Working with Subnet Masks and Supernets

Designing a Constant-Length Subnet Mask (CLSM)

To design a CLSM subnet mask, in which each portion of the network has the same number of addresses, follow these steps:

1. Decide how many subnets are needed.
2. Because the number of subnets needed must be represented by a bit pattern, add 2 to the number of subnets needed (one for the network address, the other for the broadcast address), then jump to the nearest higher power of two. If the result of the sum is an even power of two, you can use it directly.
3. Reserve bits of the host portion's address from the top down.
4. Be sure that there are enough host addresses left over on each subnet to be usable.
5. If you're using RIP, use the formula $2^b - 2$ to calculate the number of usable subnets from a mask, in which b is the number of bits in the subnet mask, and 2 is subtracted to account for the network (all-zeroes) and broadcast (all-ones) addresses that every IP network and subnetwork require. If you're using some other routing protocol in which all 0s and all 1s aren't taken, use the formula 2^b instead.

Here's an example to help you put this method to work:

1. ABC Incorporated wants 12 subnets for its Class C address 200.10.10.0. No subnet needs more than 10 host addresses.
2. Add 2 (for network and broadcast addresses) to 12 to get 14. The nearest power of two is 16, which equals 24. This means a 4-bit subnet mask is required.
3. Reserving 4 bits from the top down creates a subnet mask with the pattern 11110000. The decimal value for this number is 128 + 64 + 32 + 16, or 240. This extends the default subnet mask for the Class C address from 255.255.255.0 to 255.255.255.240. (This is because we're "stealing" 4 bits from the host portion of the address.)
4. To calculate the number of host addresses for each subnet, reverse the logic from the subnet mask. In English, this means that every bit used for the subnet mask cannot be used for host addresses. Count the number of 0s left over in the subnet mask to determine the number of bits left over for the host addresses. In this case, that number is four.
5. The same formula that's used to calculate the number of subnets works to calculate the number of hosts, in which b becomes the number of bits in the host address—namely $2^b - 2$, or $2^4 - 2$, which equals 14. (Note: Here again, for other routing protocols, it's not necessary to subtract 2 in the calculation.)

Remember, the whole purpose of this exercise is to compare the number of hosts needed for each subnet to the number you just calculated. In other words, if you need more than 14 hosts per subnet, this subnet mask does not produce the desired results. However, because only 10 hosts are needed per subnet, as stated in the requirements for ABC Incorporated, this design works as required.

As a quick summary, here's what you just did: Based on the requirements for 12 subnets in a Class C address, in which no individual subnet needs more than 10 host addresses, you calculated that a 4-bit subnet mask is required. Because the corresponding bit pattern, 11110000, equates to 240 in decimal, the default subnet mask for Class C, 255.255.255.0, must be changed to steal those 4 bits from the host portion of the address, so the actual subnet mask becomes 255.255.255.240.

Because the remaining four bits for the host portion allow up to 14 host addresses per subnet,and the requirements call for no more than 10 addresses per subnet, the design works as intended. However, always remember this last step: *check your work*—renumbering IP networks is a messy, tedious business, and you don't want to be forced to do this over!

Another Constant-Length Subnet Mask Example

Let's pick a more ambitious design this time, which shows how subnet masks or host addresses can extend across multiple octets. Use the following steps:

1. XYZ Corporation, a large multinational company with hundreds of offices worldwide, wants 300 subnets for its Class B address 178.16.0.0. No subnet needs more than 100 host addresses.
2. Add 2 (for network and broadcast addresses) to 300 to get 302. The nearest power of two is 512, which equals 2^9. This means a 9-bit subnet mask is required.
3. Reserving 9 bits from the top down creates a subnet mask with the pattern 11111111 10000000. Because this subnet mask extends across two octets, the subnet must be calculated separately for each octet. For the first host portion octet, the value is 255 because all the bits are set to 1s. For the second octet, the first bit (which equals 2^7) is the only one set to a 1; this translates to a decimal value of 128. This extends the default subnet mask for the Class B address from 255.255.0.0 to 255.255.255.128 (because we're "stealing" 9 bits from the host portion of the address, this produces an extended network prefix of /25).
4. To calculate the number of host addresses for each subnet, reverse the logic from the subnet mask. This means that every bit used for the subnet mask cannot be used for host addresses. Count the number of 0s left over in the subnet mask to determine the number of bits left over for the host address.
5. In this case, that number is 7. The same formula that's used to calculate the number of subnets works to calculate the number of hosts, in which b becomes the number of bits in the host address—namely $2^b - 2$, or $2^7 - 2$, which equals 128 – 2, or 126.

Remember,the whole purpose of this exercise is to compare the number of hosts needed for each subnet to the number you just calculated. In other words, if you need more than 126 hosts per subnet, this subnet mask does not produce the required results. But because only 100 hosts are needed per subnet, as stated in the requirements for XYZ Corporation, this design works as intended.

Designing a Variable-Length Subnet Mask (VLSM)

Designing and configuring variable-length subnet masks is more time consuming and difficult but makes much better use of available address space. One way to think of VLSM is subnetting an already subnetted address, starting with subnetting the network with the largest number of hosts and working your way down to subnetting the network with the smallest number of hosts. A simple example will explain how this might be the case. Let's assume that Plotchnik University (PU) has a Class B address, and it needs 135 subnets to support all of the various campuses, offices, and operations. Because 135 is between 128 and 256, a CLSM subnet mask requires 8 bits so that the default subnet mask for Class B—namely 255.255.0.0—becomes 255.255.255.0 for PU, providing support for the necessary number of subnets. So far, so good.

But let's suppose that over half the subnets at PU require support for 30 devices or less. That means all of those networks that could work with a 5-bit subnet mask must waste an extra three bits of space (or up to 224 unused addresses per subnet). As you can probably guess, VLSM was devised to cover this contingency. Instead of following the method from the preceding section on CLSM, you must instead:

- Analyze the requirements for individual subnets.
- Aggregate those requirements by their relationships to the nearest power of two that is at least two greater (remember the all-zero and all-one networks require representation) than the number of such subnets required.
- Use the subnets that require the largest number of devices to decide the minimum size of the subnet mask. (Remember, the smaller the subnet mask, the lower the number of subnets it supports, but the higher the number of host addresses left over.)
- Aggregate subnets that require smaller numbers of hosts within address spaces defined by the largest subdivisions (those defined in the previous step).
- Define a VLSM scheme that provides the necessary number of subnets of each size to fit its intended use best by aggregating subnets large and small to create the most efficient network traffic patterns. In other words, neighboring subnets with the same extended network prefix should be placed in the same "container" defined by a higher-level network prefix, if possible. Assume your design calls for several subnets with 3 bits of subnetting, and you have needs for 30 subnets with 5 bits of subnetting. This means that three 3-bit prefix containers will be required because each can offer only 14 unique 5-bit subnets, as it were. If you place those 5-bit subnets together within the addresses that fall into any single 3-bit subnet range and are most likely to exchange data with each other, this results in the most efficient routing behavior possible.

Thus,VLSM makes it possible to create routing hierarchies and limit traffic on the backbone by making sure that smaller subnet address spaces can access the resources they need as efficiently as possible. One way to do this is to make sure that the servers such subnets access are most frequently attached directly to that subnet, or that they're in the same 3-bit subnet address space as the original subnets, even if that 3-bit subnet space was subdivided into fourteen 5-bit containers.

Calculating Supernets

Supernets "steal" bits from the network portion of an IP address to "lend" those bits to the host. As part of how they work, supernets permit multiple IP network addresses to be combined and make them function together as if they represent a single logical network. This greatly improves performance for communications inside the local network because it eliminates the need for internal routing. Incidentally, this also permits more hosts to be addressed on a supernet than in the combination of multiple addresses. Here's why:

1 Combining eight Class C addresses steals 3 bits from the network portion of the address and adds them to the host portion of the address.

2. Thus, rather than supporting only 8 bits for the host address portion, the supernet now supports 11 bits (8 + 3) for host addresses. The resulting subnet mask looks like this: 255.255.248.0 (rather than the default 255.255.255.0). There are two ways to calculate a supernet mask. Recognizing that you're borrowing 3 bits from the right side of the third octet, you also should recognize that the resulting bitmask for that octet becomes 11111000, which calculates to 248. The other way to calculate this value is to recognize that the largest number that can be represented in three binary bits is 111, which calculates to $2^3 – 1$, or 7 in decimal. If you subtract that number from 255, you get 248,

the same answer, more quickly than with the other formula. Because both methods work, use whichever one is easiest for you.

3. The number of usable hosts for this supernet, using the old familiar $2^b - 2$ formula, is $2^{11} - 2$, which calculates to 2,046.
4. Each individual Class C address can only address 254 ($2^8 - 2$) hosts. Eight times 254 is 2,032. Because 2,046 – 2,032 = 14, supernetting eight Class C addresses yields access to some additional hosts, when compared to using each Class C address separately.

It's actually far more important that supernetting allows an entire group of hosts to be reached through a single router address, and for those hosts to communicate directly with each other, rather than requiring routing among all the hosts in that address pool.

It is hoped that this helps illustrate why supernetting is a useful tool for Internet service providers (ISPs) that can combine multiple Class C addresses to serve larger populations than might seem possible. In fact, the combination of supernetting with VLSM gives ISPs the ability to slice and dice address spaces with surprising efficiency and facility.