**5. IP Addresses, Subnet Masks, and Subnetting**

There are two sets of rules for subnetting TCP/IP networks. The original set of rules can be found in [RFC 950](http://www.google.com/search?q=RFC+950&btnI=I'm+Feeling+Lucky), and the new set of rules can be found in [RFC 1812](http://www.google.com/search?q=RFC+1812&btnI=I'm+Feeling+Lucky).  
  
<RANT>   
Although RFC 1812 came out in June of 1995(!), most certification tests still test you on the RFC 950 rules, for (in my opinion) one of the following reasons:

* Their software still follows RFC 950 rules (this is rare.)
* Since RFC 1812 simplifies things significantly, there's not enough material to test on. Test items from RFC 950 are added as "filler".
* They are ignorant of the fact that the material on their tests has been out of date for more than five years.
* They are mean-spirited, [perniciously](http://www.dictionary.com/cgi-bin/dict.pl?term=perniciously) forcing you to learn material that will never be relevant to your job.

Please keep the fact that the following information in Part A is no longer relevant to the real world; however, it may be necessary to understand it if:

* You are planning on taking one of the aforementioned tests; or
* You need to communicate with someone who holds certain certifications, and believes everything they were tested on still has some relationship to the way the 'Net works.

I still get many questions to the effect of, "I don't understand. This source says I can break this Class C into six subnets, but this other source says you can break the same network into eight subnets. What gives?" The short answer is, it depends on which RFC is valid in your environment. If you are still running an unpatched Netware 3.11 server, you will find yourself constrained by RFC 950 rules. However, a patch has been available for that 1991 platform since sometime around 1995; if you are still running version 1.00 of the Netware TCPIP.NLM, then IP Routing issues are the least of your concerns. :-)  
</RANT> 

**Part A: The World According to RFC 950 (the old way of doing things)**

An IP Address is broken up into three parts: the network portion, the subnet portion (optional), and the host portion. The size of the network portion is determined by the first byte of the address:

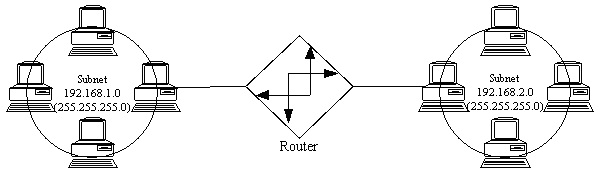
|  |  |  |
| --- | --- | --- |
| First Byte | Class | Network Mask (explained later) |
| 1-126 | "A" | 255.0.0.0 |
| 128-191 | "B" | 255.255.0.0 |
| 192-223 | "C" | 255.255.255.0 |

Note: people often refer to any subnet with a mask of 255.255.255.0 as being a class "C" network; however, the only "true" class "C" networks have a first byte in the range of 192-223. This becomes important when you start subnetting.

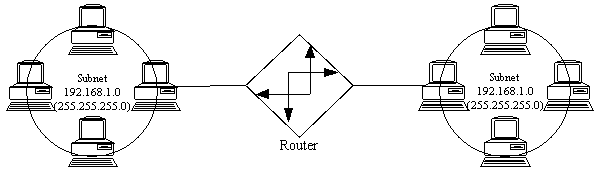
The Subnet portion of an IP address is actually optional, and, in fact, is rarely used on class "C" networks. Generally, you can subnet any network you have control over, in any valid way you want. The tricky part is understanding what is valid.  
Lets start with some ground rules:

* All hosts on the same subnet must agree on the subnet mask, particularly the routers. Otherwise, packets actually intended for another subnet may never leave the existing subnet: a host won't give to the router a packet it thinks is destined for the local segment. This behavior is important to understand: the router doesn't automatically forward packets, the hosts have to actually *give* the packets *to* the router.
* No two different subnets can include the same host address. This can get tricky when subnetting in an unusual manner.
* The top and bottom host numbers are reserved; the bottom one (usually ?.?.?.0) is shorthand for the whole subnet, and the top one (usually ?.?.?.255) is the broadcast address. Some implementations also use .0 as a broadcast address, so it is never safe to use for a host.
* The bits in the subnet portion cannot be all ones. This requires a bit of binary arithmetic to determine which subnets would be invalid

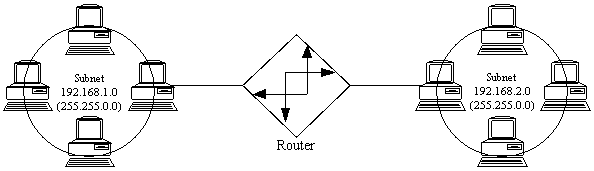
**Valid Configuration:**



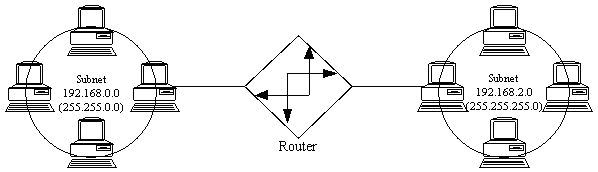
**Invalid Configurations:**



...This is invalid since the [exact] same subnet exists on both sides of the router.



...This is invalid since the same subnet exists on both sides of the router. Watch that subnet mask! (See below.)

  
*These images created using SmartDraw.*[*Click Here*](http://www.smartdraw.com/resources/centers/networkdesign/index.htm)*for a free trial copy.*

...This is invalid because a the same host address could be "valid" on either subnet, e.g. 192.168.2.100. Even though the right side subnet is valid by itself, it is actually a small piece of the left side network.

|  |  |
| --- | --- |
| **Exception!** | Address overlap of this sort is *usually* not allowed between two physical subnets: unless the router was specifically configured to "pretend" it was every address on 192.168.2.0 for its left-side interface in the diagram, it would be impossible for hosts on one side of the router to communicate with hosts on the other side. In this diagram, the 192.168.2.0 subnet is known as a "stub subnet"; the process of pretending you are hosts you're not, in order to facilitate routing packets to a stub subnet, is known as "proxy arp." No two hosts on the Internet can have the same IP address. If you create a stub subnet, no host on the "main" side can have an address that might be valid on the "stub" side. [Please also note that the diagram in question is talking about two physical subnets attached to one router, not routing tables on upstream routers, which would aggregate both networks into one route of 192.168.0.0/16.] |

The Glossy Explanation

When using a subnet mask of 255.255.0.0, the first two bytes indicate the network you're on, and the last *two* bytes indicate the host you are on that network. Very rarely will you find a network segment with 65,534 hosts on it, though. You'll only find network masking like that used closer to the Internet backbone, in the context of, "All them hosts [and subnets thereof] are thataway." Now, that brings up one of the nice features of subnet masking: you can lump a bunch of networks together by using unusual subnet masking; however, that sort of activity generally doesn't happen on the near side of the 'net.

When using a subnet mask of 255.255.255.0, the first three bytes indicate the network you're on, and the last byte is the host you are on that network. Hosts .1 through .254 are available.

By using a subnet mask of 255.255.255.128, you can split that network into two halves, the first half containing the host addresses .1 through .126, the second half containing the host addresses .129 through .254. Note that on a true class "C" network, you can't use the top subnet, since the bit in the subnet portion (one bit on a class "C") would be one (refer to ground rule "D".)

By using a subnet mask of 255.255.255.192, you can split the network into four portions, each with 64 hosts (62 usable.) Subnetwork one includes the addresses .1 through .62, subnetwork two includes the addresses .65 through .126, subnetwork three includes .129 through .190, and subnetwork four includes the hosts .193 through .254. On a true class "C" network, subnetwork four is not valid.

You can not arbitrarily cut a piece out of one network and place it on another segment; the best you can do with a given subnet (or network) is chop it in halves, or quarters, or eighths, or sixteenths... (note the "powers of two" progression; this is an effect of stealing bit positions from the host address section, and giving those bits positions to the subnet portions. It gets complicated...)

**Part B: The World According to RFC 1812 (the "new" way of doing things)**

or, By The Way - Forget Everything You Just Learned, It Became Obsolete in 1995

Under RFC 1812, things have changed..!

Perhaps the most significant change on the near side of the 'net under RFC 1812 is Classless Inter-Domain Routing (CIDR, pronounced "Cider"). Under CIDR, the concept of separate "network" and "subnet" portions is now considered outdated, and is being replaced by a "classless" addressing scheme where addresses can be "subnetted" more freely, without consideration of the "class" of address. With the removal of the subnet portion, and the liberalization of (what is now called) the network prefix, there is no longer a consideration of whether or not the bits within the subnet portion are all ones; in other words, you no longer lose a subnet when you break up what used to be known as a class "C" network. You can also aggregate formerly class "C" networks together using network prefixes fewer than 24 bits long. For example, you could combine the formerly class "C" networks 192.168.2.0 and 192.168.3.0 into a single subnet with 510 usable addresses, by using a network mask of 255.255.254.0. What you're really saying here is that the last bit of the third byte now belongs to the "host number" portion of the address, and the "network prefix" is 23 bits (two bytes and seven bits) long. Therefore, the two networks being combined must be contiguous, and the third byte must be even on the lower numbered network. You could *not* combine, for example, 192.168.2.0 and 192.168.5.0; not could you combine 192.168.11.0 and 192.168.12.0. You could follow similar rules to combine four contiguous class "C" style networks, but the third byte of the lowest numbered network would have to be a multiple of four. This sort of thing is routinely done (on an increasingly larger scale) as you get closer to the Internet backbones.

Most of the other effects of RFC 1812 and CIDR routing affect areas of the 'net closer to the backbone, and mostly work to reduce the size (or at least the rate of growth) of routing tables in backbone routers.

**Part C: Huh? (or, Perhaps you could apply an analogy to all this?)**

A good analogy for IP addressing and packet forwarding (routing) is the snail mail analogy. Consider an IP packet to be an envelope containing data, and having an address on the front. Every TCP/IP-enabled network interface can be compared to a mailbox. Every mailbox (interface) has an IP address. The four bytes of an IP address can be compared to the state, city, street, and house number fields on the front of a snail mail envelope. A router in this analogy is a post office, that sorts and forwards mail based on the address on the envelope (packet header.) If the address is on the same street (based on the subnet mask,) the envelope (packet) is sent directly to the destination mailbox (interface) via local courier (Ethernet?). If the address is determined to be on another street, or in another city or state, the envelope (packet) is delivered via local courier (Ethernet?) to the street's post office (router), where the postal workers (routing software) sort and forward mail based on established post office sorting procedures (routing tables.) The breakdown in this analogy, of course, is that no routing software has ever been known to shoot people. (Just Kidding :-)

## 6. Subnetting, Bit by Bit

### A. Binary arithmetic

You may have heard that computers represent all numbers as "bits", or "zeros and ones." It would be more fair to say that computers work primarily with groups of eight 0's or 1's, called bytes. In practice, most desktop PC's work with clumps of four bytes at a time, or 32 bits. That's why 80386 through Pentium IV processors are called 32-bit processors. [Athough Pentium class processors have some 64-bit attributes such as a 64-bit external memory bus,they still do most operations as 32-bit operations.]

Now, think back to first grade math, when the teacher was describing the decimal numbering system. As it happens, it's called "decimal" (the root of the word is from Latin *decima*, a tenth part or tithe) because it's a numbering system that uses ten numbers: the numbers zero through nine. If you need to represent a number larger than nine, you have to start adding additional digits; then the teacher described the ones place, the tens place, the hundreds place, etc. For example, the number 45678 has a four in the "ten thousands" place, a five in the "thousands" place, a six in the "hundreds" place, a seven in the "tens" place, and a 8 in the "ones" place:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ten Thousands | Thousands | Hundreds | Tens | Ones |
| 4 | 5 | 6 | 7 | 8 |

Since computers work in binary, and only have "0" and "1" to work with, they have to start new digits ("binary places", not "decimal places") as soon as they get past the number one! In decimal, the "decimal places" were all powers of ten:  
100=1,  
101=10,  
102=100,  
103=1000, etc.  
In binary, the "binary places" follow powers of two:  
20=1 (1 binary),  
21=2 (10 binary),  
22=4 (100 binary),  
23=8 (1000 binary),  
24=16 (10000 binary),  
25=32 (100000 binary),  
26=64 (1000000 binary),  
27=128 (10000000 binary),  
28=256 (100000000 binary), etc.  
  
The number 45678 is represented in binary as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Binary Places, expressed as Decimal:) | 32768 | 16384 | 8192 | 4096 | 2048 | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|  | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

(Add up the columns where you find ones: 32768 plus 8192 plus 4096 plus 512 plus 64 plus 32 plus 8 plus 4 plus 2 equals 45678!)  
Counting to Forty:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decimal** | **Binary** |  | **Decimal** | **Binary** |  | **Decimal** | **Binary** |  | **Decimal** | **Binary** |
| 1 | 1 | 11 | 1011 | 21 | 10101 | 31 | 11111 |
| 2 | 10 | 12 | 1100 | 22 | 10110 | 32 | 100000 |
| 3 | 11 | 13 | 1101 | 23 | 10111 | 33 | 100001 |
| 4 | 100 | 14 | 1110 | 24 | 11000 | 34 | 100010 |
| 5 | 101 | 15 | 1111 | 25 | 11001 | 35 | 100011 |
| 6 | 110 | 16 | 10000 | 26 | 11010 | 36 | 100100 |
| 7 | 111 | 17 | 10001 | 27 | 11011 | 37 | 100101 |
| 8 | 1000 | 18 | 10010 | 28 | 11100 | 38 | 100110 |
| 9 | 1001 | 19 | 10011 | 29 | 11101 | 39 | 100111 |
| 10 | 1010 | 20 | 10100 | 30 | 11110 | 40 | 101000 |

Now, an IP Address is four bytes, eight bits each, represented as decimal numbers with periods in between; for example, 10.5.72.230. This number can be represented in binary (remember when I said that IP Addresses are best expresses as 32-bit binary numbers? I *did* mention that, didn't I?) as b00001010.00000101.01001000.11100110. (The "b" means "binary"; that and the periods are added for your convenience.) Now, 232 (two to the thirty-second power) is 4294967296, or just over four billion. So, theoretically, there are over four billion IP addresses available to the world; so why is there a shortage? (Oh yeah, have you heard? There's a shortage. Last I checked, they're projecting to run out of IP addresses around the year 2025.) Well, as it turns out, trying to keep track of where four billion individual hosts are would be pretty much impossible for equipment today, and certainly impossible for equipment many years ago when TCP/IP routing was being developed. So, routing was (over)simplified by splitting the IP address space into "classes"; those IP addresses whose first byte was in the range 1-126 would belong to networks of 16,777,214 (224-2) hosts; these were called "Class A" networks, and there are 127 of them. In Class A networks, the first eight bits are the "network portion", and the last 24 bits are the "host portion." Those IP addresses whose first byte was in the range 128-191 were called "Class B" networks of 65,534 (216-2) hosts, and there were 16,384 (that's (192-128)\*256) of them. That's 16 bits for the network portion, and 16 bits for the host portion. "Class C" networks, where the first byte is in the range 192-223, have a 24 bit network portion, and an 8 bit host portion. Note how neatly everything lines up on byte boundaries:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Network bits** | **Network Mask** | **Network Mask (binary)** |
| A | 8 | 255.0.0.0 | b11111111.00000000.00000000.00000000 |
| B | 16 | 255.255.0.0 | b11111111.11111111.00000000.00000000 |
| C | 24 | 255.255.255.0 | b11111111.11111111.11111111.00000000 |

Now, since it's unlikely that a network administratior is going to want to have some 16,777,214 (nearly seventeen million) hosts on the same network segment(!), network administators were allowed to administratively split up their networks by subnetting them. Routing on the Internet backbones was fairly simple... until they started to hit the Class C networks hard. If your company needed 1,000 IP addresses, you'd probably get four Class C networks to accomodate them... but that would add four individual routes propagated to every "backbone" router on the Internet! Hence the need to split up networks on other than just byte boundaries.  
  
This is where everything got hard.  
  
It turns out that you can combine four "Class C" networks together into one routing table entry by using a subnet mask (aka Network Prefix) of 255.255.252.0. But not just any four; as it happens, they must be contiguous, and the third byte of the first network must be a multiple of four (like the number 204 is.) If you want to join eight of them together, the first network must be a multiple of eight (which the number 204 is not.) If you want to join ten networks together... well, you can't. Ten is not a power of two. Funny how everything follows powers of two...

### B. Boolean Logic and The Binary "AND"

Named after the nineteenth-century mathematician George Boole, Boolean logic is a form of algebra in which all values are reduced to either TRUE (1) or FALSE (0). All math performed by modern computers is done using Boolean algebra. A few basic operations:

|  |  |  |
| --- | --- | --- |
| **Operation** | **Result** | **Examples** |
| AND | true if A *AND* B are true | 1 AND 1 = 1 1 AND 0 = 0 0 AND 1 = 0 0 AND 0 = 0 |
| OR | true if A *OR* B are true | 1 OR 1 = 1 1 OR 0 = 1 0 OR 1 = 1 0 OR 0 = 0 |
| XOR (eXclusive Or) | true if *either* A *or* B are true | 1 XOR 1 = 0 1 XOR 0 = 1 0 XOR 1 = 1 0 XOR 0 = 0 |
| NOT | opposite of A | NOT 1 = 0 NOT 0 = 1 |

The binary "and" operation is often used when you want to see only certain bits of a given byte-- a procedure called "masking." Some of you may have seen a similar thing in school; some of my teachers used to conduct multiple-choice tests where you would fill in a circle cooresponding to the answer you thought was correct. The teacher would then take an overlay, or mask, and place it over the answer sheet. This overlay had holes only where the marking spots for the correct answers were, and the teacher would mark any answers where he/she didn't see a mark, as incorrect. The subnet mask is used in this fashion by the computer to determine which address bits are in the network portion of an IP address, and which bits are used for the host, or workstation, portion.

### C. The Subnet "Mask"

The subnet mask is used to figure out what network you're on. The reason it's called a "mask" is the same reason the tape you use to cover trim when painting is called "masking tape"; you use it to cover up the parts you don't want to deal with right now. Did you notice how, in a binary AND, any time B is zero, the result is zero? And any time B is one, the result is whatever A is? Hmmm.....  
  
The primary use of the subnet mask (from our perspective at the Near Side of the 'Net) is for workstations to determine whether or not the server or workstation they're trying to talk to (the "destination IP address") is on the same subnet as itself; if the destination IP address is on your subnet, you'll send the IP packet directly to the other computer via the Ethernet or Token Ring (or whatever) network you're on, without bothering the router... at all! **The first routing decision made on an IP packet is made by the workstation sending it; it decides whether or not to send the packet to a router.** Doing this is a four step process:

1. **Step 1:** Convert the IP Addresses to Binary.  
   If necessary, the IP address is converted from the familiar dotted-decimal into a 32-bit binary value. It sucks as much for the computer to do it as it does for humans to do it, but computers generally complain less, and they're good at math :-)
2. **Step 2:** Apply Source subnet mask to Source addresses:  
   The **network portion** of the **workstation's IP address** is determined by performing a binary AND operation on the workstation's IP address and its subnet mask. This operation "masks off" all of the bits of the "host portion" of the IP address, and leaves the "network portion" behind for comparison with the destination's network portion. Hey, wait a minute? How do we know what the subnet mask of the destination is?
3. **Step 3:** Apply **Source** subnet mask to **Destination** addresses:  
   As it happens, we don't care what the subnet mask of the destination is. We only care if the destination is on our same network segment! **Since every workstation on our network segment shares the same subnet mask, we can apply our subnet mask to the destination to determine if its network portion matches ours.** So, **the network portion of the destination workstation's IP address that we can use to see if it matches ours** is determined by performing a binary AND operation on the destination IP address and **our subnet mask**.
4. **Step 4:** Compare the derived **network** portions for equality:  
   At this point, we can compare the network portions we have masked from the source and destination IP addresses to see if they're the same. If they are, then we **must be on the same subnet** so we send the packet directly; if they are different, even by only one bit, the destination is on another network segment...somewhere. We don't know where. Maybe the router does...

OK, so let's try this a few times ourselves; get a few IP addresses and subnet masks together and plug 'em into Daryl's Subnet Calculator! (The next section of the Primer.) Requires JavaScript to be enabled on your browser. If you're reading a hard copy of this, the full URL is http://ipprimer.windsorcs.com/subnet.cfm .  
  
Remember the part about combining four "Class C" networks together? Watch your binary arithmetic:   
*(network prefix bits shown in green)*

|  |  |
| --- | --- |
| **Networks** | **Networks, in Binary** |
| 192.168.8.0 | b11000000.10101000.00001000.00000000 |
| 192.168.9.0 | b11000000.10101000.00001001.00000000 |
| 192.168.10.0 | b11000000.10101000.00001010.00000000 |
| 192.168.11.0 | b11000000.10101000.00001011.00000000 |
| Mask, 255.255.252.0 | b11111111.11111111.11111100.00000000 |

Notice how all of the bits above the ones in the subnet mask stay the same; following the rules above, all hosts on these networks, if you apply the mask, are on the same network. This was called "supernetting", but now is called "CIDR Routing", pronounced "Cider Routing".  
  
Doing it wrong:   
*(carefully watch the network-portion bit in****red****)*

|  |  |
| --- | --- |
| **Networks** | **Networks, in Binary** |
| 192.168.10.0 | b11000000.10101000.00001**0**10.00000000 |
| 192.168.11.0 | b11000000.10101000.00001**0**11.00000000 |
| 192.168.12.0 | b11000000.10101000.00001**1**00.00000000 |
| 192.168.13.0 | b11000000.10101000.00001**1**01.00000000 |
| Mask, 255.255.252.0 | b11111111.11111111.11111**1**00.00000000 |

Oops-- seems the sixth bit of the third byte changed within the network prefix portion (the part above the 1's in the subnet mask), so with the given subnet mask (22 bits, or 255.255.252.0), 10.0 and 11.0 would ALWAYS be on a different network aggregation than networks 12.0 and 13.0. Confused? Play with it in the [Subnet Calculator](http://www.ipprimer.com/subnet.cfm), and compare the network portions.

### D. "Slash" Notation

Subnet masks are often abbreviated using a forward slash "/" and the number of "one" bits in the mask. For example, a network 192.168.1.0 with a subnet mask of 255.255.255.0 can be expressed as 192.168.1.0/24 (since 255.255.255.0 is 24 binary ones followed by eight binary zeros.) Therefore, a /25 subnet is a subnet with a mask of 255.255.255.128, and a /26 subnet has a mask of 255.255.255.192, etc.

### E. A Neat Trick

Now that you actually understand the binary arithmetic behind subnet masking (well, I hope you do, anyway) we can cover some of the neat tricks for computing subnet masks. To determine the number of hosts on a given subnet (assuming the subnet is smaller than class "C",) simply subtract the last number of the subnet mask from 256. For example, a subnet mask of 255.255.255.224 has 32 hosts (256-224=32.) Then you can just divide the result into 256 to determine the number of subnets (256/32=8.) So, using a subnet mask of 255.255.255.224 gives you 8 subnets of 32 hosts each. Of course, this only works when you are subtracting a number that is a power of two (1, 2, 4, 8, 16, 32, 64, or 128.) When the network prefix is larger than class "C", you can determine how many class "C" networks are aggregated by subtracting the third byte from 256-- so a network prefix of 255.255.240.0 is an aggregation of (256-240) 16 class "C" networks.  
*Thanks to Gael M. for this tip.*

### F. In closing...

Why all this crap about binary arithmetic? Do I *have* to know this stuff? I'm afraid so; subnet masks are created and used on a bit-by-bit basis; in order to effectively use subnet masks that *don't* fall on byte boundaries (like 255.255.255.0 does), you have to determine what hosts are on each subnet by using binary arithmetic. It sucks, it's hard, it's confusing (espically since IP addresses and masks are expressed in decimal instead of hexadecimal notation) but you must use and understand IP addresses and subnet masks as binary.