Notice that since the first three octets contain network ID bits, and the network ID is the same for every subnet, they never change. We don't even really need to look at them in binary form, though for clarity we will do so.

Here's how we determine the subnet IDs and addresses, again, starting with 0 (see Figure 78):

0. Subnet #0 has a subnet ID of 0, or 000 in binary. To find the address, we start with the network address in binary, and substitute "000" for the subnet ID bits. Well gee, those bits are already all zero! What this means is that the address for subnet #0 is the same as the address for the network as a whole: 211.77.20.0.

   This is always the case: subnet #0 always has the same address as the network.

1. Subnet #1 has a subnet ID of 1 in decimal or 001 in binary. To find the address we substitute "001" for the subnet ID bits, to yield the following:

   11010011 01001101 00010100 **001**00000

   Converting to decimal, we get 211.77.20.32.

2. Subnet #2 has a subnet ID of 2, or 010 in binary. To find its address we substitute "010" for the subnet ID bits, to give:

   11010011 01001101 00010100 **010**00000

   Which is 211.77.20.64 in binary.

3. Subnet #3 has a subnet ID of 011. As we can see the first three octets of the address are always 211.77.20. The last octet here is "**011**00000", which is 96 in decimal, so the whole address is 211.77.20.96.

Starting to see a pattern here? Yep, the address of any subnet can be found by adding 32 to the last octet of the previous subnet. This pattern occurs for all subnetting choices; the increment depends on how many bits we are using for the subnet ID. Here, the increment is 32, which is 2^5; 5 is the number of host ID bits left after we took 3 subnet ID bits.

4. Subnet #4 is 100, address is 211.77.20.128.
5. Subnet #5 is 101, address is 211.77.20.160.
6. Subnet #6 is 110, address is 211.77.20.192.
7. Subnet #7 is 111, address is 211.77.20.224.

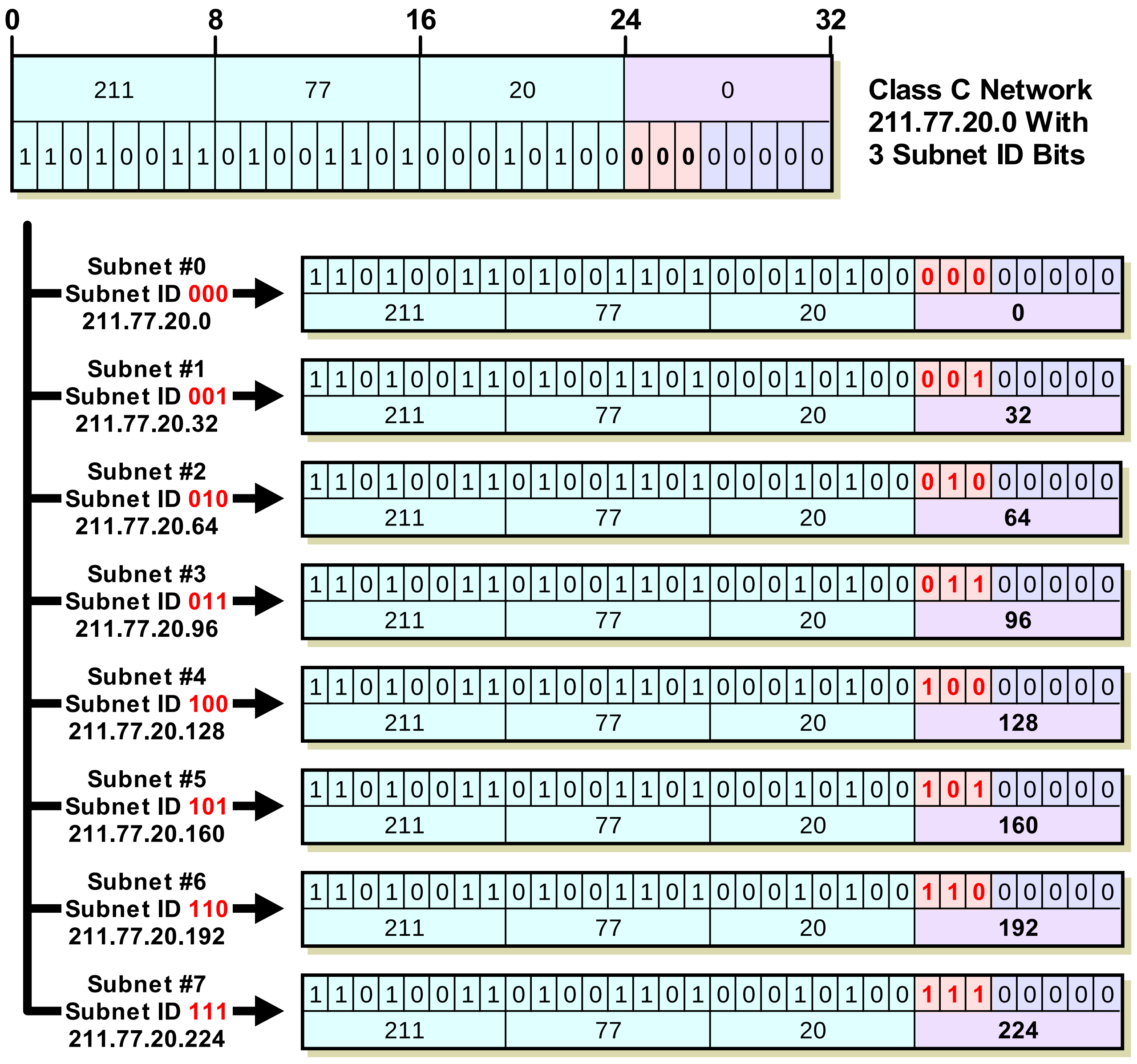

Figure 78: Determining Subnet Addresses For A Class C Network

This diagram shows each of the 8 possible subnets created when we use 3 bits for the subnet ID in a Class C network. The binary subnet ID is simply substituted for the subnet bits, and the resulting 32-bit number converted to dotted decimal form.

Key Concept: The subnet addresses in a subnetted network are always evenly spaced numerically, with the spacing depending on the number of subnet ID bits.

We only needed seven subnets in our example, #0 through #6. Subnet #7 would be a spare. Notice that the last subnet has the same last octet as the subnet mask for our network? That's because we substituted “111” for the subnet ID bits, just as we did when we calculated the subnet mask.

Class B Subnet ID and Address Determination Example

Let's look at our other example now, Class B network 166.113.0.0. In binary this is:

0100110 01110001 00000000 00000000

We are using 5 bits for the subnet ID, leaving 11 host ID bits. The network address with the subnet ID bits highlighted is:

0100110 01110001 **00000**000 00000000

Here, only the third octet will ever change for the different subnets. The first two will always be “166.113” and the last octet will always be zero. There are 32 possible subnets; I'll list the first few so you can see the pattern (refer to Figure 79 as well):

0. Subnet #0 has a subnet ID of 00000. This means the address will be 166.113.0.0, the network address, as we would expect.
1. Subnet #1 has a subnet ID of 00001. The address becomes:

   10100110 01110001 **00001**000 00000000

   This is 116.113.8.0 in decimal.
2. Subnet #2 has a subnet ID of 00010, giving an address of 116.113.**00010**000.0 or 116.113.16.0.
3. Subnet #3 has a subnet ID of 00011 and a subnet address of 116.113.24.0.

Again, the pattern here is obvious: you add 8 to the third octet to get successive addresses. The last subnet here is #31, which has a subnet address of 116.113.248.0, which has the same third and fourth octets as our subnet mask of 255.255.248.0.

Using Subnet Address Formulas to Calculate Subnet Addresses

Since the subnet addresses form a pattern, and the pattern depends on the number of subnet ID bits, it is possible to express the subnet addresses using a single formula for each subnetting option. I have shown these formulas for each of Classes A, B and C in the subnetting summary tables. The formulas can be used to directly calculate the address of subnet #N, where N is numbered from 0 up to one less than the total number of subnets, as we have done above.

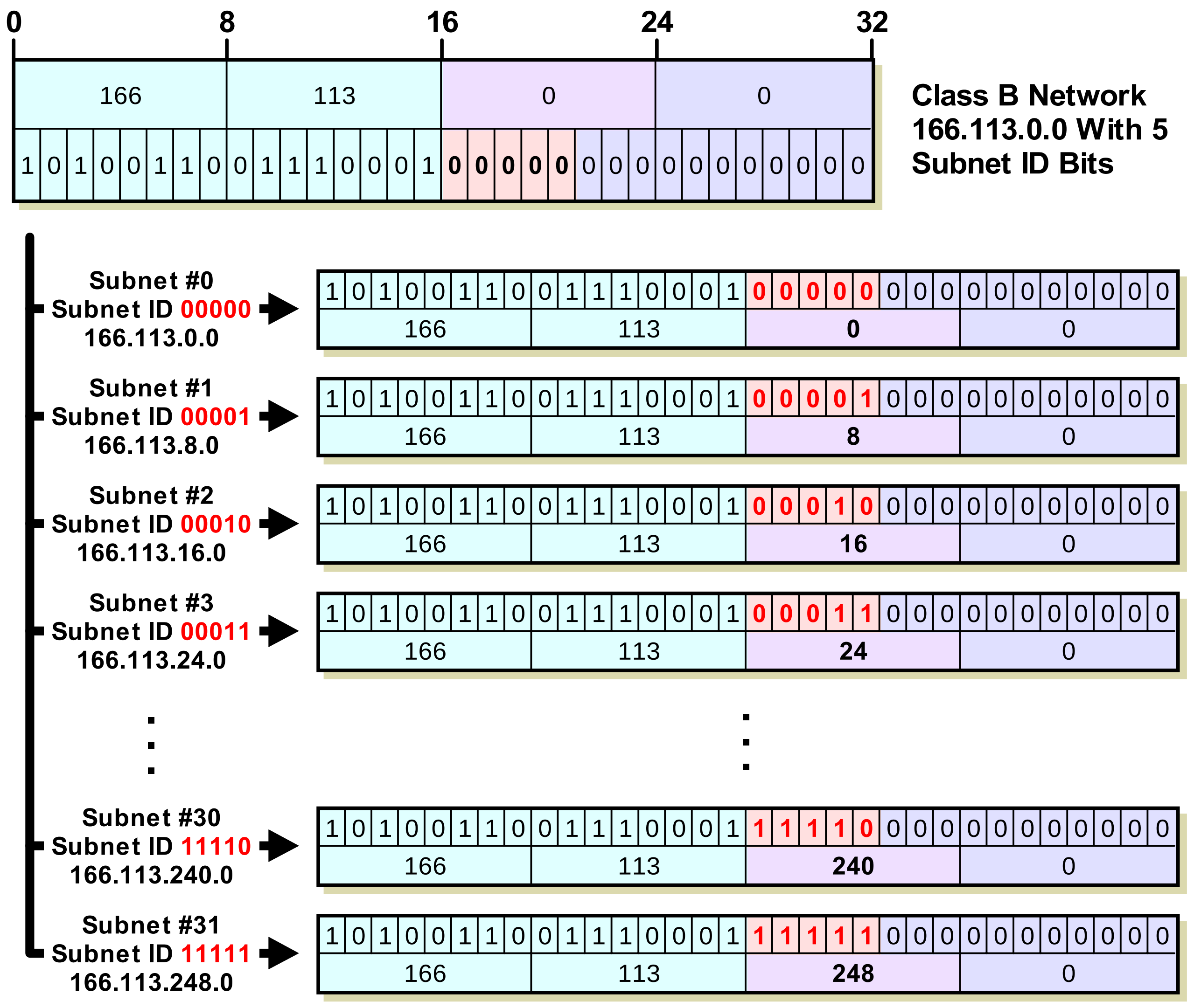

Figure 79: Determining Subnet Addresses For A Class B Network

This is the same as Figure 78, but for a Class B network with 5 subnet ID bits (I have not shown all 32 subnets, for obvious reasons!)

Basic Subnet Formula Calculations

In these formulas, the network ID bits are shown as “x.”, or “x.y.” or “x.y.z.” for the three classes. This just means that the subnet addresses have as those octets whatever the numbers are in those octets for the network address. In our examples, “x.y” would be “166.113” for our Class B network, and “x.y.z” would be “211.77.20” for our Class C.

When the number of subnet bits is 8 or less, the formula is relatively simple, and a calculation is done for only one octet, as a multiplication of N, such as “N*4” or “N*32”. This is usually the case, since the number of subnets is usually less than 256, and it's the case with both of our examples.

In our Class C network with 3 subnet ID bits, the formula from the table is “x.y.z.N*32”. For this network, all subnets are of the form “211.77.20.N*32”, with N going from 0 to 7. So, subnet #5 is 211.77.20.(5*32), which is 211.77.20.160, as we saw before. Similarly, in our Class B network with 5 subnet ID bits, the formula is x.y.N*8.0. In this case “x.y” is 166.113. Subnet #26 would have the address 166.113.(26*8).0, or 166.113.208.0.

This is pretty simple stuff, and make the formulas a good short-cut for quickly determining subnet addresses, especially when there are many subnets. They can also be used in a spreadsheet. The only place that using the formulas requires a bit of care is when the number of subnet bits is 9 or more. This means the subnet identifier crosses an octet boundary, and this causes the formula to becomes more complex. So consider the rest of this topic optional, and skip it if you don't want to complicate your brain. ☺

Subnet Formula Calculations With More Than 8 Subnet Bits

When the number of subnet bits is greater than 8, some of the octets are of the form “N divided by an integer”, such as “N/8”. This is an *integer division*, which means “divide N by 8, keep the integer part and drop the fractional part or *remainder*”. Other octets are calculated based on the *modulo* of N, shown as “N%8”. This is the exact opposite: it means, “divide N by 8, drop the integer and keep the remainder”. For example, 33/5 in integer math sssis 6 (6 with a remainder of 3, drop the remainder, or alternately, 6.6, drop the fraction). 33%5 is 3 (6 with a remainder of 3, drop the 6, keep the remainder).

Let's take as an example our Class B network and suppose that for some strange reason we decided to use 10 bits for the subnet ID instead of 5. In this case, the formula is “x.y.N/4.(N%4)*64”. Subnet #23 in this case would have the address “166.113.23/4.(23%4)*64. The 23/4 becomes just 5 (the fractional.75 is dropped). 23 modulo 4 is 3, which is multiplied by 64 to get 192. So the subnet address is “166.113.5.192”. Subnet #709 would be “116.113.709/4.(709%4)*64, which is 116.113.177.64.

Subnet Formula Calculations With More Than 16 Subnet Bits

Okay, now for the real fun. If you subnet a Class A address using more than 16 bits for the subnet ID, you are crossing *two* octet boundaries, and the formulas become very … interesting, involving both integer division *and* modulo. Suppose we were in charge of Class A address 21.0.0.0 and choose to subnet it. However, we sat down to do this after having had a few stiff ones at the office holiday party, so our judgment is a bit impaired. We decide that it would be a great idea to choose 21 bits for our subnet ID, since we like the number 21. This gives us a couple million subnets.

The formula for subnet addresses in this case, is “x.N/8192.(N/32)%256.(N%32)*8”. Yikes. Well, this is a bit involved—so much so that it might be easier to just take a subnet number and do it in binary, the long way. But let's take an example and see how it works, for, say, subnet #987654. The first octet is of course 21. The second octet is 987654/8192, integer division. This is 120. The third octet is (987654/32)%256. The result of the division is 30864 (we drop the fraction). Then, we take 30864%256, which yields a remainder of 144. The fourth octet is (987654%32)*8. This is 6*8 or 48. So subnet address #987654 is 21.120.144.48.

(Don't drink and drive. Don't drink and subnet either. ☺)

IP Subnetting Step #5: Determining Host Addresses For Each Subnet

Once we know the addresses of each of the subnets in our network, we use these addresses as the basis for assigning IP addresses to the individual hosts in each subnet. We start by associating a subnet base address with each physical network (since at least in theory, our subnets correspond to our physical networks). We then sequentially assign hosts particular IP addresses within the subnet (or in a different manner, if we prefer!)

Determining host addresses is really quite simple, once we know the subnet address. All we do is substitute the numbers 1, 2, 3… and so on for the host ID bits in the subnet address. We must do this in binary of course, and then convert the address to decimal form. Again, we can do some "short-cutting" once the rather obvious pattern of how to assign addresses emerges. We'll look at those later in the topic.

Class C Host Address Determination Example

Let's start with our Class C example again, 211.77.20.0, which we divided into 8 subnets using 3 subnet bits. Here's how the address appears with the subnet bits shown highlighted, and the host ID bits shown highlighted and underlined.:

11010011 01001101 00010100 **00000000**

The first subnet is subnet #0, which has all zeroes for those subnet bits, and thus the same address as the network as a whole: 211.77.20.0. We substitute the numbers 1, 2, 3 and so on for the underlined bits to get the host IDs. (Remember that we don't start with 0 here because for the host ID, the all-zero and all-one binary patterns have special meaning). So it goes like this:

1. The first host address has the number 1 for the host ID, or "00001" in binary. So, it is:

   11010011 01001101 00010100 **00000001**

   In decimal, this is 211.77.20.1.

2. The second host address has the number 2 for the host ID, or "00010" in binary. Its binary value is:

   11010011 01001101 00010100 **00000010**

   In decimal, this is 211.77.20.2

I'm sure you get the picture already; the third host will be 211.77.20.3, the fourth 211.77.20.4 and so on. There is a maximum of 30 hosts in each subnet, as we saw before. So, the last host in this subnet will be found by substituting 30 (11110 in binary) for the host ID bits, resulting in a decimal address of 211.77.20.30.

We can do the same thing for each of the other subnets; the only thing that changes is the values in the subnet ID bits. Let's take for example, subnet #6. It has “110” for the subnet bits instead of “000”. So, its subnet base address is 211.77.20.192, or:

11010011 01001101 00010100 **11000000**

We assign hosts to this subnet by substituting 00001, then 00010, then 00011 for the host ID bits as before:

1. The first host address is:

   11010011 01001101 00010100 **11000001**

   Or 211.77.20.193.
2. The second host address is:

   11010011 01001101 00010100 **11000010**

   Or 211.77.20.194.

And so on, all the way up to the last host in the subnet, which is 211.77.20.222. Figure 80 shows graphically how subnet and host addresses are calculated for this sample network.

One more address we may wish to calculate is the broadcast address for the subnet. This of course is one of the special cases, found by substituting all ones for the host ID. For subnet #0, this would be 211.77.20.31. For subnet #6, it would be 211.77.20.223. That's pretty much all there is to it.

Class B Host Address Determination Example

We can do the same thing for our Class B network, naturally. The address of that network is 166.113.0.0. Now, say we want to define the hosts that go in subnet #13. We substitute 13 in binary (01101) for the subnet ID bits, to get the following subnet address, shown with the subnet ID bits highlighted and the host ID bits highlighted and underlined:

10100110 01110001 **01101000 00000000**

This is the subnet address 166.113.104.0. Now, we have 11 bits of host ID, so we can have a maximum of 2,046 hosts. The first is found by substituting “000 00000001” for the host ID bits”, to give an address of 166.113.104.1. The second host is 166.113.104.2, and so on. The last is found by substituting “111 11111110”, to give an address of 166.113.111.254. Note that since the host ID bits extend over two octets, two octets change as we increment the host ID, unlike our Class C example. The broadcast address is 166.113.111.255.

Key Concept: In a subnetted network, the address of host #H within subnet number #S is found by plugging in the binary value of *S* for the network’s subnet ID bits, and the binary value of *H* for subnet’s host ID bits.

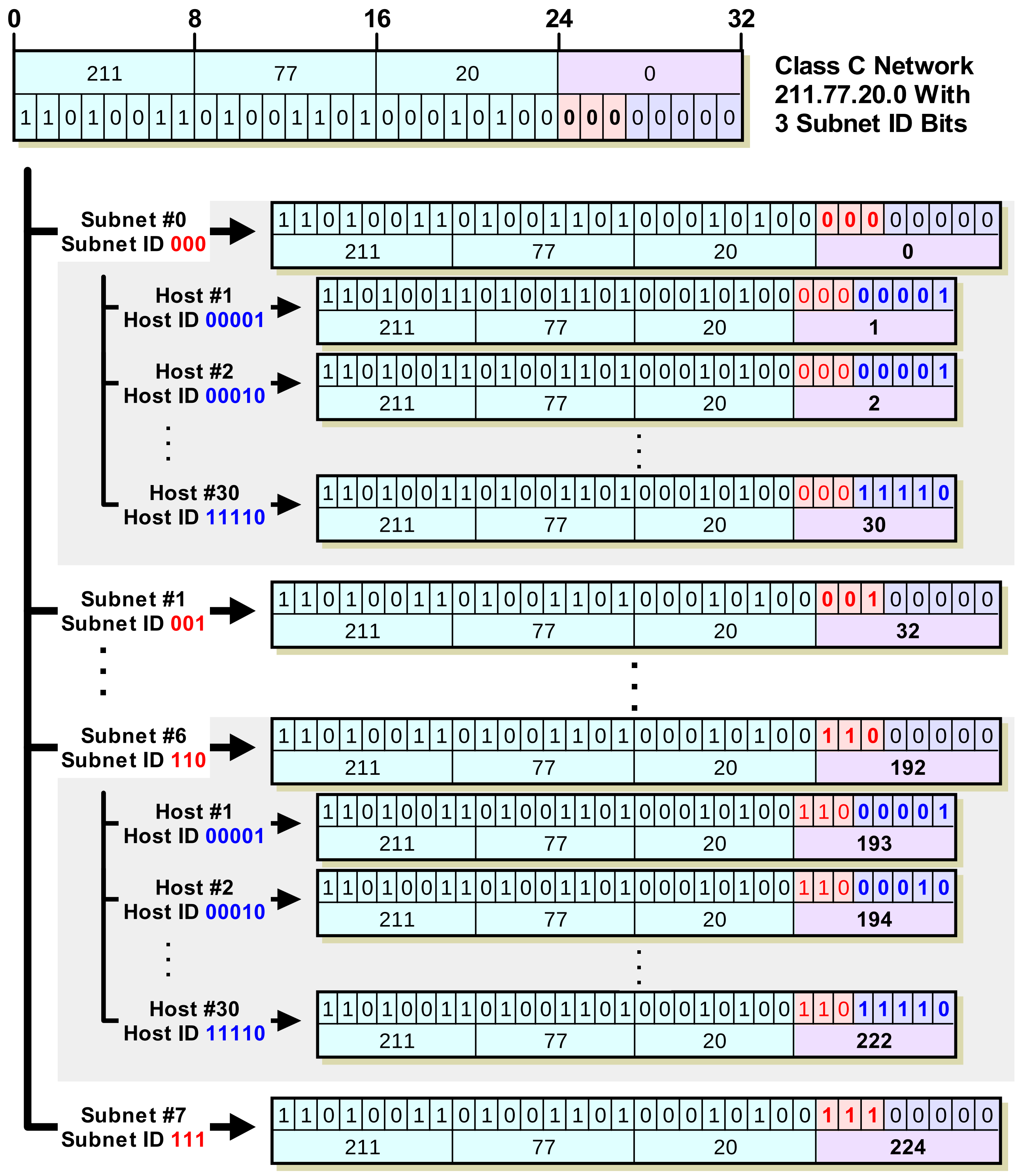

Figure 80: Determining Host Addresses For A Class C Network

This diagram shows how both subnet addresses and host addresses are determined in a two-step process. The subnet addresses are found by substituting subnet ID values (shown in red) for the subnet ID bits of the network. Then, for any given subnet address, we can determine a host address by substituting a host number (shown in blue) for the host ID bits within that subnet. So, for example, host #2 in subnet #6 has “110” for the subnet ID and “00010” for the host ID, resulting in a final octet value of “11000010” or 194.

"Shortcuts" For Quickly Computing Host Addresses

As you can see, defining the host IDs is really quite straight-forward. If you can substitute bits and convert to decimal, you have all you need to know. You can also see that as was the case with defining the subnet addresses, there are patterns that you can use in defining host IDs and understanding how they work. These generally define ways that we can more quickly determine certain host addresses by working directly in decimal instead of bothering with binary substitutions. This is a bit more complex conceptually, so only proceed if you are feeling a bit brave.

The following are some of the "shortcuts" you can use in determining host IP addresses in a subnet environment:

- **First Host Address:** *The first host address is always the subnet address with the last octet incremented by 1*. So, in our class C example, subnet #3's base address is 211.77.20.96. The first host address in subnet #3 is thus 211.77.20.97.
- **Subsequent Host Addresses:** After you find the first host address, to get the next one you just add one to the last octet of the previous address. If this makes the last octet 256 (which can happen only if there are more than 8 host ID bits) you "wrap around" this to zero and increment the third octet.
- **Directly Calculating Host Addresses:** If the number of host ID bits is 8 or less, you can find host #N's address by adding "N" to the last octet's decimal value. For example, in our class C example, subnet #3's base address is 211.77.20.96. Therefore, host #23 in this subnet has an address of 211.77.20.119.

  If there are more than 8 bits in the host ID, this only works for the first 255 hosts, after which you have to "wrap around" and increase the value of the third octet. Consider again subnet #13 in our Class B example, which has a base address of 166.113.104.0. Host #214 on this subnet has address 166.113.104.0, but host #314 isn't 166.113.104.314. It is 166.113.105.58 (host #255 is 166.113.104.255, then host #256 is 166.113.105.0, and we count up 58 more (314-256) to get to #314, 166.113.105.58).
- **Range Of Host Addresses:** The range of hosts for any subnet is determined as follows:
  - **First Address:** Base address of subnet with last octet incremented by one.
  - **Last Address:** Base address of ***next subnet after this one***, less two in the last octet (which may require changing a "0" in the last octet to "254" and reducing the value of the third octet by 1).

  For example, consider subnet #17 in our Class B example. Its subnet address is 166.113.136.0. The address of subnet #18 is 166.113.144.0. So, the range of hosts for subnet #17 is 166.113.136.1 to 166.113.143.254.
- **Broadcast Address:** *The broadcast address for a subnet is always one less than the base address of the subsequent subnet*. Or alternately, one more than the last "real" host address of the subnet. So, for subnet #17 in our Class B example, the broadcast address is 166.113.143.255.

Did I just confuse you? Well, remember, these are shortcuts and sometimes when you take a shortcut you get lost. ☺ Just kidding, it's really not that hard once you play around with it a bit.

In closing, remember the following quick summary when working with IP addresses in a subnet environment:

1. The network ID is the same for all hosts in all subnets, and all subnets in the network.
2. The subnet ID is the same for all hosts in each subnet, but unique to each subnet in the network.
3. The host ID is unique within each subnet. Each subnet has the same set of host IDs.
4. Subnetting is fun!