Appendix C:

IP Subnetting

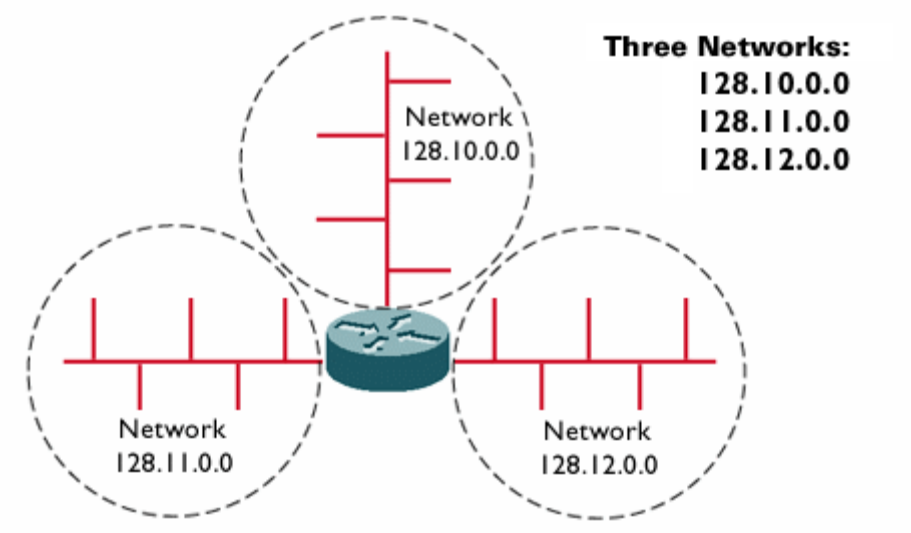
When the Internet Protocol was first proposed in the early 1980s, classful addressing, in theory, seemed to provide a limitless amount of address space. In practice, it became quickly apparent that there were limitations. The major drawback to the system is that, although easy to understand and implement, the boundaries set by the Class A, B, and C addresses do not foster efficient use of the available addresses.

- The Class A address space uses a full 50 percent of the available address space, but it allows for only 126 separate networks. Because there are actually very few organizations assigned Class A addresses, a large portion of the total address space goes unused.
- The Class C address, with only 254 possible hosts, is often too small, causing an organization to move to a Class B address. But the Class B, with ~65,000 hosts, is often too large, causing tens of thousands of addresses to go unused. Remember, each link must be uniquely identified, so if an organization has two separate data-link segments with 300 stations on each, it will need two Class B addresses and will effectively use 130,000 addresses for 300 stations!
- In response to this problem, organizations use multiple Class C addresses instead of using a single Class B address. But this has the negative impact of increasing the size of the global Internet routing table because more networks need to be tracked.
- Because every data link needs to be uniquely identified, there just are not enough network addresses to go around. The rapid growth of corporate intranets, when compounded with the explosion of the Internet, has created a demand for network addresses that the original classful addressing scheme could not meet.

In order to address these problems, a modification to the system is needed that allows the addresses to be used more efficiently. In 1985, RFC 950 was written to standardize a procedure for dividing Class A, B, and C networks into smaller, more manageable sections. This procedure is known as subnetting.

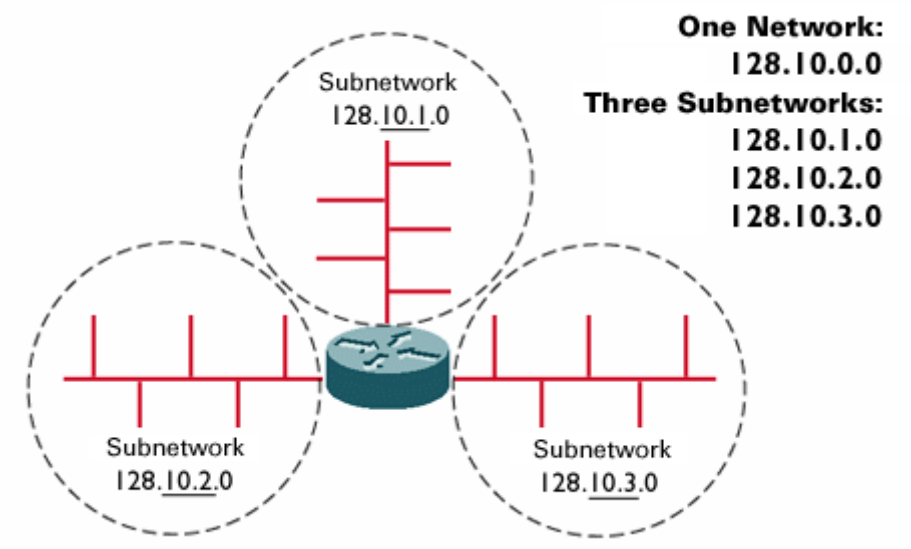
The classful address scheme creates a two-level hierarchy in the Internet: a top level representing the Internet as a whole, and a level below representing the individual networks. For the reasons stated above, what is really needed is a three-level hierarchy that allows for networks to be divided into smaller segments, or subnets.
To accomplish this, the host portion of the address is broken down into two sections, a subnet number and the remaining host portion:

Two-Level Classful Hierarchy:



	Network Prefix	Host Portion

Three-Level Subnet Hierarchy:



	Network Prefix	Subnet Number	Host Portion

The subnet number now identifies a local segment attached to the router. Because only the network portion of the address is advertised to the Internet, subnetworks are only locally significant. Here's how subnetting works:
Suppose you have been assigned the network address of 128.60.0.0. Converting this from dotted-decimal notation into binary, you get:

Network Prefix																Host Portion															
128								60								0								0							
1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

If you take the first four bits of the host portion and use them to identify subnets, you get the following possible binary combinations:

Network Prefix																Subnet				Host Portion											
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**0**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
1	*0*	0	0	0	0	0	0	0	0	1	1	1	1	0	0	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0

Converting these addresses back to dotted-decimal notation, you get:

Network Prefix		Subnet/Host Portion	
128	60	**0**	0
128	60	**16**	0
128	60	**32**	0
128	60	**48**	0
128	60	**64**	0
128	60	**80**	0
128	60	**96**	0
128	60	**112**	0
128	60	**128**	0
128	60	**144**	0
128	60	**160**	0
128	60	**176**	0
128	60	**192**	0
128	60	**208**	0
128	60	**224**	0
128	60	**240**	0

Now in addition to having the major network of 128.60.0.0, you also have up to 16 subnetworks: 128.60.0.0, 128.60.16.0, 128.60.32.0,...128.60.240.0. Each of these subnetwork addresses can be used to define a data-link segment, with the remaining 12 bits of each subnet being used to identify specific hosts on those segments.

Because the network prefix is the only portion of the address that is significant to the Internet, the subnets are not visible outside the private network of the local organization. The route from the Internet to any subnet of a given IP address is the same, no matter which subnet the destination host is on. It is the job of the local routers to determine which subnet a particular host is on. This setup reduces the complexity of the Internet routing table because only a single network address is needed to reach an organization, and it also prevents the depletion of available network addresses because each data link does not need to take up a full IP network.
If we look at the 128.60.16.x subnet and start assigning hosts, we can create addresses 128.60.16.0 through 128.60.31.255:

Network Prefix	Subnet	Host Portion

128	60	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	**0**
128	60	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**
128	60	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	**1**	**0**
128	60	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	**1**	**1**
128	60	**0**	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**
.....																	
128	60	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**
128	60	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
128	60	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
128	60	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**
128	60	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Anything beyond 128.60.31.255 is on the next subnet. The 128.60.32.0 to 128.60.47.255 subnet follows:

Network Prefix		**Subnet**				**Host Portion**											
128	60	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	**0**
128	60	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	**1**
128	60	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	**1**	**0**
128	60	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	**1**	**1**
128	60	**0**	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	**1**	**0**	**0**
.....																	
128	60	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**
128	60	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
128	60	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
128	60	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**
128	60	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

The bits that are used for the network number and subnet number are commonly referred to together as the extended network prefix.

Subnet Masking

A device needs a way to tell what subnet it is located on. As you learned in the addressing section, a device learns the class of its address from the most significant bits of the address (for example, if the first 2 bits are "10," the device knows that the first 16 bits of the address indicate its network and the last 16 bits indicate its host address). With subnetting, a way is needed to tell the device to look beyond its class to determine its subnet. This is done via a subnet mask.
A subnet mask is a 32-bit binary number that corresponds bit for bit to the IP address of the device. The bits of the subnet mask are set to 1 if the system examining the address should treat the corresponding bit in the IP address as part of the extended network prefix. The bits in the mask are set to 0 if the system should treat the bit as part of the host number. For example, a subnet mask for the example above would read:

11111111111111111111000000000000

There are sixteen 1s for the network address portion and four 1s for the subnet. The dotted-decimal notation is again used as a way to simplify reading subnet masks. The mask above would be read as 255.255.240.0.

11111111	11111111	11110000	00000000
255	255.	240.	0

Another way to represent the mask above is to annotate the number of bits in the subnet mask. The mask above could also be referred to as 20 bits of masking, or /20 following the address: 128.60.16.40 /20.

NOTE: When present, routing protocols carry the full 32-bit subnet mask and not just a one-byte field in their header that contains the number of bits in the extended-network prefix. When a device is configured with an IP address, it now needs two pieces of information in order to calculate what its host address is and what its subnet and network are: the actual address and the mask. The device calculates what subnet it is on by doing a logical "AND" between its address and the mask. Performing an "AND" operation means that anytime you "AND" a 0 value to another 0 or a 1 value, the result is 0. Only a 1 ANDed with another 1 value will result in a 1 value. Here's how it works:
0 AND 0 IS 0
0 AND 1 IS 0
1 AND 1 IS 1
Some examples follow:

Example 1: Class B
Let's use a Class B address to illustrate how subnetting works. Let's say you were assigned the Class B address 172.16 from the Network Information Center (NIC). First determine how many subnets you need, and how many nodes per subnet you need to define. A typical (and easy-to-use) Class B subnet mask would be 8 bits. Since the third octet is the first "free" octet for Class B, you will start there. So, an 8-bit subnet mask would be 255.255.255.0. This means you have 254* subnets available and 254 addresses for nodes per subnet.
**Why are there only 254 subnets available instead of 256 (0–255)? You should not use subnet 0 or a subnet of all 1s. With an all 1s subnet mask, this is also your broadcast address. You can configure this subnet, but it is neither proper nor recommended to make your subnet the same as your broadcast address. Subnet 0 is also not recommended.*

Example 2: Class B
Let's say you have just assigned an interface the address 172.16.10.50 with a mask of 255.255.255.0. What subnet is it in? First represent the bits in binary (for Class B, you start with the third octet since octets 1 and 2 are fixed).

SUBNET HOST
00001010 00110010 (address representation - 10.50)
11111111 00000000 (subnet mask representation - 255.0)

00001010 00000000 (results of logical "AND" - subnet 10)

This address is in subnet 10 (172.16.10.0). Valid addresses for subnet 10 would be 172.16.10.1 through 172.16.10.254. Address 172.16.10.255 is the broadcast address for this subnet. According to the standard, any host ID consisting of all 1s is reserved for broadcast.

Example 3: Class B
Let's say you have a need for more subnets than 254. (Remember, this is the maximum number of subnets in a single octet.) Sticking with our Class B address, let's configure an 11-bit subnet. This means we will use all 8 bits from our third octet and the first three bits from the fourth octet. The subnet mask is now 255.255.255.224 (128 + 64 + 32 = 224). Now you need to find out what subnet the following address is in: 172.16.10.170 255.255.255.224. First, denote the address in binary representation (just octets 3 and 4 for a Class B address) like this:

00001010 10101010 (address representation 10.170)
11111111 11100000 (subnet mask representation 255.224-first 11 bits subnet)

00001010 10100000 (results of logical "AND") 10 160

So, the address here is in subnet 172.16.10.160. The valid addresses for this subnet are 172.16.10.161 through 172.16.10.190 (.191 is the broadcast address). As soon as you hit 10.192, the bits in the subnet change and you move into subnet 10.192.

Example 4: Class B
Consider an example where the mask is shorter than one octet. Now we want only a few subnets, but need many hosts per subnet. We'll use a 3-bit subnet mask. Now we have: 172.16.65.170 255.255.224.0 (the mask is now the first 3 bits of the third octet). What subnet is this address in?

01000001 10101010 (address representation 65.170)
11100000 00000000 (subnet mask representation 224.0)

01000000 00000000 (results of logical "AND" - subnet 64) 64

So the subnet here is 172.16.64.0. The range of addresses that would fall into subnet 64 would be 172.16.64.1–172.16.95.254, with 172.16.95.255 as the broadcast address. The next subnet would be 172.16.96.0. Class A and Class C map out exactly as Class B. The only differences are the octet at which subnetting starts and how many octets you can use for subnetting.

Example 5: Class C

Suppose the NIC assigned the address 192.1.10. You will need to use the fourth octet for your subnetting needs. Let's use a 4-bit subnet mask and map out the following address: 192.1.10.200 255.255.255.240:

```
 11001000 (address representation for 200)
 11110000 (subnet mask representation for 240)
 --------
 11000000 (results of logical "AND" - 128+64=192)
```

So, address 192.1.10.200 is in subnet 192. The valid range of addresses in this subnet would be 192.1.10.192 through 192.1.10.206, with .207 as the broadcast address. The next subnet would be .208.

Keeping the same subnet mask, you can choose different addresses to be in different subnets. For instance, address 192.1.10.17 255.255.255.240 is in subnet 16 and, therefore, has another unique subnet address, with valid addresses in the range of 192.1.10.17 through 192.1.10.30. If you want no subnetting, use these default masks (255 — strictly follow number, 0 — wildcard):

Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0

How does a router using a routing protocol that does not transmit masking information know what mask to use when it learns a new route? If the router has a subnet of the same network number assigned to a local interface, it assumes that the learned subnetwork was defined using the same mask as the locally configured interface. If the router does not have a subnet of the learned network number assigned to a local interface, the router has to assume that the network is not subnetted and applies the natural classful mask of the route.

Subnetting Tips/Tricks

It may be useful at some point to be able to quickly do subnet calculations and to answer basic addressing questions without having to look at the binary representation of an address. There are several techniques involving decimal numbers that make answering some subnetting questions rather simple. Questions that may be answered with these techniques include the following:

- What subnet is a given address/mask in?
- What is the broadcast address for the subnet of a given address/mask?
- What is the valid range of hosts for the subnet of a given address/mask?

Actually, all three of these questions can be answered fairly easily with a little simple math. Let's consider the same address that was previously used in Example 3 above. The address in that example was 172.16.10.170 with a mask of 255.255.255.224.

To answer the questions above, simply start by subtracting the decimal value of the last nonzero octet of the subnet mask from 256. In this case, the mask extends into the fourth octet, which has a value of 224. By subtracting 224 from 256, we get 32. This value is the number that dictates the boundary for valid subnets. In this case, each subnet is a multiple of 32. Thus, there is the 172.16.10.0 subnet, the 172.16.10.32 subnet, the 172.l6.10.64 subnet, and so on, up to the 172.16.10.224 subnet.

To answer the first question of exactly which subnet an address is part of, you must simply find the lower boundary of the subnets the address falls between. Since the address given in the example was 172.16.10.170, it is between subnets 172.16.10.160 and 172.16.10.192.

Thus, the address is in the 172.16.10.160 subnet (the same answer found in Example 3 above). One easy way to arrive at this lower boundary is to start at zero and keep adding the value found from the initial subtraction (32 in this case) until the subnet just below the address is reached. The correct subnet has been reached when adding this value one more time results in a subnet that is higher than the address in question. In this case, 0 + 32 = 32, 32 + 32 = 64, 64 + 32 = 96, 96 + 32 = 128, 128 + 32 = **160**. If 32 was added once more, the result would be the 192 subnet, a result that cannot be correct because the address in question has a host number of 170.

To answer the second question, simply add the number from the initial subtraction (32 here) to the subnet found in the first question and subtract one. In this example, the next subnet would be 172.16.10.192 (160 + 32= 192). Subtracting one from the last octet yields 172.16.10.191, the broadcast address of the 172.16.10.160 subnet. Again, this is the same answer pointed out in Example 3 above.

Finally, the third question can be answered by adding 1 to the subnet address found from the first question and subtracting 1 from the broadcast address found in the second question (essentially the same as subtracting 2 from the next subnet address). In our case, adding 1 to the 172.16.10.160 subnet yields 172.16.10.161 as the first valid host address. Subtracting 1 from the broadcast address of 172.16.10.191 yields 172.16.10.190 as the last valid host address in the subnet. Thus, the range of valid host addresses is 172.16.10.161 – 172.16.10.190 (the same answer found in Example 3).

Subnet Design Considerations

Below is a handy chart that can help you determine how much subnetting to use:

Host/Subnet Quantities Table

Class B No. bits	Mask	Effective Subnets	Effective Hosts
2	255.255.192.0	2	16382
3	255.255.224.0	6	8190
4	255.255.240.0	14	4094
5	255.255.248.0	30	2046
6	255.255.252.0	62	1022
7	255.255.254.0	126	510
8	255.255.255.0	254	254
9	255.255.255.128	510	126
10	255.255.255.192	1022	62
11	255.255.255.224	2046	30
12	255.255.255.240	4094	14
13	255.255.255.248	8190	6
14	255.255.255.252	16382	2

Class C No. bits	Mask	Effective Subnets	Effective Hosts
2	255.255.255.192	2	62
3	255.255.255.224	6	30

4	255.255.255.240	14	14
5	255.255.255.248	30	6
6	255.255.255.252	62	2

*Subnet all zeroes and all ones excluded.
*Host all zeroes and all ones excluded