

# Sterler Subnetting System

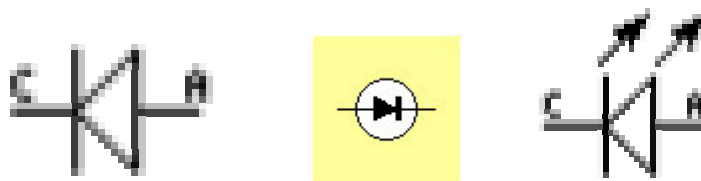
The following paper is presented as a logic system to explain the mechanics of IP addressing, subnet masking and CIDR notation. I call it the Sterler Subnetting System. I have developed this system over a period of years while teaching TCP/IP to Network administrators or those aspiring to be network administrators. It is not the only way to understand IP addressing and subnetting. Rather, it is “one” approach to understanding. IP addressing and subnetting is an elusive concepts to learn and retain for most people. My hope is that you will find this a digestible explanation of how IP works. I have had great success getting this methodology across in the classroom. Literally hundreds of students have profited from my system. However, let me give a word of caution: IP addressing and subnetting is not easy for most people. It takes due diligence. In other words you must work hard to understand subnetting. If, after thorough concentration and effort, a concept is not clear in this paper, please email me at [robert@sterler.com](mailto:robert@sterler.com) and I will try to bridge the gap. So, without further prologue, let's get started.

## Binary math

TCP/IP is based upon binary arithmetic. Binary means a base 2 numbering system. Our common usage numbering system is base 10. With our currency we have 10 digits – zero through nine to work with. A base 2 numbering system has two conditions - on or off; one or zero. The best model for demonstration purposes of a base 2 numbering system is the simple light switch. The light is either on or off. There are two conditions for a simple switch – no more and no less. The two conditions of a simple switch – open or closed. When the switch is closed the current can flow and a light is on. When the switch is open, no current can flow. A switch is a very digital device – it is either on or off.



A computer is a very digital device, just like a switch. As a matter of fact a computer is literally made up of millions of switches. The basic component of modern electronics is the transistor. A transistor is made up from a few diodes. A diode is a switch. Here are the electrical symbols of some simple diodes; there are a lot of different types:



The one with arrows pointing outward is an LED (light emitting diode). When the voltage on the anode (positive side marked A) is higher than the voltage on the Cathode (negative side marked C) – a light will shine or emit. What we say about a diode is that it is either forward biased or reversed biased. If it is forward biased the voltage is in the proper polarity and magnitude, so, that current can flow. It is on, just like a closed switch and has a logic of one; in the case of the LED it will shine. If it is reversed biased, the voltage polarity is applied to the opposite end of the diode, so that current cannot flow. In other words it is switched off and has a logic of zero.

Diodes go to make up transistors. Transistors are what CPU's; logic gates and shift registers are made from. Diodes are the basic building blocks of a computer and all digital electronics. A computer processes Bits in the form of ones and zeros. The combinations of ones and zeros are how we represent useful information. The

diode and/or switch is the physical mechanism which helps us derive useful Bits of information from ones and zero electrical states. All of the information in computers is represented by ones and zeros.

## Bits and Bytes

A Bit is a single piece of data; either a one or a zero. A byte is eight Bits grouped together. So we can use bytes as a table for representing letter, numbers and other symbols. For example in ASCII (American National Standard Code for Information Interchange) the capital letter “A” is represented by the number 65. So a byte of information in binary notation would look like:

**01000001**

So when a computer wants to represent a capital letter “A” it uses the byte with a numerical value of 65. A good link to the complete ASCII table at the following location: <http://www.ascii.cl/htmlcodes.htm>

## Powers of 2

Now we can ask: “how is the number 65 represented by this group of 8 Bits or one byte?” The answer is very logical. The position of each Bit has a numerical weight. The right most Bit has a weight of one; the next a weight of two, 4, 8, 16, 32, 64, 128. Since we are dealing in binary, the first Bit is mathematically represented as two to the zero power. Any number to the zero power equals one. This is a mathematical axiom or constant. **This constant ( $2^0 = 1$ ) is the key to understanding binary arithmetic and TCP/IP subnetting.** The next number over to the left is two to the power of one ( $2^1 = 2$ ); which equals two. The next is two to the power of two ( $2^2 = 4$ ), or two squared, which is four. A table of the powers of two looks like this:

$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32
$2^6$	64
$2^7$	128

One of the most significant features of the exponential “powers of two” is that  $2^0$  is the only odd number in a table that continues infinitely. All the other numbers of the power of 2 are even. **In addition, all the numbers double in value as we sequentially increment.** If one remembers the mathematical constant that 2 to the power of zero, equals, one. All one has to do to reconstruct the table is double the answer and increment the exponent. The value of the next exponent of the power of 2 is always double the previous number. So when we look at the Bit pattern of a byte, it looks like this:

128	64	32	16	8	4	2	1
00000000							
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

The Bit at the far right is  $2^0$ , which is always 1. The next Bit to the left is  $2^1$ , which is 2, doubles the value of one. Each successive Bit value is double the Bit value before it. Now to understand the how an ASCII character of capital “A” has a look of 01000001 and a numerical value of 65, we analyze it like this:

128	64	32	16	8	4	2	1
01000001							
$64 + 1 = 65$							

Any number between zero and 255 can be represented with eight Bits. Understanding the Bit pattern in these eight Bits is the key to binary math in TCP/IP and subnet masking. Interpreting the decimal equivalent of any eight Bits is accomplished by knowing the principles of the “powers of two”. **The key to understanding the powers of two is knowing that 2 to the zero power equals one.** Then just increment the power and double the answer. Lets look at another character – the copyright symbol © which is represented by ASCII decimal notation of 169.

128	64	32	16	8	4	2	1
10101001							
$128+32+8+1 = 169$							

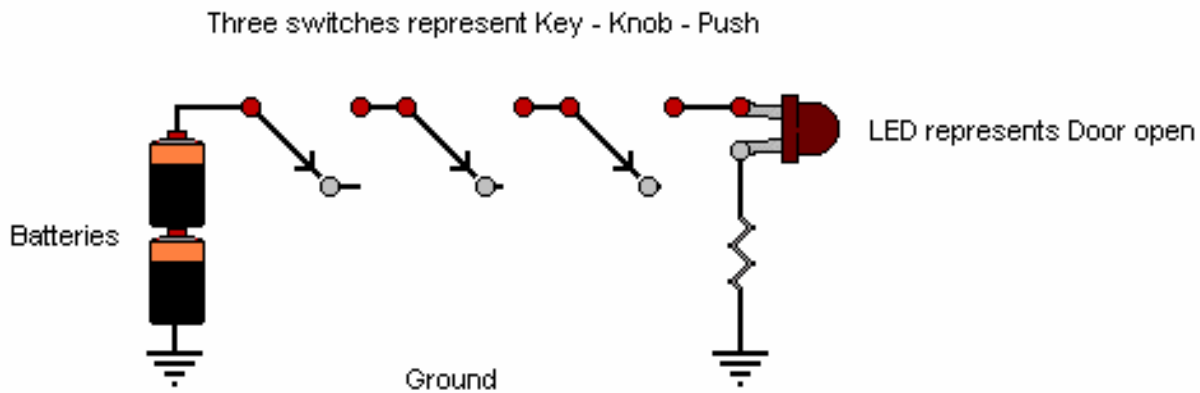
Each number one represents the place value of the binary number. In the example above we have a 1, 8, 32 and 128 for a total value of 169. So, when the computer sees a byte with a binary value of 10101001 or 169; it knows that this means ASCII for the copyright symbol ©.

# Logic or gates

The computer only knows ones and zeros, but, it can do an infinite number (theoretically) of different calculations, just using ones and zeros. It can analyze different types of logic sequences using binary math and logic gates. Logic gates are transistors (made from diodes) that are put together in various configurations that require a sequence of events to happen in order to produce a desired output. For example, if you want to go inside your house by the front door; you will need to do some certain things. Put the key in the lock and turn; turn the door knob; and push. Now in this example I am specifying three things or tasks – key – knob – push. This problem of opening the door could be represented by a logic gate. An AND gate to be exact, and, it's electrical symbol would look like the following:



Here we have three inputs (key – knob – push) that are required for an output (door open). We can also show the logic of this 3 input switch more mechanically or graphically by the flowing diagram:



One of the important points to glean from these logic diagrams is the possibilities or permutations on the inputs and outputs. Each input has two possible conditions – on or off. This is base two or binary. With three inputs the equation becomes  $2^3 = 8$ . Let's look at this in table format for more clarity:

Switch # 1	Switch # 2	Switch # 3	LED condition
0	0	0	off
0	0	1	off
0	1	0	off
0	1	1	off
1	0	0	off
1	0	1	off
1	1	0	off
1	1	1	ON

Of the eight possible switch combinations only one produces an output on the LED or Door. These graphical and mathematical pictures are preparation to understand the number of Hosts in a subnet or the number of

subnets in a mask. There are many different types of logic gates – AND, OR, NAND, NOR, XOR and they can be combined in intricate logic permutations.

## IP addresses

Let's take a closer look at what a IP address means and how it uses binary arithmetic. IP addresses are 32 Bits that are divided into 4 octets of 8 Bits each. 4 times 8 equals 32. Each Bit has two possibilities, thus, base 2 or binary. So, two to the 32 power ( $2^{32}$ ) equals 4,294,967,296. 4 Billion plus is the number of total IP addresses in version # 4 of TCP/IP. What this means is that between all zeros and all ones (in an IP address) there are 4 billion plus possible combinations. Think about this visually and count 0, 1, 2 3, etc:

00000000.00000000.00000000.00000000  
00000000.00000000.00000000.00000001  
00000000.00000000.00000000.00000010  
00000000.00000000.00000000.00000011

And so on to 4,294,967,296 at all ones

11111111.11111111.11111111.11111111

We really don't deal with all 32 Bits at one time in IP addressing. We analyze each octet (8 Bits) individually. A subnet mask is also not usually written in binary either because it is too cumbersome. We convert each octet to decimal notation (base ten), for example all ones:

11111111.11111111.11111111.11111111

Equals

255.255.255.255

Computers speak binary, humans speak decimal. If you have an IP address of

206.13.145.123

It is a unique number among 4 billion plus possibilities. And in binary would look like this:

11001110.00001101.10010001.01111011

Remember the place value of each octet in order to determine the decimal equivalent of a binary number:

128 64 32 16 8 4 2 1
10010001
$128+16+1 = 145$

## The ABC's of Class Addresses

IP addresses come in 5 classes or ranges of numbers and one special class:

- A is from 1 to 126
- B is from 128 to 191
- C is from 192 to 223
- D is from 224 to 239
- E is from 240 to 255
- And the special class is 127 or the loopback address.

All class identification is done in the left most octet by analyzing the binary Bit pattern and converting to decimal. The following table illustrates the Bit pattern for each class of address:

CLASS	high order Bits	First octet range	Provided Hosts
A	0xxxxxxx	1-126	16,777,216
B	10xxxxxx	128-191	65,536
C	110xxxxx	192-223	256
D	1110xxxx	224-239	N/A
E	1111xxxx	240-255	N/A
Loopback	01111111	127	16,777,216

In an "A" address, the left most Bit must always be "zero", all the other Bits can be whatever (zero or one) – designated by an X. In a "B" address, the left most Bit must be a "one" and the next Bit must be a "zero". And so on for the other classes. Class A addresses have providence over the remaining three octets for designation of hosts. This means that an A address can have up to 16,777,216 hosts. The last three octets have 8 Bits each. 8 Bits times 3 octets equals 24 Bit combination. Therefore, the number of host possibilities is a function of 2 (because each Bit can either be a zero or one) to the exponential power of 24 or  $2^{24}$  which equals 16,777,216. Class B addresses have providence over the last 2 octets for a total possibility of 65,536 hosts. Class C addresses have dominion over the last octet for 256 host possibilities. The D class is multicast and not subnetted. The E class is experimental and not subnetted. The loopback address (127) always has a Bit pattern of 01111111 for the high order Bits in the first octet. The loopback can have 16 million plus address incarnations in the last three octets, all doing the same thing. Examples of class A addresses are as follows:

63.125.35.214  
121.224.26.38  
12.224.253.236

Class B address examples would look like the following:


128.221.205.31

191.23.2.254  
165.26.251.250


Class C address examples would look like the following:

192.163.222.20  
198.4.2.3  
222.222.222.222

For a class C address the first three octets are part of the network portion of the IP address. The last or fourth octet contains information about the host portion of the IP address:

Network                      Host  
  
222.222.222.222

For a class B addresses the first two octets are part of the network portion of the IP address. The last two octets contain information about the host portion of the IP address:

Network                      Host  
  
128.222.222.222

For a class A address the first octets is concerned with the network portion of the IP address. The last three octet contains information about the host portion of the IP address:

Network                      Host  
  
122.222.222.222

There are 126 class A addresses and each address can have 16 million plus hosts. That is not to say that all class A addresses have or are using that many hosts. A network with 16 million members is too big and doesn't occur in the real world. We never put 16 million computers on the same wire because there is not enough bandwidth for all computers to talk. We break them up into more manageable groups or subnets.

## The Subnet Mask

So, now we arrive at the heart of the matter – the subnet mask. A subnet mask is used to divide the 4 billion plus IP addresses into smaller, more manageable groups. A subnet is a subset of a larger group. A mask is imposed on top of an IP address in order to define the dimensions of itself and its neighbors. An IP address or a subnet mask is not designed to be by itself; they each define and enhance one another when coupled together. A subnet mask defines the grouping of contiguous IP addresses. Contiguous means consecutive number sequences, like 0, 1, 2, 3, 4, 5, 6, 7. This is a group of eight contiguous consecutive numbers. A mask creates groups or groupings.

We say that a subnet mask has a defining point. The defining point in a subnet mask is where the ones and the zeros meet. For purposes of this analysis we will assume that all subnet masks have contiguous Bit patterns. There are special exceptions, but that would be an advanced topic which will not be covered here. A standard subnet mask, as defined here, has all the ones grouped together and all the zeros together. This is also defined in CIDR notation, which we will talk more about in a little while. **The important point here is to realize that the defining point in a subnet mask is always in one of three octets where the ones meet the zeros.** Standard Subnet mask analysis is always done at the defining point in one of three octets. We have class A, B and C subnet masks. We also say that a subnet mask does three primary things:

1. It separates the network portion (1's) of the IP address from the host portion (0's).
2. It tells us how many hosts there are, by counting the zeros, to determine the block size.
3. It tells us how many subnets there are, by counting the number of ones in the subnetted octet.

The default A subnet mask looks like the following:

255.0.0.0

It means that the network portion of the IP address is masked (covered) by the one's (255) and the host portion is defined by the zero's (0.0.0) In binary it looks like the following:

11111111.00000000.00000000.00000000

The same thesis is true for a B or C class subnet mask. The difference is that the B mask has two octets for network and two for hosts. The C mask has three octets for network and 1 for hosts. The whole point of a subnet mask is to make smaller groups from larger groups. Let's focus our attention now on analyzing the class C subnet mask because this is used most in the industry. **Remember, the defining point in a subnet mask is where the ones meet the zeros.** The reason is because it separates the network and host parts of the IP address. So the operative octet is always the one which contains the first zero.

Let's use a real world scenario to explain how to use a subnet mask. Say you work for a small company called "T Company" and your ISP (internet service provider) has assigned you a full class C IP address. Your address is 200.1.1.X with a default mask of 255.255.255.0. Since it is a full class C address there are 256 host possibilities – 0 to 255, represented by the X. However, we cannot use the first host address in any subnet because it is used as the identifier for that subnet, so the "0" number in our example is excluded. And we cannot use the last number in any subnet because it is used as the broadcast address for the subnet. So the number 255 in our example is also excluded. This leaves us with 254 usable IP addresses in our default class C subnet. The hard and fast rule is that the network ID and broadcast address from any subnet must always be excluded from usable addresses. In the subnet we are analyzing we always subtract two in our calculation of usable addresses.



## The Real World

Let's say we have 50 people working in our company right now, so, we have some room for expansions with a full class C IP address. We could easily put all 50 people in a default C subnet and everybody would be able to communicate. However the ten people in accounting deal with very sensitive information which we want no possibility of information leaking out. We want accounting on their own subnet. So we divide our 256 IP's in two groups with a 128 subnet mask. A class C 128 subnet mask would look like this in binary:

11111111.11111111.11111111.10000000

In decimal it is 255.255.255.128. **Remember we said that the defining point in a subnet mask is where the ones meets the zeros – the subnetted octet.** In the subnetted octet we count the number of ones, which when used as an exponent of 2, will tell us the number of subnets.  $2^1 = 2$ . We have two subnets. To find the number of hosts we count the number of zeros in the subnetted octet and use the answer as an exponent of 2.  $2^7 = 128$ . We have 128 hosts in two different groups! Each octet has 256 hosts or possibilities, because each octet is made up of 8 Bits. So, we can always check our calculation of subnets and hosts by multiplying each component. In our example we have 2 subnets and 128 hosts in each. 2 times 128 equals 256. Our IP's that we got from the ISP was 200.1.1.X. the accountants can go in the group of IP's 200.1.1.0 to 127 and the rest of the company can be in the group from 200.1.1.128 to 255. In the first group, the 0 address and the 127 address are excluded from usable hosts because they are network ID and broadcast respectively. In the second group 128 & 255 are excluded for the same reasons.

Let's consider for a moment that we have a R&D department in our T Company that has ten users who are bandwidth hogs. They are very sophisticated users that can regularly bring the network to its knees by over utilization. We want to isolate this group in their own subnet just like accounting. So now we need at least three groups. Remember we said that subnetting requires contiguous high order Bits. Therefore we can only create an even number of subnets, except the first or default subnet. We must create 4 subnets for our company (and have one unused extra). The following chart shows the all of the incarnations of subnets and hosts:

Bit Pattern & Bit Value	Subnet Mask/	Block size/	Provided Subnets/	Class C Usable	Class B Usable	Class A Usable
128 64 32 16 8 4 2 1	Bit Value	Octet Host #	# of Groups	Hosts - 2	Hosts - 2	Hosts - 2
00000000	0	256	1	254	65,534	16,777,214
10000000	128	128	2	126	32,766	8,388,606
11000000	192	64	4	62	16,382	4,194,302
11100000	224	32	8	30	8190	2,097,150
11110000	240	16	16	14	4094	1,048,574
11111000	248	8	32	6	2046	524,286
11111100	252	4	64	2	1022	262,142
11111110	254	2	128	0	510	131,070
11111111	255	1	256	0	254	65,534

If we looked in the table under provided subnets for the number of subnets that we need, which is 4, we would see that the mask is 255.255.255.192. However, it is always a better practice in math to be able to derive your answers, rather than memorize a table. The Bit pattern in a 192 subnet is the key. We are hunting for subnets, because it is the main criteria in our analysis at this point. We need 4 subnets. High order Bits (ones) always designate subnets. A 192 mask has 2 high order Bits –  $2^2 = 4$ . A subnetting problem is either driven by a necessity for certain number of hosts or a number of subnets or both. In our problem we have plenty of host addresses. What we need to create is more subnets. Subnets is driving our search. Sometimes one has to engage in a balancing act between provided subnets and usable hosts. **Regardless, we are always looking for the defining point in a subnet mask – where the ones meets the zeros.**

The real key to doing subnet masking with great confidence is knowing the powers of 2. Remember the secret to the powers of 2, is that 2, to the zero power, equals one; and each successive power is just double the last answer. The following is a table of the powers of two; understand the logic pattern that starts with  $2^0$ :

$2^0$	1	$2^7$	128	$2^{14}$	16,384	$2^{21}$	2,097,152
$2^1$	2	$2^8$	256	$2^{15}$	32,768	$2^{22}$	4,194,304
$2^2$	4	$2^9$	512	$2^{16}$	65,536	$2^{23}$	8,388,608
$2^3$	8	$2^{10}$	1,024	$2^{17}$	131,072	$2^{24}$	16,777,216
$2^4$	16	$2^{11}$	2,048	$2^{18}$	262,144	$2^{28}$	268,435,456
$2^5$	32	$2^{12}$	4,096	$2^{19}$	524,288	$2^{30}$	1,073,741,824
$2^6$	64	$2^{13}$	8,192	$2^{20}$	1,048,576	$2^{32}$	4,294,967,296

## Who is talking to whom?

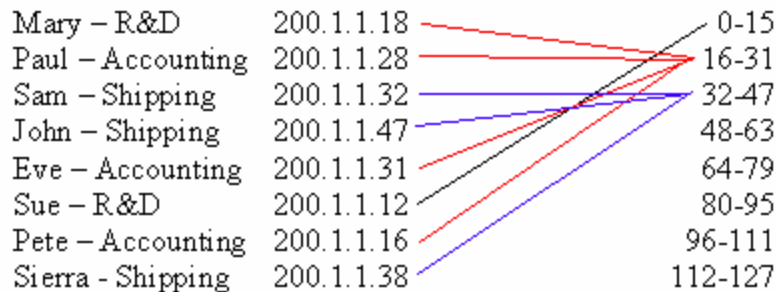
A real world problem that we often encounter is client computers not being able to talk. They are working fine, but cannot reach other computers. Many times it is because a computer is placed in the wrong subnet or given an illegal address. Lets look at some assigned host addresses and configured subnets and determine who is able to talk. T Company has grown to 120 users and 10 departments that we want all isolated. We still have plenty of IP addresses with our full class C. If we use a subnet mask of 255.255.255.240; we will get 16 subnets and 14 usable hosts in each subnet. We say 14 usable because we have to minus the network ID (all zeros) and broadcast address (all ones). Let's take a sampling of assigned addresses and explore typical problems that we may encounter.

Mary – R&D	200.1.1.18
Paul – Accounting	200.1.1.28
Sam – Shipping	200.1.1.32
John – Shipping	200.1.1.47
Eve – Accounting	200.1.1.31
Sue – R&D	200.1.1.12
Pete – Accounting	200.1.1.16
Sierra - Shipping	200.1.1.38

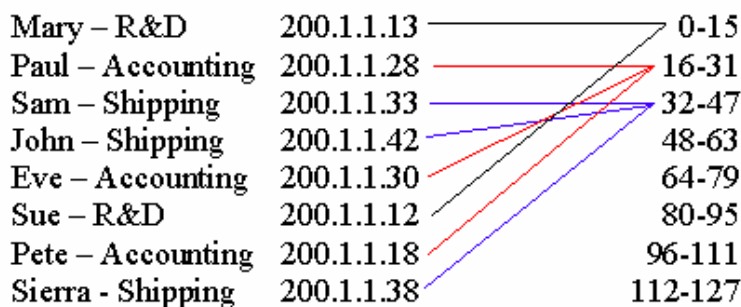
The first thing we have to do is determine which hosts are in what subnet. We do this by understanding the subnet mask, which is 255.255.255.240. A 240 mask has 16 subnets and 16 hosts in each subnet. So, we skip count by 16 to determine group or block size boundaries as follows:

0-15
16-31
32-47
48-63
64-79
80-95
96-111
112-127
And so on

Skip counting is counting by groups. For example: 2, 4, 6, 8, 10, or 5, 10, 15, 20, etcetera. Now when put this information next to each other, we can easily see who is talking to who:



We can see that Sue & Mary are in different groups and Mary should move. Paul, Eve and Pete are in the right group. Sam, John and Sierra are grouped properly. However there are still problems with the Accounting and Shipping subnets. Eve has been assigned the broadcast address for the accounting subnet. Pete has been assigned the network ID for the accounting subnet. Sam in Shipping has the Network ID for that subnet. And John has the broadcast address in the Shipping subnet. Let's modify things a Bit to fix things up:



Now we have the right people in the right groups and all have valid IP addresses. Subnetting is easy when you follow a methodology. The basis of the methodology presented here is working with the powers of 2.

## CIDR Notation

CIDR (classless interdomain routing) is an RFC standard which, among other things, allows us to associate any subnet mask with any class address. In other words, we can associate a class C subnet mask on either C, B or A addresses. Likewise a class B mask can be associated with either class B or A or C. The same goes for Class A, but the practicality of class A subnet mask is dubious because the group size is too big. CIDR notation works directly with the binary Bits. CIDR notation is largely replacing the conventional way we see a subnet mask written. Instead of writing a mask like 255.255.255.240, in CIDR notation we simple write /28. The /28 represents the high order binary Bits – 28 contiguous ones. Look at and count all the ones in the subnet mask - 255.255.255.240:

11111111.11111111.11111111.11110000

Any IP address can be easily expressed and written in CIDR notation. For example an address of 200.1.1.13 with a subnet mask of 255.255.255.240; can be more easily expressed and understood as:

200.1.1.13/28

Subnetting is easier when we think and use CIDR notation. The reason is we are dealing directly with the binary Bits and we no longer need to do a conversion from decimal notation. We just use basic arithmetic to calculate our hosts and subnets. We have 4 octets in every IP address –  $4 \times 8 = 32$ . So, we subtract the CIDR number from 32. The remainder will always tell us the number of zero's, which is the block size or number of hosts as an exponent of 2. We then divide the CIDR number by 8, the remainder tells us which octet has the defining point and the number of subnets (1's) in this octet as an exponent of 2.

Let's try another example of analyzing an IP address with CIDR notation to determine the number of hosts and subnets.

63.123.21.33/29

Here we have a class A address with a class C subnet mask. 32 minus 29 equals 3. The remainder 3 as an exponent yields the equation  $2^3 = 8$ . We have 8 hosts in each subnet. Now we divide 29 by 8 which equal 3 with a remainder of 5. The 3 tells us the subnetted octet is the Class C range. The remainder of 5 (as an exponent) tells us how many subnets we have in this class C range ( $2^5 = 32$ ). So now we know that there are 32 subnets and each subnet has 8 hosts. We can check our work by using the simple rule that subnets times hosts must always equal 256 ( $8 \times 32 = 256$ ).

**CIDR notation in conjunction with the powers of 2, is fastest and easiest methodology for constructing and solving subnetting problems.** The next page is a summation of all that we talked about to resolve subnetting issues. You can use it as a guide, refresher and memory booster. I always tell my students that you only have to memorize one fact to be a good subnetter:

$$2^0 = 1$$

## Sterler Subnetting System

CLASS	high order Bits	1 <sup>st</sup> octet range	Default Subnet Mask	Provided Hosts	CIDR Notation
A	0xxxxxxx	1-126	255.0.0.0	16,777,216	/8
B	10xxxxxx	128-191	255.255.0.0	65,536	/16
C	110xxxxx	192-223	255.255.255.0	256	/24
D	1110xxxx	224-239	N/A	N/A	N/A
E	1111xxxx	240-255	N/A	N/A	N/A
Loopback	01111111	127	N/A	16,777,216	N/A

Bit Pattern & Bit Value	Subnet Mask/ Bit Value	Block size/ Octet Host #	Provided Subnets/ # of Groups	Class C Usable Hosts-2	Class B Usable Hosts-2	Class A Usable Hosts-2
00000000	0	256	1	254	65,534	16,777,214
10000000	128	128	2	126	32,766	8,388,606
11000000	192	64	4	62	16,382	4,194,302
11100000	224	32	8	30	8190	2,097,150
11110000	240	16	16	14	4094	1,048,574
11111000	248	8	32	6	2046	524,286
11111100	252	4	64	2	1022	262,142
11111110	254	2	128	0	510	131,070
11111111	255	1	256	0	254	65,534

2 <sup>0</sup>	1	2 <sup>7</sup>	128	2 <sup>14</sup>	16,384	2 <sup>21</sup>	2,097,152
2 <sup>1</sup>	2	2 <sup>8</sup>	256	2 <sup>15</sup>	32,768	2 <sup>22</sup>	4,194,304
2 <sup>2</sup>	4	2 <sup>9</sup>	512	2 <sup>16</sup>	65,536	2 <sup>23</sup>	8,388,608
2 <sup>3</sup>	8	2 <sup>10</sup>	1,024	2 <sup>17</sup>	131,072	2 <sup>24</sup>	16,777,216
2 <sup>4</sup>	16	2 <sup>11</sup>	2,048	2 <sup>18</sup>	262,144	2 <sup>28</sup>	268,435,456
2 <sup>5</sup>	32	2 <sup>12</sup>	4,096	2 <sup>19</sup>	524,288	2 <sup>30</sup>	1,073,741,824
2 <sup>6</sup>	64	2 <sup>13</sup>	8,192	2 <sup>20</sup>	1,048,576	2 <sup>32</sup>	4,294,967,296

The defining point in a subnet mask is in only one of three octets and it does 3 primary things:

- 1- It separates the network portion of the IP address from the host portion
- 2- It tells us how many hosts there are – the block size, using the # of Zero's in the subnetted octet
- 3- It tells us how many subnets there are – using the # of ONES in the subnetted octet

The number of subnets times the number of hosts must equal 256 because each octet has 256 possibilities

The defining point in a subnet mask is where the ones (NETWORK/SUBNETS) meet the zeros (HOSTS)

Skip count with the block size # to determine each subnet range and what hosts belong in each subnet block

Always exclude the Network ID address (all zero's) and the Broadcast address (all ones) from every subnet

The CIDR notation is equal to the number of high order Bits that are contiguous ones

We divide the CIDR # by 8 to determine the subnetted octet and the remainder identifies the defining point

We determine the block size by subtracting the CIDR # from 32 and use the answer as an exponent of 2