IP numbers

What is an IP number (IP address)?

- Every computer (host) on the Internet is identified by an **IP number**
- Every IP number is **different** - if two computers had the same IP number, the network would not know which one to deliver data to!
- An IP number is a **32-bit binary number**, that is, a string of 32 ones and zeros.

For convenience of writing, we break IP numbers into four 8-bit pieces and convert them to decimal, with a period between each part:

```
e.g.  IP number:   00001010100010111100101110000011
                      /      /         \       \
                 00001010 10001011 11001011 10000011

Decimal version:    10   .   139  .   203  .  131
```

The conversion is done by giving a value to each '1' in the binary number then adding them together. The rightmost bit counts as "1", the next one as "2", the next one as "4" etc.

e.g. to convert 11001011 to decimal:

```
+-------- 128
|+------- 64
||+------ 32
|||+----- 16
||||+---- 8
|||||+--- 4
||||||+-- 2
|||||||+- 1
||||||||
||||||||
11001011  = 128 + 64 + 8 + 2 + 1 = 203
```

It's easier to use a binary conversion table instead. Remember though that this conversion is just to make IP numbers easier to type; the computer uses the binary version.

Ranges of IP numbers

What are the smallest and largest IP numbers? The smallest will consist of all 0's, and the largest all 1's.

```
Smallest:  00000000000000000000000000000000   =   0.0.0.0
Largest:   11111111111111111111111111111111   =   255.255.255.255
```

There are 2^{32} = just over 4,000,000,000 IP numbers in total.

Allocation of IP numbers

- To make sure all IP numbers are unique, they are allocated in blocks
- The InterNIC assigns large blocks of IP numbers to service providers, who in turn break them down into smaller blocks for their customers
- You in turn can break them down into smaller blocks for each of your own networks

To create blocks, IP numbers are broken into two parts: the **network number (prefix)** and **host number**. To specify the position of the split, you give a slash and the number of bits in the network number part.

For example, the network "199.2.192.64**/26**" has 26 bits of network number; the remaining 6 bits are the host number.

For IP routing to work, it is vital that all hosts on one network have the *same* network number (in this example, the left-hand 26 bits), and *different* host numbers.

In other words, in our example network, all the IP numbers must look like this:

```
Network 199.2.192.64 / 26

11000111 00000010 11000000 01xxxxxx
\____________________________/\____/
  network number (26 bits)   host (6)
```

where "xxxxxx" can be any combination of six 0's and 1's. The lowest value is given by all 0's, and the highest value all 1's.

```
11000111 00000010 11000000 01000000   =  192.2.192.64
11000111 00000010 11000000 01000001   =  192.2.192.65
11000111 00000010 11000000 01000010   =  192.2.192.66
                             ......
                             ......
11000111 00000010 11000000 01111111   =  192.2.192.127
```

Note that a network number like 199.2.192.64/26 is not by itself an IP number; rather, it defines a **range** of IP numbers which can be used by hosts on a single network.

In practice, the first and last values in each range are not permitted for real hosts (they are reserved). In the example above, .64 and .127 are reserved, so this network can actually use numbers 199.2.192.65 to 126, enough for 62 hosts altogether.

Subdividing networks

All hosts on a single physical network (such as an ethernet segment) must have the same network number, so if you have more than one network you must subdivide the block of numbers which have been allocated to you.

You do this by increasing the length of the network number (thus reducing the length of the host number) - giving you more networks each with fewer hosts. If you've done it correctly you will end up with ranges of IP numbers which **do not overlap**.

One approach to "doing it right" is to increase the length of the network number *one bit at a time*. For example, you split a /24 into two /25's, a /25 into two /26's, and so on.

Here is an example of dividing the network 206.27.238.0 / 24 :

```
  206   .   27   .   238
11001000 00011011 11101110 xxxxxxxx     206.27.238.0/24   before
-------------------------------------------------------------------
11001000 00011011 11101110 0xxxxxxx     206.27.238.0/25   after
11001000 00011011 11101110 1xxxxxxx     206.27.238.128/25
                           ^
         Look! another bit in the network number
```

This gives you two networks each with 128 IP numbers (126 excluding the reserved ones). If that's what you want, fine; otherwise, continue by splitting one or both of those networks further.

```
11001000 00011011 11101110 0xxxxxxx     206.27.238.0/25
-------------------------------------------------------------
11001000 00011011 11101110 00xxxxxx     206.27.238.0/26
11001000 00011011 11101110 01xxxxxx     206.27.238.64/26
                            ^

11001000 00011011 11101110 1xxxxxxx     206.27.238.128/25
-------------------------------------------------------------
11001000 00011011 11101110 10xxxxxx     206.27.238.128/26
11001000 00011011 11101110 11xxxxxx     206.27.238.192/26
                            ^
```

Now, one /24 network (256 IP numbers) has been divided into four /26 networks with 64 IP numbers each. If you work them out they are 0-63, 64-127, 128-191 and 192-255, which is fine (i.e. they don't overlap). Since you must discard the first and last number in each range, the usable numbers are 1-62, 65-126, 129-190 and 193-254.

Here is an example of a **WRONG** way to subnet this network:

```
First network:  206.27.238.0/25
Second network: 206.27.238.64/26

Why is it wrong?

206.27.238.0/25  = 206 . 27 . 238 . 0xxxxxxx  <-- range 0 to 127
206.27.238.64/26 = 206 . 27 . 238 . 01xxxxxx  <-- range 64 to 127  OVERLAP!
```

By splitting some networks and not others, more complex allocations are possible with different lengths of network number. You are unlikely to come across a more complex example than this one:

```
11001000 00011011 11101110 xxxxxxxx     (original network)
-------------------------------------------------------------
11001000 00011011 11101110 000000xx     206.27.238.0/30     .0 to .3
11001000 00011011 11101110 000001xx     206.27.238.4/30     .4 to .7
11001000 00011011 11101110 000010xx     206.27.238.8/30     .8 to .11
11001000 00011011 11101110 000011xx     206.27.238.12/30    .12 to .14
11001000 00011011 11101110 0001xxxx     206.27.238.16/28    .16 to .31
11001000 00011011 11101110 001xxxxx     206.27.238.32/27    .32 to .63
```

```
11001000 00011011 11101110 01xxxxxx     206.27.238.64/26    .64 to .127
11001000 00011011 11101110 1xxxxxxx     206.27.238.128/25   .128 to .255
```

Since you can't use the first and last IP number in each range, a subnet like 206.27.238.4/30 only actually gives you two IP numbers (.5 and .6), so /30 is the smallest practical subnet. A common use for a tiny subnet like this is for a serial link between two routers, which needs only one IP number for each end of the link.

Network masks

The split between network number and host number can also be represented by a **network mask**; this is just a number with a '1' in every position of the network part, and a '0' for every bit position of the host part. For example, the netmask for a /26 network is 26 "1"s followed by 6 "0"s.

A lot of software will require you to enter a netmask in either dotted decimal or (occasionally) hexadecimal form; just use a conversion table.

```
e.g.   /26 = 26 x "1", rest are "0"
           = 11111111 11111111 11111111 11000000  (binary netmask)
           =   255   .  255   .  255   .   192    (decimal netmask)
```

Classes *(historical)*

IP address space was originally divided into "classes" each with a fixed netmask. You can tell the class from the *first byte* of the IP number, and hence work out the original netmask. Some software will use this as the default netmask if you don't specify one explicitly.

```
0xxxxxxx       0-127    =  class A,  /8   255.0.0.0
10xxxxxx       128-191  =  class B,  /16  255.255.0.0
110xxxxx       192-223  =  class C,  /24  255.255.255.0

Example: 130.16.4.7 is a Class B network since 130 is between 128 and 191
```

Remaining numbers are reserved for special or experimental purposes, such as multicasting (224-239)

(*) In the bad old days, after you had divided a network into subnets, you were not permitted to use the first range of IP numbers, and sometimes not the last range either. This was obviously very wasteful of numbers and this requirement has been discontinued - but you may have to tell your router the good news before it will work. For example, on a Cisco, the command is "ip subnet-zero". The TCP/IP stack originally supplied with Novell 3.x refused to let you use the first and last subnets at all, but an update is now available.

Last Updated 13 June 1999