IP Subnetting

This appendix discusses what subnetting is and gives some examples of how IP addresses can be separated into a network part and a host part. It describes a simple college subnet and shows how subnet masks are used to decode IP addresses. The final section covers binary numbers and converting to and from decimal.

Subnetting provides a mechanism for dividing a large unwieldy network into smaller sections that can be managed more effectively. It also offers better network security as measures can be applied at network boundaries and may reduce overall network traffic.

Subnetting does not provide additional IP addresses - only private IP addressing using NAT does that.

What is Subnetting

Subnetting is a way of partitioning a network at the IP level by dividing a block of addresses into a number of smaller sets.

To understand what is happening, first look at some small numbers rather than a full IP address.
Take the set of 'addresses' 100 to 129.
If computers used decimal representation, this network of 30 addresses might naturally be split into:

100 - 109

110 - 119

120 - 129

The first two digits represent the subnet number (10,11,12), while the final digit gives the address within the subnet.

Since computers use binary representation (base 2) rather than decimal (base 10) an equivalent pattern only emerges if the addresses are put into their binary form and the sets are broken into powers of 2 rather than powers of 10. (See below for an explanation of binary numbers.)

Looking at the first few addresses in the set, if the numbers are broken after the fifth digit then the two subnets, 01100 and 01101, are clearly visible. (Note that the network does not split into the same subnets as above.)

Decimal	Binary	Subr
100	01100100	0110
101	01100101	0110
102	01100110	0110
103	01100111	0110
104	01101000	0110
105	01101001	0110
106	01101010	0110
107	01101011	0110

etc.

Since an address could be split into network address and host number at any point, the boundary for a particular network must be known. This is achieved using a subnet mask that is the same length as the address, and has ones in the subnet address position and zeroes afterwards. For the above example the subnet mask would be 11111000 - five ones for the five digits of the subnet and zeroes for the remainder of the address - or equivalent to 248 in decimal.

Note that since each subnet has an associated subnet mask, the sizes of the subnets, and consequently the numbers of addresses in them, can vary.
This form of mask enables the address of the subnet to be extracted by performing a logical AND on an address and the subnet mask, which in turn makes it easy for the routing software to decide whether a destination address is in the local network or not and route the packets accordingly.

Full Length Examples

If the principle is extended to full length IP addresses, it produces longer addresses and masks. An IP address is four bytes long and these four bytes hold both the network address and the host number within the network. The associated mask defines the extent of each part of the IP address.

Considering the IP addresses 193.62.83.10 and 193.62.83.108 and an associated mask of 255.255.255.224:

The first becomes:	11000001 00111100 01010011 00001010
With a mask of	11111111 11111111 11111111 11100000
ANDing gives	
Subnet address of	11000001 00111100 01010011 00000000

So IP address 193.62.83.10 lies within the subnet that starts at 193.62.83.0.

The second becomes:	11000001 00111100 01010011 01101100
With a mask of	11111111 11111111 11111111 11100000
ANDing gives	
Subnet address of	11000001 00111100 01010011 01100000

So IP address 193.62.83.108 lies within the subnet that starts at 193.62.83.96

Note that the mask used above contains 27 ones. An alternative way of specifying the mask that is associated with an address is to append that number, which is known as the 'prefix length', to the IP address: e.g. 193.62.83.10/27

Defining Subnets

Anycollege has decided to subnet their facilities, to separate the administrative functions from the faculties and to separate the computer department from the rest of the network.
They have been allocated Janet IP addresses starting at 193.62.83.0. They want to allow for a maximum of 128 hosts for the faculties and 32 each for administration and computing. To conserve address space, the start positions of the subnets have been selected to minimise padding in the address space by starting with the largest subnets and working down in size.
They have therefore defined the following structure.

Name	The relative position* in this address space at which this subnet starts	The subnet mask of this sub
Faculties	0.0.0.0	255.255.255.128
Administration	0.0.0.128	255.255.255.224
Computing	0.0.0.160	255.255.255.224

* The relative position of the first subnet is always given as 0.0.0.0, i.e., the start of the address space allocated to the college (but see note below). For each subsequent subnet the start position is selected to allow for the maximum possible number of hosts in the subnets that precede it.

NOTE: Historically the use of 'all ones' or 'all zeros' in either the subnet or the host part of the address was not allowed. The restriction on using 'all ones' in the host part remains as this represents the subnet broadcast address (i.e. 193.62.83.127 is a broadcast on 193.62.83.0/25). The restriction on the use of the 'all zeros' address may still be in place for older equipment.

The subnet mask defines the number of hosts available on a subnet. The mask 255.255.255.224 leaves five bits for the host address, which could theoretically hold 32 addresses. However, in practice there are only 30 generally available addresses on the two small subnets and 126 on the large one.

This reduction in the number of available addresses is the price that is paid for subnetting.

Decoding IP Addresses

The following table gives examples of IP addresses in Anycollege's small networks.

		Network
Mask	255.255.255.224	11111111.11111111.111111
Address (1)	193.62.83.135	11000001.00111100.010100
Address (2)	193.62.83.164	11000001.00111100.010100
Address (3)	193.62.83.144	11000001.00111100.010100

ANDing the mask and addresses shows clearly which subnet the IP-address belongs to: addresses 1 and 3 belong to the subnet starting at (1000 0000) or .128; address 2 belongs to the subnet starting at (1010 0000) or .160.

The following table shows an IP address in the larger subnet. Note that the mask used is shorter, thereby allowing for more hosts.

		Network
Mask	255.255.255.128	11111111.11111111.111111
Address (4)	193.62.83.96	11000001.00111100.010100

ANDing the mask and address shows clearly that IP address 4 belongs to the subnet starting at (0000 0000) or .0, i.e. the 'start' of the network (but see note above for restrictions on older equipment).

RFC 1219 On the assignment of subnet numbers [1] describes a way of flexibly allocating subnets that defers having to decide where to draw the subnet boundary.

Binary Numbers

Binary numbers use base 2 rather than base 10, which is used for decimal numbers. So rather than needing 10 characters to represent a number (various combinations of 0-9), the binary system needs only two (0 and 1) and similarly the position of a character indicates multiplication by powers of 2 rather than powers of 10.

So 101 in decimal means 1*100 + 0*10 + 1*1, making a hundred and one.

And 101 in binary means 1*4 + 0*2 + 1*1, making five.

The rightmost character of a binary number represents the number of ones (20), the next digit to the left the number of twos (21), the next digit the number of fours (22), and so on.

To convert binary to decimal

As an example - the binary number 1110 0101 is to be converted to decimal.

Identify what each position represents (work from right to left starting at 1 and multiplying by two on each left shift):

1	1	1	0	0
128	64,	32,	16,	8,,

Multiply and sum:

1*128 + 1*64 + 1*32 + 0*16 + 0*8 + 1*4 + 0*2 + 1*1 = 128+64+32+4+1 = 229 base 10 (decimal)

For speed, ignore the multiplication and just add the value of columns with a one in them, but the full method works for any base and is worth knowing.

To convert decimal to binary

For those familiar with it, this conversion is most quickly done using the hexadecimal system (see below). Otherwise it can be simply achieved by repeated division by two, recording the

remainder each time.

Starting with 229 base 10 (decimal) so that we can check the result with the previous example:

229 / 2 = 114 remainder 1

114 / 2 = 57 remainder 0

57 / 2 = 28 remainder 1

28 / 2 = 14 remainder 0

14 / 2 = 7 remainder 0

7 / 2 = 3 remainder 1

3 / 2 = 1 remainder 1

1 / 2 = 0 remainder 1

The binary equivalent reads from the bottom up, so 229 base 10 (decimal) = 11100101 base 2 (binary).

Hexadecimal Numbers

Hexadecimal numbers use base 16 rather than base 10, which is used for decimal numbers. So rather than needing 10 characters to represent a number (various combinations of 0-9), the hexadecimal system needs 16 and similarly the position of a character indicates multiplication by powers of 16 rather than powers of 10.

So	101 in decimal means	1*100 + 0*10 + 1*1 making

And 101 in hexadecimal means 1*256 + 0*16 + 1*1 making

The rightmost character of a hexadecimal number represents the number of ones (16^0), the next digit to the left the number of 16s (16^1), the next digit the number of 256s (22), and so on.

Since hexadecimal requires 16 characters to represent its digits rather than the 10 used for decimal notation, the first six letters of the alphabet are also used.

In addition, because 16 is a power of 2 (2^4) there is a direct relationship between hexadecimal and binary numbers: each hexadecimal digit representing a group of four binary digits as shown in the following table.

Decimal	Binary	Hexa
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9

10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Any binary number can therefore be represented as a (much shorter and more memorable) string of hexadecimal digits, e.g. using the previous example, the binary number 11100101 appears in hexadecimal as E5 (using the conversion table above, 1110 = E, 0101 = 5).

To Convert Hexadecimal to Decimal

As an example - the hexadecimal number E5 is to be converted to decimal. As for the conversion from binary to decimal, identify what each position represents by working from right to left, starting at 1 and multiplying by the base value (in this case 16) on each left shift:

16 1
E 5
Multiply and sum:

E (or 14)*16 + 5*1 = 224 + 5 = 229 base 10 (decimal)

To Convert Hexadecimal to Binary

Using decimal or hexadecimal rather than the underlying binary is easier to understand and much less prone to transcription errors. Hexadecimal, however, has the added advantage of giving easy access to the underlying binary pattern, which can be particularly useful when working with masks.
That the mask 1111 1111 .1111 1111 .1111 1111 .1000 0000 is rendered in decimal notation as 255.255.255.128 is not obvious, but the mapping to FF.FF.FF.80 is trivial using the table above.

1111	1111	1111	1111	1111
F	F	F	F	F

Equally, the reverse conversion from the address C1.3E.53.96 to binary is a simple matter of substituting the binary equivalents from the same table - much more straightforward than converting from the decimal, 193.62.83.96.

C	1	3	E	5
1100	0001	0011	1110	0101

Links
[1] http://www.faqs.org/rfcs/rfc1219.html