CHAPTER 4

Variable-Length Subnet Masks

The preceding chapter examined the powerful innovation known as subnetting in its original form: Fixed-Length Subnet Masking (FLSM). At its introduction, FLSM was called simply subnetting. By any name, it was a revolutionary and necessary evolution of the IP address architecture that enabled a tremendous reduction in the waste of IP addresses. With the benefit of hindsight, we can see that FLSM was but a half step in the right direction. Its single greatest benefit was that it validated the concept of borrowing bits from the host field of an IP address to create locally significant subnetwork identification addresses. But the simplifying assumption of permitting just one subnet mask for all subnets created from a network address proved to be both unnecessary and inefficient.

In reality, subnets are hardly ever of the same approximate size. Consequently, FLSM's one-size-fits-all design philosophy created a substantial number of wasted addresses. Solving this conundrum was easy: Permit the creation of variable-length subnets. In theory, this would enable subnets to be created more efficiently by making each subnet mask specifically tailored to each subnet.

To help make this esoteric concept a bit more real, I'll use a sample network to show you how subnetting works mathematically. Throughout this chapter, we'll build on this sample network and look at some interesting things, including practical implications. We'll also look at the challenges of managing an address space subnetted into flexibly sized subnets using a technique known as Variable-Length Subnet Masking.

Variable-Length Subnetting in the RFCs

Ordinarily, I would point out an IETF source document that is the basis for an Internet technology and then expound on that technology. In the case of Variable-Length Subnet Masking (VLSM), there is no clear-cut genesis document. Searching the Internet or the RFC Editor's database turns up a variety of references, mostly in documents dedicated to other topics. The more salient of these tangential reference documents are RFC 1009 and RFC 1878. They provide you with the context for the development of variable-length subnets and supporting mathematics and helps you appreciate a more thorough examination of VLSM. The following sections discuss each of these RFCs.

RFC 1009

The first "official" acknowledgment that you could use multiple subnet masks came in June 1987 with RFC 1009. Although this RFC focused on requirements for Internet gateways, the perceived benefits of a flexible approach to subnetting were identified. The rationale for supporting flexible or variable subnet masks was an acknowledgment of the inefficiency of trying to use a single mask for multiple subnets. Additionally, the RFC's authors acknowledged that subnetting was strictly a local phenomenon that had no impact on global routing. Consequently, enabling subnets to be created with different sized masks within the same network address offered tremendous benefits with no disadvantages. The flexibility was deemed critical to a continued ability to cope with the Internet's future expected growth.

Here are some of the rules stipulated in this RFC:

- Not assigning subnet addresses with values of either all 1s or all 0s.
- It was recommended, but not required, that the host address's highest-order bits be used to create the subnet addresses. This ensured that both network and subnetwork addresses remained contiguous. For example, let's see what happens if the Class C-sized network address 192.168.125.0 is subnetted with a subnet mask of 255.255.255.224. Translating that address to binary results in **11000000.10101000.01111101.*000***00000. The last 8 bits are the only ones used for host addresses, so you are exhorted by the RFC to use this field's highest-order bits to create your subnet address. I have indicated those bits in bold italic to make it very clear what is meant by highest-order bits.
- The Internet would route to the subnetted location using only the network portion of the IP address. Local gateways (known more commonly today as routers) would be required to route to specific destinations using the entire extended network prefix. The *extended network prefix,* to refresh your memory, is the network address plus the subnet address. This prefix can be seen only when you view the address in binary form. Let's look at this a little closer using the same example as before. The extended network prefix for address 192.168.125.0 with a subnet mask of 255.255.255.224 is indicated in bold italic in the following bit string: ***11000000.10101000.01111101.000***00000. The local router makes its forwarding decisions based on both the network and subnetwork portions of an IP address.

Remember that because a subnet mask is of local significance only, nothing you do at the subnet level has any impact whatsoever on routing *to* your network. At the time RFC 1009 was published, many network administrators had figured out the mathematics of VLSM on their own and were not only creating variable-length subnets, but also nesting multiple levels of subnets within other subnets! Thus, the Spartan description of guidelines on how to support variable-length subnets in RFC 1009 amounted to little more than an acknowledgment of what was already becoming common practice.

The term *VLSM* is not used in RFC 1009. Instead, the RFC's authors seem to prefer describing this phenomenon as *flexible subnet masks*.

Standardization Without Ratification

It is generally understood and accepted that VLSM was an evolutionary step forward that was made possible by the successes of FLSM. However, few understand that VLSM wasn't explicitly and separately defined until long after it was in common use.

You might be wondering how on earth someone could use something like VLSM if it hasn't been formally developed or sanctioned by the IETF. That's a fair question. It certainly isn't the normal way a technology becomes accepted and deployed throughout the Internet user community. The answer is remarkably simple: VLSM didn't have to be sanctioned, because the capability already existed. We just lacked the sophistication to appreciate it. That probably sounds a little cryptic, so let me try explaining it in more-tangible terms. In a local-area network (LAN), subnet masks are configured in three general locations:

- A router's interface to a LAN
- Management ports on all LAN devices such as hubs or switches
- All the hosts connected to that LAN

In order for everything to work properly in an FLSM environment, all interfaces within a network must use the same mask. This includes all the hosts, the router interface, and the management ports on the LAN hubs or switches. But each of these interfaces is configured separately. No requirement in the FLSM specifications mandated a sanity check across all of a router's interfaces to ensure that the same-sized mask was being used. Some early routing platforms gave the administrator a caution but accepted the flawed configuration. Thus, nothing stopped you from assigning different-sized subnet masks to different router interfaces, even though each of those interfaces might have been configured with IP addresses from the same network address.

In lieu of IETF standardization, grassroots creativity allowed FLSM to deliver greater capability than its creators ever imagined possible. The IETF first acknowledged the informal practice of "flexible" subnetting in RFC 1009, released way back in June 1987. However, they didn't grant legitimacy to that practice until they started developing another technology, *Classless Interdomain Routing* (CIDR). A rather inauspicious start for an invaluable capability!

RFC 1878

RFC 1878 is an Informational RFC released in December 1995. It defined no new technology or protocol, but it offered greater clarification on the mathematical trade-offs between the number of hosts and the number of subnets that could be created with various-sized network blocks. In fact, RFC 1878 was titled "Variable Length Subnet Table for IPv4."

One of the more useful tables in this RFC is excerpted in Table 4-1. This table demonstrates the correlation between the number of subnets and hosts you can define with any given-size mask. The mask size is indicated in both the familiar decimal terms and a new notation that

was introduced in CIDR. This notation explicitly identifies the extended network prefix by using a slash (/) followed by a number. The slash can be thought of as a flag that indicates that the following numbers specify how many bits in the IPv4 address are used for network and subnetwork addresses. Thus, the number after the slash, when subtracted from 32, yields the number of bits allocated to host addressing.

Please note that all values in Table 4-1 are gross and are not adjusted for usability.

Table 4-1 *Mathematical Correlation Between Subnets and Hosts*

Decimal Mask	CIDR Notation	Number of Possible Host Addresses	Size in Terms of Class-eBased Networks
128.0.0.0	/1	2,147,483,648	128 Class A
192.0.0.0	/2	1,073,741,824	64 Class A
224.0.0.0	/3	536,870,912	32 Class A
240.0.0.0	/4	268,435,456	16 Class A
248.0.0.0	/5	134,217,728	8 Class A
252.0.0.0	/6	67,108,864	4 Class A
254.0.0.0	/7	33,554,432	2 Class A
255.0.0.0	/8	16,777,216	1 Class A
255.128.0.0	/9	8,388,608	128 Class B
255.192.0.0	/10	4,194,304	64 Class B
255.224.0.0	/11	2,097,152	32 Class B
255.240.0.0	/12	1,048,576	16 Class B
255.248.0.0	/13	524,288	8 Class B
255.252.0.0	/14	262,144	4 Class B
255.254.0.0	/15	131,072	2 Class B
255.255.0.0	/16	65,536	1 Class B
255.255.128.0	/17	32,768	128 Class C
255.255.192.0	/18	16,384	64 Class C
255.255.224.0	/19	8,192	32 Class C
255.255.240.0	/20	4,096	16 Class C
255.255.248.0	/21	2,048	8 Class C
255.255.252.0	/22	1,024	4 Class C
255.255.254.0	/23	512	2 Class C
255.255.255.0	/24	256	1 Class C

Table 4-1 *Mathematical Correlation Between Subnets and Hosts (Continued)*

Decimal Mask	CIDR Notation	Number of Possible Host Addresses	Size in Terms of Class-eBased Networks
255.255.255.128	/25	128	1/2 Class C
255.255.255.192	/26	64	1/4 Class C
255.255.255.224	/27	32	1/8 Class C
255.255.255.240	/28	16	1/16 Class C
255.255.255.248	/29	8	1/32 Class C
255.255.255.252	/30	4	1/64 Class C
255.255.255.254	/31	2	1/128 Class C
255.255.255.255	/32	1	Single-host route

Table 4-1 should exhibit some interesting mathematical patterns. If these patterns aren't yet self-evident, fear not: It gets easier with practice! Basically, you will see a repetition of a numeric sequence in the Decimal Mask column. A single bit set equal to 1 in the leftmost column of a binary octet carries a decimal value of 128. The initial 2 bits, when set to 11, yield a decimal equivalent of 192. Thus, the increments of decimal numbers must follow the pattern you first saw in Chapter 2: 128, 192, 224, 240, 248, 252, 254, and 255.

The next interesting pattern you should recognize is that, starting with the /32 mask (which references just one host address), you are doubling the number of possible host addresses with each bit you add to the host field. This doubling is complemented with a halving of the number of subnets available. Start at the bottom of the Number of Possible Host Addresses column. For every bit you add to the host address field, you double the quantity of addresses available. With each bit you remove from the host field, you are halving the number of available hosts in that sized network. Thus, a /25 network offers exactly half the number of host addresses that are available in a /24 network. Understanding this relationship is necessary for understanding both VLSM and CIDR. We'll look at CIDR in much more detail in Chapter 6, "Classless Interdomain Routing (CIDR)."

If you'd like to read more about the guidelines for VLSM contained in RFC 1878, here's the URL:

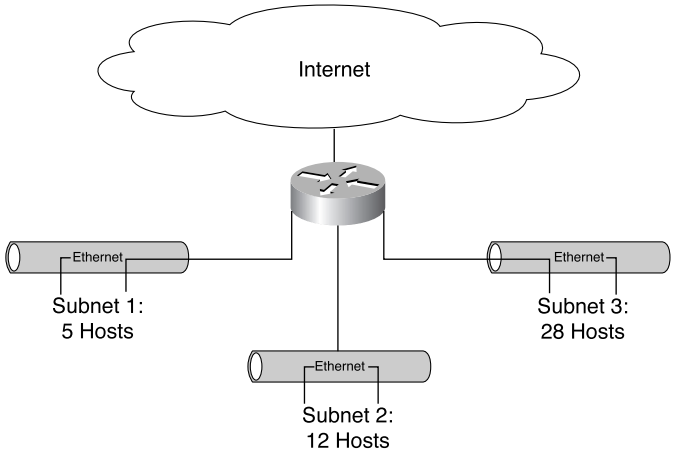
www.ietf.org/rfc/rfc1878.txt

The Inefficiencies of FLSM

Chapter 3, "Fixed-Length Subnet Masks," showed you how FLSM lets you conserve the IP address space by creating locally significant subnetwork addresses. The benefit of this is that you can use a single network address to service multiple local networks. But in the real world, those local networks are seldom the same size. Thus, implementing FLSM actually

wastes IP addresses. To better illustrate this point, consider the network shown in Figure 4-1. We will use this basic network diagram as the basis for exploring VLSM throughout this chapter.

Figure 4-1 *Sample Network*



This example is simplified throughout this chapter for the sake of demonstrating the relative efficiencies of VLSM versus FLSM. It is not intended as a realistic example, nor is it indicative of how you would actually subnet a block of network addresses. Consequently, you might notice that there are no IP addresses in the subnets assigned to LAN switch or hub management ports. Similarly, some of the subnet address ranges do not end neatly on the bit boundaries indicated by their mask. Although this scheme would work, it is not ideal. Again, it is intended solely to demonstrate relative efficiencies. It's not a guide for how to subnet on the job. We'll look at how to do that in subsequent chapters. For now, we'll just concentrate on the basics, and that requires some simplifying assumptions.

In Figure 4-1, you can see that the local subnets are of very different sizes. Subnet 1 contains just five hosts. Subnet 2 contains 12 hosts, and Subnet 3 contains 28 hosts.

Without any form of subnetting, you might have to resort to using three different Class C networks for each of these subnetworks. That would be a tremendous waste of addresses: 765 total addresses for just 45 hosts! Subnetting a traditional Class C network using fixed-length subnet masking requires you to use a mask large enough to satisfy the largest of the three subnets. That would be Subnet 3 with its 28 hosts. As you saw in Chapter 3, this would require subnetting the entire 24-bit network address with a mask of 255.255.255.224. That mask borrows 3 bits from the host field to create a 27-bit extended network prefix. The result is a 5-bit host field that lets you define six usable subnets, each with 30 assignable IP addresses. When I say "assignable," I'm referring to the reservation of both the all-0s and

all-1s addresses. There are 32 mathematically possible addresses per subnet, but you need to reserve one host address (the all-0s address) for subnet identification and the all-1s address for IP-based broadcasting within that subnet.

NOTE The notion of reserving all 0s and all 1s is found throughout the history of the Internet and the Internet's address space. Confusion results because this convention has been applied to both subnet addresses and host addresses.

As *host* addresses, these values are necessary within a subnet for subnet identification and broadcasting. Broadcasting is a transmission technique that sends the same packets of information to every connected host within a specific scope. In this particular case, the all-1s host address within any given subnet would be used to communicate with all hosts within that subnet.

This practice differs starkly from the obsolete practice of reserving all-0s and all-1s *subnet* addresses. Reserving those subnet addresses was arbitrary and was recommended solely for the sake of maintaining consistency of language within the network engineering community.

In this example, a total of 45 addresses are wasted through inflexibility. Using a mask of 255.255.255.224 to accommodate the needs of your largest subnet (Subnet 3) results in varying surpluses of addresses available in each subnet. Table 4-2 demonstrates how FLSM wastes addresses through the inflexibility of its subnetting mask.

Table 4-2 *Inefficiencies in FLSM*

Subnet Number	Number of Hosts	Excess IPs
Subnet 1	5	25
Subnet 2	12	18
Subnet 3	28	2

Using FLSM saves IP addresses in this example when compared to just giving each subnetwork its own Class C network address. Of course, you could make a very convincing argument that you would never want to precisely size a subnet, even if doing so were technically feasible. Such an effort might well result in a very painful renumbering exercise in the future when growth inevitably occurs. We'll look at how you can manage growth within a subnet later in this chapter. For now, let's look at how VLSM can improve on the efficiency with which you can use an address space.

Comparing VLSM to FLSM

VLSM greatly improves the efficiency with which the sample network can be subnetted. Instead of just using a 3-bit subnetwork field (mask of 255.255.255.224) for all fields, you can use whichever size mask makes sense for each subnet. In Chapter 3, we looked at a table that correlated the trade-offs between the number of hosts and the number of subnets that could be created from a 24-bit network address. That table, and that entire chapter, were focused on Classical IP—in other words, the way things used to be! Reserving the all-0s and all-1s subnets is no longer necessary or desirable. Consequently, the tables in this chapter reflect the availability of all mathematically possible subnet addresses.

With that caveat in mind, take a look at Table 4-3. Notice the gaps between subnet sizes? This should be a very familiar pattern by now, because it is a function of binary mathematics. Intuitively, you should recognize that VLSM won't be perfectly efficient, simply because subnet sizes (created from a 24-bit network address) increment from two usable host addresses to 6, to 14, to 30, and then to 62. We'll talk about this phenomenon later in this chapter, including why such gaps might be useful.

Table 4-3 *Hosts Versus Subnets in a Traditional Class C Network*

Number of Bits in the Network Prefix	Subnet Mask	Number of Usable Subnet Addresses	Number of Usable Hosts Per Subnet
2	255.255.255.192	4	62
3	255.255.255.224	8	30
4	255.255.255.240	16	14
5	255.255.255.248	32	6
6	255.255.255.252	64	2

Armed with this information, you can return your attention to the sample network. Looking at Table 4-2, Subnet 3 (which contains 28 hosts) would still require 5 bits for host addressing, so the best-size mask would still be 255.255.255.224. Subnet 1, however, would be better off with a mask of 255.255.255.248. That mask allocates 5 bits to the subnet identification field and 3 bits to host identification. A 3-bit host field yields six usable host addresses, which is perfect for the five-host subnet, and it still leaves an extra address for future growth. Subnet 2, with its 12 hosts, would be best-served with a mask of 255.255.255.240, because that mask evenly splits the host address into 4 bits for the subnet and 4 bits for host identification. This lets you assign 14 unique host addresses within a subnet.

Table 4-4 demonstrates the binary and decimal mathematics of the right-sized mask for each of the three subnets. Given that the first three octets are all high values of 255 or 11111111 in binary, I conserve space and save your eyesight by showing only the contents of the mask's last octet. The first three octets are indicated with the constant *n*. The binary values in this table adhere to my convention of showing the bits representing the subnet ID in a bold italic font.

Table 4-4 *Finding the Right Mask Size Per Subnet*

Subnet Number	Number of Hosts	Mask	Binary Value of Mask's Last Octet
Subnet 1	5	255.255.255.248	*n.n.n.**11111***000
Subnet 2	12	255.255.255.240	*n.n.n.**1111***0000
Subnet 3	28	255.255.255.224	*n.n.n.**111***00000

As you can see, the most efficient subnet mask for each of the three subnets differs for each network. The downside of carefully tailoring a subnet mask to a subnet is the limitation of capacity for future growth. Ideally, you want some room for future growth, but predicting how much growth will be experienced is more of an art than an exact science. For the sake of the example, you need only 3 bits for host addressing in Subnet 1, 4 bits for host addressing in Subnet 2, and 5 bits for host addressing in Subnet 3. Using VLSM lets you use those three different masks within the same network address to achieve a dramatic reduction in wasted or unusable IP addresses.

Table 4-5 compares the relative efficiency of FLSM versus VLSM in the sample network.

Table 4-5 *Comparing the Relative Efficiency of FLSM Versus VLSM*

	FLSM with a Mask of 255.255.255.224		VLSM	
Subnet Number	Number of Hosts Supported	Number of IPs Wasted	Number of Hosts Supported	Number of IPs Wasted
Subnet 1	5	25	5	1
Subnet 2	12	18	12	2
Subnet 3	28	2	28	2

Quickly summing the number of IP addresses wasted using FLSM for the sample network vis-à-vis VLSM reveals that there can be a dramatic improvement. In the sample network, FLSM requires you to use a 3-bit mask, resulting in a waste of 45 IP addresses in just three subnets. That wasted amount drops to just five with a VLSM scheme.

A Practical Application

To better demonstrate how VLSM works in practical terms, Table 4-6 shows the progression from the sample network's base address (192.168.125.0) through the defined subnets. Pay particular attention to the binary and decimal translations for each subnet's base and terminal addresses. In decimal terms, you are progressing sequentially through the address space. In binary terms, you can see that each network uses a different combination of high-order bits in the last octet to identify the subnet. This might seem strange, but it is eminently

logical. I distinguish between host and subnet bits in the binary address by indicating the subnet bits in bold italic and then delineating the two subfields with a dash (-). Ordinarily, you wouldn't find a dash in the middle of a bit string.

Table 4-6 *Subnetting with VLSM in a 24-Bit Network*

	Binary Network + Subnet Address	Decimal Translation
Base	**11000000.10101000.01111101**.00000000	192.168.125.0
Unassigned (Subnet 0)	**11000000.10101000.01111101.*000***00000	192.168.125.0
Unassigned (Subnet 0)	↓	↓
Unassigned (Subnet 0)	**11000000.10101000.01111101.*000***11111	192.168.125.31
Subnet 1	**11000000.10101000.01111101.*00100***000	192.168.125.32
Subnet 1	↓	↓
Subnet 1	**11000000.10101000.01111101.*00100***111	192.168.125.39
Subnet 2	**11000000.10101000.01111101.*0010***1000	192.168.125.40
Subnet 2	↓	↓
Subnet 2	**11000000.10101000.01111101.*0010***1111	192.168.125.47
Subnet 3	**11000000.10101000.01111101.*001***10000	192.168.125.48
Subnet 3	↓	↓
Subnet 3	**11000000.10101000.01111101.*001***11111	192.168.125.63
Unassigned	**11000000.10101000.01111101.**01000000	192.168.125.64
Unassigned	↓	↓
Unassigned	**11000000.10101000.01111101.**11111111	192.168.125.255

The unassigned space illustrated in Table 4-6 can be used in a number of different ways. Here are two of the more feasible scenarios for using this space:

- Any existing subnet can suddenly expand beyond the surplus afforded by its current mask.
- A new group of users might have to be supported, which necessitates the creation of a new subnet.

Both of these scenarios and their implications on subnetting schemes are explored in the remainder of this section.

Adding a Subnet

In the example used throughout this chapter, Subnets 1 through 3 have been used, with Subnet 0 idle. It is equal in size to a subnet defined with a mask of 255.255.255.224, but the 32 addresses it would contain are not used. I did this on purpose to show you that in many cases, old subnetting rules can be very persistent. Consequently, it is not uncommon to find Subnet 0 unused in networks today. Under today's rules for subnetting, this space doesn't have to lie fallow. To continue with the sample network, if a requirement emerges for a new subnet with 30 or fewer devices, Subnet 0 could be pressed into service.

Additional unallocated addresses remain in the high end of the network address block. Addresses 64 through 255 are unused, so it is possible for additional subnets to be created in the future. Thus, you have some options for satisfying new requests for subnets. Depending on the number of hosts in a new subnet, you could assign Subnet 0 or carve Subnet 4 out of the currently unassigned address space (addresses 64 through 255). A more nettlesome question is how you accommodate growth within the existing subnets.

Outgrowing a Subnet

As you were looking at Tables 4-4 and 4-5, did you notice that even VLSM isn't perfectly efficient? There are still wasted addresses. That's a direct function of the binary math that is the foundation of the address space itself, rather than a flaw in any particular approach to carving the space into subnets. You can create a subnet on any bit boundary, but each bit increments by a factor of 2. Remember: The rightmost bit in an octet has a decimal value of 1, the bit to the immediate left carries a value of 2, then 4, then 8, then 16, then 32, then 64, and ultimately 128 for the leftmost bit in the octet. Consequently, you must form your subnets in this sequence from powers of 2.

You could look at this architectural feature as a negative in that it results in wasted space. Alternatively, you could take a more pragmatic perspective and appreciate the positive implications of this feature. For example, even if it were possible to create subnets of the precise size you require for any given subnet, would you really want to tailor it so concisely? Many things can happen that would require you to add endpoints to a subnet. For example, the user community on any of your subnets might hire someone new. The same thing would hold true for technological innovation. It wasn't that many years ago that printers didn't have a built-in network interface card (NIC), and you had to use a server to spool print requests to them. A more commonly encountered scenario is that the group you are supporting can simply outgrow its subnet.

The point is that there are many reasons why the number of host addresses in any given subnet could change over time. Trying to add a few addresses to a tightly constructed subnetted scheme can be painful. Depending on the extent of the growth, you might find it necessary to completely renumber one or more subnets! That might not sound so bad, but it is not fun, and your users might not appreciate having to experience it either. Plus, you might discover a relatively common practice: application developers who hard-code IP addresses into their software. Renumbering a network will cause every such application to fail!

To illustrate this point, let's assume that Subnet 2 of the sample network needs to grow by five endpoints. Unfortunately, that subnet has only two available IP addresses within its assigned range (192.168.125.40 through 192.168.125.47), and Subnet 3 picks up right where Subnet 2 ends, so there is no opportunity to just change the mask's size to encompass sequential but unassigned addresses. Because Subnets 2 and 3 are numerically contiguous, your only choices both involve renumbering the endpoints within Subnet 2. You have to move them to a section of the address space that offers more addresses. Your two options are as follows:

- Move the endpoints from Subnet 2 to Subnet 0.
- Move the endpoints from Subnet 2 to the newly created Subnet 4 using previously unassigned addresses from the high end of the address block.

Table 4-7 shows you what the subnetting scheme would look like if you were to renumber the endpoints in Subnet 2 to use the range of addresses in Subnet 0. Doing so results in the allocation of 30 usable host addresses to a group of users that requires only 17, but you don't have any better options! The coarseness of the architecture works against you. A mask of 255.255.255.240 would yield only 14 usable hosts, which is inadequate. However, a mask of 255.255.255.224 (1 less bit in the subnet prefix) yields 30 usable hosts and is the only feasible solution. This should seem familiar to you, but if it doesn't, just refer back to Table 4-4.

Table 4-7 *Moving Subnet 2 Hosts to Subnet 0*

	Binary Network + Subnet Address	Decimal Translation
Base	**11000000.10101000.01111101**.00000000	192.168.125.0
Subnet 0	**11000000.10101000.01111101.*000***00000	192.168.125.0
Subnet 0	↓	↓
Subnet 0	**11000000.10101000.01111101.*000***11111	192.168.125.31
Subnet 1	**11000000.10101000.01111101.*00100***000	192.168.125.32
Subnet 1	↓	↓
Subnet 1	**11000000.10101000.01111101.*00100***111	192.168.125.39
Unassigned (formerly Subnet 2)	**11000000.10101000.01111101.**00101000	192.168.125.40
Unassigned (formerly Subnet 2)	↓	↓
Unassigned (formerly Subnet 2)	**11000000.10101000.01111101.**00101111	192.168.125.47
Subnet 3	**11000000.10101000.01111101.*001***10000	192.168.125.48

Table 4-7 *Moving Subnet 2 Hosts to Subnet 0 (Continued)*

	Binary Network + Subnet Address	Decimal Translation
Subnet 3	↓	↓
Subnet 3	**11000000.10101000.01111101.*001***11111	192.168.125.63
Unassigned	**11000000.10101000.01111101.**01000000	192.168.125.64
Unassigned	↓	↓
Unassigned	**11000000.10101000.01111101.**11111111	192.168.125.255

Your second option for satisfying the outgrowth of Subnet 2 is to create a new subnet from the unassigned addresses. Table 4-8 demonstrates how this would be done. Pay particular attention to the binary and decimal translations for Subnet 4 to see how that would be accomplished.

Table 4-8 *Moving Subnet 2 to the Newly Created Subnet 4*

	Binary Network + Subnet Address	Decimal Translation
Base	**11000000.10101000.01111101.**00000000	192.168.125.0
Unassigned (Subnet 0)	**11000000.10101000.01111101.**00000000	192.168.125.0
Unassigned (Subnet 0)	↓	↓
Unassigned (Subnet 0)	**11000000.10101000.01111101.**00011111	192.168.125.31
Subnet 1	**11000000.10101000.01111101.*00100***000	192.168.125.32
Subnet 1	↓	↓
Subnet 1	**11000000.10101000.01111101.*00100***111	192.168.125.39
Unassigned (formerly Subnet 2)	**11000000.10101000.01111101.**00101000	192.168.125.40
Unassigned (formerly Subnet 2)	↓	↓
Unassigned (formerly Subnet 2)	**11000000.10101000.01111101.**00101111	192.168.125.47
Subnet 3	**11000000.10101000.01111101.*001***10000	192.168.125.48

continues

Table 4-8 *Moving Subnet 2 to the Newly Created Subnet 4 (Continued)*

	Binary Network + Subnet Address	Decimal Translation
Subnet 3	↓	↓
Subnet 3	**11000000.10101000.01111101.*001***11111	192.168.125.63
Subnet 4	**11000000.10101000.01111101.*010***00000	192.168.125.64
Subnet 4	↓	↓
Subnet 4	**11000000.10101000.01111101.*010***11111	192.168.125.95
Unassigned	**11000000.10101000.01111101.**01100000	192.168.125.96
Unassigned	↓	↓
Unassigned	**11000000.10101000.01111101.**11111111	192.168.125.255

If you look carefully at the progression of numbers—particularly the binary numbers—you will see a very familiar pattern. Both Subnets 3 and 4 use a 3-bit subnet mask. This makes it easy to see why one subnet ends with ***001*11111** and another begins with ***010*00000**. In the other cases, mismatched subnet prefixes make it a bit tougher to follow the logic. In those cases, ignore my visual cues about subnet formation and just look at the 8-bit string as a whole. Then, things make more sense.

Keep Careful Records!

Looking back over the previous three tables, you must draw one simple, inescapable conclusion: It is critical to maintain an accurate and up-to-date record of address assignments! Subnetting, in its original incarnation, was deemed practical only if you used a fixed-length mask to subdivide an entire network address. Although this wasn't the most efficient way to subdivide a network, it was still far more efficient than the previous method of operation, which was to secure separate network addresses for each of your subnets.

The grassroots innovation of flexible subnetting happened outside the auspices of the IETF. Thus, there was no solid base of research to draw on, no well-worn trail to follow, and no set of tools to rely on. If you chose to ignore the recommendation of using a single-sized subnet mask for your subnetting, you were on your own! This was a simplifying assumption embedded in the original subnetting RFCs. It gave ordinary network administrators a chance to improve the efficiency with which they consumed IP address space without creating a mathematics exercise that could have qualified as a Herculean task.

Despite the risks of stepping outside the safe confines of an RFC, flexible subnetting became the dominant paradigm. Technical personnel realized that the price they had to pay for this was the creation and maintenance of an accurate database of address assignments. This has never been more true!

Summary

Despite its beginning as an illegitimate child of an IETF ratified technology, "flexible subnetting" proved itself vastly superior to the original form of subnetting with fixed-length masks. In and of itself, VLSM (as it later became known) represented a tremendous advance in the sophistication of the IP address space. Yet it would have an even greater contribution in the future—a contribution that no one could have foreseen. You see, in the mid-1990s, the Internet was experiencing unprecedented, phenomenal growth. This growth rate sorely tested the scalability of the Internet's mechanisms, including its address space.

Sadly, the IP address space demonstrated that its original architecture was not up to the task. The IETF sprang into action, launching numerous working groups to evaluate how best to shore up the failing address space. Numerous efforts were focused on developing stopgap fixes to shore up the sagging address space. These fixes are outlined and described in the next chapter. One notable development was Classless Interdomain Routing (CIDR). CIDR was made possible by having VLSM break the psychological barrier of variable-length network prefixes. CIDR is an important-enough aspect of the IP address space to warrant its own chapter. We will explore it in detail in Chapter 6.