

1 GSW... IPv4 Addressing

For as long as I've been working with the Internet protocols, people have been saying that IPv6 will be replacing IPv4 in a couple of years' time. While this remains true, it's worth knowing about IPv4 addresses. Even when IPv6 does take over the core of the Internet, there will probably be small pockets of IPv4 networks left around, and a knowledge of how IPv4 addresses work will be useful. That's what this chapter is about: IPv4 addresses, bit-masks, and how routing tables work.

Unlike the 48-bit hardware or MAC 'addresses' associated with Ethernet cards (see the Chapter on Project 802 LANs), IPv4 addresses really are addresses: they contain information about where the node is on the network. They're composed of two fields, a network identifier (which identifies which network the node with this IP address is on) and a node or host identifier (which identifies the individual node on the network). The network identifier comes first¹:

Network Identifier	Node/Host Identifier
--------------------	----------------------

Also, unlike these 48-bit MAC addresses, IPv4 addresses are only 32-bits long (just over four billion unique addresses) and that has caused a lot of trouble as the Internet has expanded².

Both addressing schemes came from two different backgrounds: IP was designed from the outset to service a wide-area network where packets might have to be forwarded through several intermediate nodes before reaching their final destination. IP addresses therefore needed to contain information about the location of the destination node. 802 LANs were designed to run on a shared medium, where everyone could talk directly to everyone else, and there is no need for the 'addresses' to mean anything.

1.1 Introduction to IPv4 Addresses

IPv4 addresses are always 32 bits long³, and are most commonly written in what's known as *dotted decimal notation*, such as X.X.X.X where X is a decimal number from 0 to 255, for example 176.54.23.207 is a valid IPv4 address. Each decimal number represents eight bits, so the total IPv4 address is 32-bits long.

Hexadecimal	B036174D			
Dotted-decimal	176	54	23	77
Binary	10110000	00110110	00010111	01001101

In theory then, there are $2^{32} \approx 4.3$ billion IPv4 addresses: not quite enough to give everyone in

¹ Note this is the other way round from postal addresses, where the house number (the equivalent of the node identifier) comes first; however it's the same way round as post codes, where the first few characters identify the town, and the remainder the street and house.

² In fact, if IPv4 addresses were 48-bits long, the single largest motivation for IPv6 would be removed, and it's possible IPv6 would not happen at all.

³ IP version 6 addresses are 128 bits long.

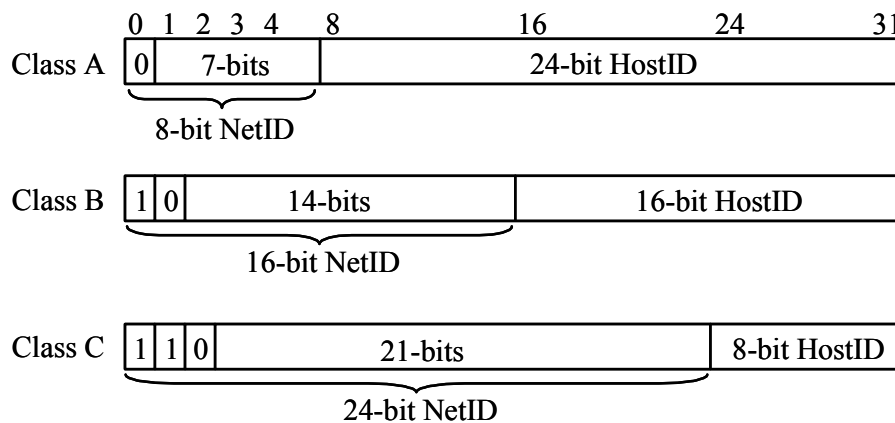
the world one each, but pretty close. The designers of the original ARPANET thought this was a suitably large number, and would be fine for all future use of their network. In fact, IPv4 addresses are in very short supply, are due to run out completely in a few years, and already some bodes have had to be made⁴ to make sure that there are enough to go around. Why? Well...

In the beginning (RFC 760) IP addresses were divided into two parts: an 8-bit network number (*NetID*) and a 24-bit host number (*HostID*), identifying the machine in that network. That allowed $2^8 - 2 = 254$ networks⁵ to join the Internet, each with $2^{24} - 2 = 16777214$ nodes⁶. Two things soon became apparent: far more than 254 organisations would want to join, and none of these networks contained anything approaching 16 million nodes.

Asking around, it became obvious that some networks had the potential to be very large, with millions of nodes (for example, IBM's network), whereas others were very small, and were unlikely to ever have more than a few nodes. Assigning all of these networks the same (huge) number of addresses to use was very wasteful.

The proposed solution was known as *classful addressing* (RFC 791). The available address space was divided up into five *classes*, known as class A, B, C, D and E. (Addresses in the Class D addresses space were intended for multicasting, and addresses in the class E space are reserved and not available for use; the other three classes were used for the addresses of individual nodes on the Internet.)

Class A addresses had the same format as the original scheme: an 8-bit network identifier and a 24-bit host identifier identifying an individual node on the relevant network. Class B addresses had a 16-bit network identifier and a 16-bit host identifier, so there were a lot more class B addresses, although each was limited to $2^{16} - 2 = 65534$ nodes. Class C addresses had a 24-bit network identifier and an 8-bit host identifier, so there were thousands of possible class C networks, although they were all limited to a maximum of 254 nodes.



⁴ The most important of which is known as network address translation (NAT).

⁵ Why 254? Because as usual in networking all '1's means "all / everyone" and all '0's means "this / not-known" and neither can be used to identify an individual network.

⁶ Again, you have to subtract two of the possible addresses, since all '1's means "all / everyone" and all '0's means "this / not-known" and neither can be used to identify an individual node on a network.

Figure 1-1 – IPv4 Classes A, B and C

This allowed much more efficient use of the IP addressing space: anyone who only had a few nodes on their network could be given a class C address and 256 IP addresses, rather than over 16 million. You could tell which class your address belonged to by looking at the top few bits of the IP address: all class A addresses began with a '0', all class B addresses with the two bits '10' and all class C addresses with the three bits '110'.

There are, in theory, only 126 possible class-A networks⁷, and originally only half of them were assigned. Any IP address with a first byte of 9, for example, belongs to IBM⁸.

There are 16382 possible Class B addresses were very popular, you could have a reasonably large number of hosts (up to 65534) and many organisations thought this would be a very suitable number.

There was an apparently huge number of possible class C addresses (2,097,150 to be exact), but they weren't popular. They were fine for anyone with just one small network and no plans to expand, but anyone who thought their network could grow to more than 254 nodes wanted a class B address, to avoid the complete nightmare of having to go around and change all the IP addresses in the network when adding the 255th node. This resulted in most class B networks having only a few hundred nodes on them at most, which wasn't a very efficient use of the possible IPv4 address space (each class B network was capable of supporting 65534 nodes, after all), and it meant that the class B network identifiers soon began to run out.

Consequently, although almost all of the possible IP addresses were assigned, only a few percent were actually used.

Address	Range of NetIDs	Range of HostIDs
Class A	0.0.0.0 - 127.0.0.0	0.0.0 - 255.255.255
Class B	128.0.0.0 - 191.255.0.0	0.0 - 255.255
Class C	192.0.0.0 - 223.255.255.0	0 - 255

Figure 1-2 Address Ranges for Class A,B and C Networks

There are a few special cases as well that are useful to know about, in particular:

- 0.0.0.0 All zeros. You might think this one should mean "me". It does, but only when the node doesn't know its own IP address, and it's only ever used as the source IP address, so it could perhaps more accurately be described as "I don't know yet". If you want to send a packet to yourself, and you don't know your own IP address, you can always use the destination IP address of 127.0.0.1

⁷ Once again, you can't use all zeros or all ones as the network identifier. Also this is 126 networks in theory only: in practice some of these network addresses are reserved: for example the class A address 10.0.0.0 is reserved for private use, and any packets addressed to such a network should not be forwarded by any router on the Internet.

⁸ Or at least it does now. Originally it belonged to Fort Bragg's packet radio network. One military camp, with one network, and the rights to 0.4% of all the IP addresses in the world. It's enough to make the Chinese weep.

- 127.0.0.1 This is the *local loopback* address: any packet sent to this IP address⁹ will be received by the node that sent it, but no other node. In fact, it would never leave the node: packets to this address are never transmitted over any network. (You could of course use your own IP address – but using 127.0.0.1 avoids the need to find out what your own IP address is, and can be used even when you don't have an IP address. Also, send a packet to your own node address, and it will be transmitted onto the local Ethernet, so other people could see the packet. Send a packet to 127.0.0.1, and no-one else will ever see the packet.)
- 255.255.255.255 In other words, all ones. This does not, as you might expect, mean “all nodes on all networks”. It's just a *limited broadcast* address, and packets sent to this address are received by all nodes on the current network only, but not forwarded through any routers¹⁰. It's an address that a station can use to shout for help when it has no idea which network it is on.
- The NetID followed by all zeros This is known as the *network address*. It's not used, and cannot be assigned to any individual node.
- The NetID followed by all ones This is the *directed broadcast* address. Packets sent to this address should be received by all nodes on the same network, but not by any nodes on any other networks.

Note there is no address that means ‘the whole Internet’. If there were, some idiot somewhere would send millions of packets to it, and irritate everyone. Some people are like that.

1.1.1 An Example Internetwork using IPv4

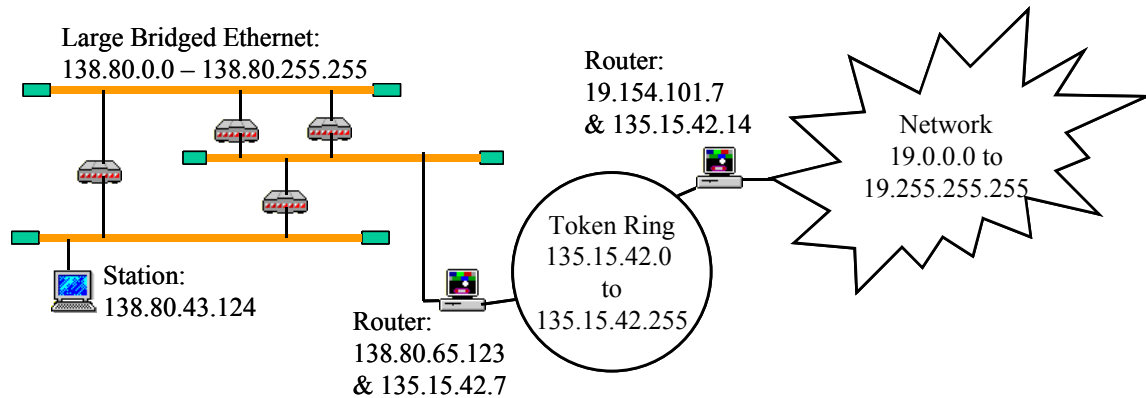


Figure 1-3 – Example Internetwork using IPv4

Note that routers have at least two IP addresses: one for each network to which they are attached.

⁹ Actually, this should happen for any address in the range from 127.0.0.1 to 127.255.255.254, but conventionally 127.0.0.1 is used.

¹⁰ There are some exceptions to this, but they have to be explicitly supported in the routers, and

1.1.2 *Where to get an IP Address*

IPv4 Addresses are running out. At the moment, they are managed for the world by the Internet Assigned Numbers Authority (IANA) (<http://www.iana.org>) who assign blocks of numbers to the five regional authorities¹¹. In Europe, our local authority is RIPE (Reseaux IP Europeans) (at <http://www.ripe.net>)¹². If you want a set of IP addresses, you could go to them, but unless you want a large block of addresses, they'll probably refer you to a local Internet Service Provider (ISP) offering service in your country (these providers can both assign you some IPv4 addresses, and give you a connection to the Internet). There are dozens of these ISPs: their details are on the RIPE website.

1.2 Classless Addressing and the Current Situation

While classful addressing helped, it still resulted in rather inefficient use of the rapidly diminishing number of unassigned IPv4 addresses. Anyone who wanted more than 254 nodes on their network had to have a block of at least 65536 addresses, and probably used only a small fraction of these.

A couple of ideas came along to help: firstly, *subnetting* (RFC 950). Subnetting allows class A and B networks be split into subnetworks with smaller ranges, so that several networks could share a class A or B address. (You could split class C addresses as well, but that's not as useful.) Secondly: *supernetting*, which is in some senses the opposite: supernetting allows several consecutive class C networks to be joined together to make a larger network space. For example, anyone who wanted 300 IPv4 addresses could request two consecutive class C address spaces, and combine them into one large network with room for up to 510 nodes.

Of course, this makes a mockery of the class system. Essentially we now have a classless system whereby people are assigned a range of IP addresses, and the number of addresses they have has nothing much to do with whether their addresses fall into the traditional class A, class B or class C ranges. Which creates a new problem: if you can't tell how big a network is from looking at the top few bits of the address (and thereby seeing if it's a class A, B or C address), how can you tell how big the network is?

Answer: by using a bit mask. The bit mask is a 32-bit number composed of a series of '1' bits followed by a series of '0' bits. (There are therefore only 33 possible bit masks, ranging from 0 to 255.255.255.255). The masks have '1' bits where the IPv4 address has its NetID field, and '0' bits where the IPv4 address has a HostID field.

For example: a traditional class-B address would have a bit mask of 255.255.0.0, since the top sixteen bits are the network identifier, and the bottom sixteen bits are the host identifier. A traditional class-C address would have a bit mask of 255.255.255.0. Constructing bit-masks in this way makes it very easy to work out which network an IP address belongs to, you just do a logical AND of the IP address with the bit mask. The result is the *network address*.

¹¹ RIPE for Europe, AfriNIC for Africa, ARIN for North America, LACNIC for Latin America and the Caribbean, and APNIC for Asia and the Pacific.

¹² If you want to know who's reserved what IP addresses in the UK, then <http://www.ripe.net> is the place to go and look.

Since masks always contain a run of '1's followed by a run of '0's, and they're always 32-bits long, you can write them using just one number: the number of '1' bits in the mask. So, for example, IBM's class-A addressing space could be written 9.0.0.0/8 any IP address with the first eight bits equal to "00001001" (i.e. nine in binary) is in IBM's addressing space.

An example: suppose a computer had an IP address of 130.1.9.1, with a subnet mask of 255.255.248.0 (this could also be written 130.1.9.1/21). What is the network number and host number? How many computers could be on this network?

Firstly, there are 21 bits in the bit mask, and ANDing the bit mask with the address gives

$$130.1.9.1 \text{ AND } 255.255.248.0 = 130.1.8.0 \quad (0.1)$$

That's the network number: 130.1.8.0. The remaining $32 - 21 = 11$ bits in the IPv4 address can be found by doing a logical AND with the inverse of the bit mask:

$$130.1.9.1 \text{ AND } 0.0.7.255 = 0.0.1.1 \quad (0.2)$$

so the host identified could be written as 0.0.1.1, or 1.1, or just 257.

As to how many hosts are on this network: any number between '000.00000001' and '111.11111110' is a valid host-ID, which makes 2046 of them (remember all zeros and all ones have special meanings and can't be assigned to host computers).

Note that this also means that the number of IPv4 addresses in a given range is always a power of two. Anything / Y means that there are $(32 - Y)$ bits left for the node identifier, and that means there are $2^{(32 - Y)}$ possible IPv4 addresses: $(2^{(32 - Y)} - 2)$ can be assigned to individual nodes (the other two are the network address and the broadcast address). A few examples:

<i>Range</i>	<i>Network Address</i>	<i>Broadcast Address</i>	<i>Number of Individual Addresses Available</i>
172.0.0.0 / 31	See footnote ¹³		
172.0.0.0 / 30	172.0.0.0	172.0.0.3	2
172.0.0.0 / 29	172.0.0.0	172.0.0.7	6
172.0.0.0 / 28	172.0.0.0	172.0.0.15	14
172.0.0.0 / 27	172.0.0.0	172.0.0.31	30
172.0.0.0 / 26	172.0.0.0	172.0.0.63	62
172.0.0.0 / 25	172.0.0.0	172.0.0.127	126
172.0.0.0 / 24	172.0.0.0	172.0.0.255	254
172.0.0.0 / 23	172.0.0.0	172.0.1.255	510
172.0.0.0 / 22	172.0.0.0	172.0.3.255	1022

¹³ You might think that it's impossible to have a network mask 31 bits long: this range only contains two IPv4 addresses, and after taking out the network and broadcast address, this leaves none left for individual nodes. In fact, it can be done (see RFC 3021), although you have to turn off the broadcast address. If you want to use a broadcast address for these networks, you have to use the limited broadcast address 255.255.255.255.

172.0.0.0 / 20	172.0.0.0	172.0.15.255	4094
172.0.0.0 / 17	172.0.0.0	172.0.127.255	32766

and so on. In each case (except / 31), the broadcast address is $2^{(32 - Y)} - 1$ higher than the network address.

Another example: the company I used to work for, Madge Networks, was assigned some IP addresses by RIPE, and the RIPE entry looked like this:

uk.madge (Madge Networks Ltd) 19981221 212.107.128/19 ALLOCATED PA

That means they had the rights to use IP addresses starting at 212.107.128.0, with a total network identifier mask which starts with 19 '1' bits, with the remaining bits in the network mask being zero. How many IP addresses do they have? What is the largest IP address they can assign to any computer?¹⁴

1.2.1 Classless Inter-Domain Routeing (CIDR)

...and so it was decided that all this class-based stuff had been a bad idea in the first place, and we'd be better off without it¹⁵. The new scheme, using bit-masks, is called *Classless Inter-Domain Routeing* (CIDR) and it replaced the previous classful system. The only real drawback is that instead of just having to send someone 32-bits (the network address) to tell them about a new network, you now have to send them a 32-bit network address and a 32-bit mask, and routers need some additional memory to store the bit masks.

However, once you've got classless addressing, there is a very useful technique you can use to make the job of routers on the Internet easier. This is *variable-length subnet masking* (VLSM). The idea is that the network mask corresponding to a certain IPv4 address doesn't have to be the same length everywhere on the Internet. Different routers can store different subnet masks for the same IPv4 address.

At first sight, that might seem rather odd – however, consider a typical small network, attached to the Internet, as shown in the figure below. Assume that the network administrator of this network has been assigned the IP addresses 19.53.0.0/22. That gives $2^{(32 - 22)} - 2 = 1022$ different IP addresses, which here are distributed around five Ethernet segments.

Without VLSM, all of these routers would have to have entries in their routing tables for all of these five subnetwork segments. The routing table for router **A** would have to look something like:

IP Address	Send Packet to:	Send Packet from:
19.53.0.0 / 25	Router C	Port b
19.53.0.128 / 25	Router C	Port b
19.53.1.0 / 25	Router B	Port b
19.53.1.128 / 25	Router B	Port b

¹⁴ Answer: 8192, and 212.107.159.254 (they can't use 212.107.159.255, that would be a broadcast to all computers in the network).

¹⁵ IPv6 doesn't have classes at all. Lesson learned.

19.53.2.0 / 24	Direct to destination	Port b
Everywhere else	ISP	Port a

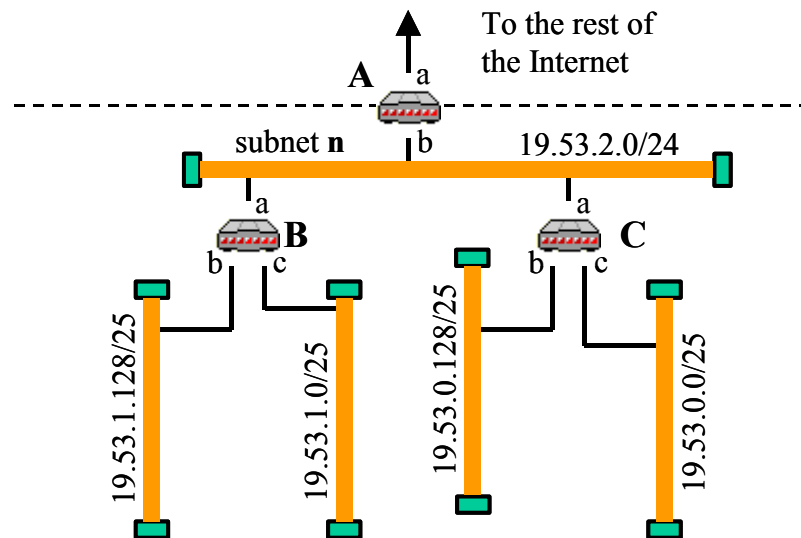


Figure 1-4 A Small Network Attached to the Internet

However, router **A** can use a technique called *route aggregation*. All it really needs in its routing table is:

IP Address	Send Packet to:	Send Packet from:
19.53.0.0 / 24	Router C	Port b
19.53.1.0 / 24	Router B	Port b
19.53.2.0 / 24	Direct to destination	Port b
Everywhere else	ISP	Port a

and all any router in the core of the Internet needs to know is that to deliver a packet anywhere in the range 19.53.0.0 / 22, it should be sent to router **A** in this network. That makes the routing tables in the core routers very much smaller, and that's very good thing: it saves memory, speeds up the looking up of destination addresses in the router's tables, and reduces the amount of information that has to travel between routers as they continually try to keep their routing tables up-to-date.

Then there's router **B**. All it really needs in its routing table is:

IP Address	Send Packet to:	Send Packet from:
19.53.0.0 / 24	Router C	Port a
19.53.1.128 / 25	Direct to destination	Port b
19.53.1.0 / 25	Direct to destination	Port c
19.53.2.0 / 24	Direct to destination	Port a
Everywhere else	Router A	Port a

It doesn't need to know that the address space 19.53.0.0 / 24 is divided into two subnetworks by router **C**. It just sends packets to anywhere in this address space to router **C**, and lets router **C** worry about how the network is sub-divided on the other side.

Note that the IPv4 address 19.53.1.73 is, according to router **B**'s routing table, in the subnetwork 19.53.1.0 / 25. In router **A**'s routing table, it's in the subnetwork 19.53.1.0 / 24. And in the routing tables of the core routers in the Internet, it's in the network 19.53.0.0 / 22. The same IPv4 address, but it's got three different bit masks of different lengths in different routers. It might seem a bit confusing at first, but it saves a lot of time and effort keeping all the routing tables in the Internet up-to-date. Now, the network administrator of this network can divide one of these sub-networks again if he wanted to, or transfer some addresses from one sub-network to another, and the rest of the Internet wouldn't need to know.

1.2.2 A Slight Variation: Unequal Sized Subnetworks

In the last example, all the subnetworks on ports b and c of routers **B** and **C** were the same size: they could each support 126 nodes. What if one network grew much faster than the other, so you wanted to have 200 nodes on one Ethernet, and only 20 on the other. Could you do this?

In theory, it's not hard. You could, for example assign the IPv4 addresses from 19.53.0.0 to 19.53.0.31 to the smaller subnetwork (subnet s in the diagram below), giving enough IPv4 addresses for 30 nodes (19.53.0.1 to 19.53.0.30); and the range from 19.53.0.32 to 19.53.0.255 to the larger subnetwork (subnet r in the diagram):

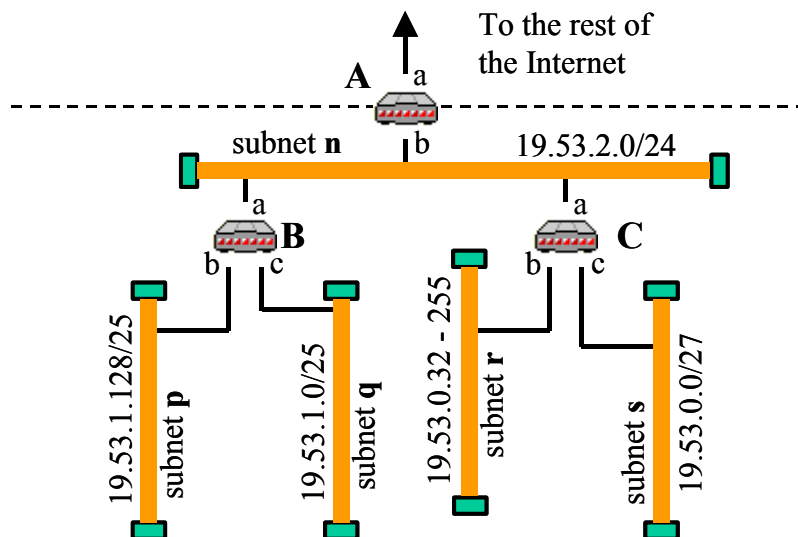


Figure 1-5 Example Network with Unequal Subnetworks

It's easy to see how subnet s could be put into the routing table for router C, it's just the range 19.53.0.0 / 27. But what about subnet r? How can you express the range of IPv4 addresses from 19.53.0.32 to 19.53.0.255 in terms of a network address and a bit mask?

The simple answer is: you can't. Not with just one network address and one bit mask, anyway. The problem is that a bit mask always leaves an integer number of bits available for the node identifier; and that implies the number of IPv4 addresses associated with a network must be a power of two. Here, we've got $255 - 32 + 1 = 224$ addresses assigned to subnet r, and that's not a power of two.

One solution is to express this range of addresses as the sum of several ranges, all of which do have a number of IPv4 addresses that's a power of two, for example:

19.53.0.32 / 27 Range from 19.53.0.32 to 19.53.0.63 (32 addresses)

19.53.0.64 / 26 Range from 19.53.0.64 to 19.53.0.127 (64 addresses)
 19.53.0.128 / 25 Range from 19.53.0.128 to 19.53.0.255 (128 addresses)

but that takes three entries in the routing table, and creates three subnetworks. It's a bit of a waste of IPv4 addresses too: there are now six addresses in the range of IPv4 addresses assigned to this subnetwork that can't be given to individual nodes: three network addresses, and three broadcast addresses, one for each of these ranges.

Fortunately, there's a solution. Assign the range 19.53.0.0 / 27 to subnet **s**, and the range 19.53.0.0 / 24 to subnet **r**. Then router **C**'s routing table would look like this:

IP Address	Send Packet to:	Send Packet from:
19.53.0.0 / 27	Direct to destination	Port c
19.53.0.0 / 24	Direct to destination	Port b
19.53.1.0 / 24	Router B	Port a
19.53.2.0 / 24	Direct to destination	Port a
Everywhere else	Router A	Port a

Now you've probably spotted that there's an interesting problem with this: the IPv4 addresses from 19.53.0.0 to 19.53.0.31 are now matched by both of the first two entries in this routing table. How does the router know where to send these packets? Answer: to the destination with the longest bit mask. In this case, that means they go out of port c, since this entry in the table has a longer bit mask (27 bits, as opposed to the second entry in the table, that only has a 24 bit long bit mask). The IPv4 addresses from 19.53.0.32 to 19.53.0.255 only match the second entry, so they get sent out of port b.

This makes it very easy to represent the entry for "everywhere else", you can represent this range as "0.0.0.0 / 0". All IPv4 addresses lie in this range: all that is required is that the top zero bits in the address are equal to 0.0.0.0, which is always true. However, since the bit-mask is zero bits long, this is a routing table entry of last resort. If a packet's destination address matches any other entry in the routing table, it will not be used. This gives a final routing table of:

IP Address	Send Packet to:	Send Packet from:
19.53.0.0 / 27	Direct to destination	Port c
19.53.0.0 / 24	Direct to destination	Port b
19.53.1.0 / 24	Router B	Port a
19.53.2.0 / 24	Direct to destination	Port a
0.0.0.0 / 0	Router A	Port a

You might also have spotted some other potential problems: for a start, the network addresses for subnet **r** and subnet **s** are the same: 19.53.0.0. However, since a packet is never sent to the network address, and networks are identified by the combination of their network address and bit-mask, there is never any confusion here. Also, the directed broadcast address on subnet **s** is 19.53.0.31, and this is a valid IPv4 address that could be assigned to a node on subnet **r**. Again, this isn't a problem in practice, as long as you don't assign this address to any node on subnet **r**.

In fact, this method actually saves an IPv4 address: since the network address of these two networks is now the same, there are only three IPv4 addresses in the range 19.53.0.0 to 19.53.0.255 that cannot be assigned to an individual node: 19.53.0.0 (network address), 19.53.0.31 (broadcast address for subnet **s**) and 19.53.0.255 (broadcast address for subnet **r**).

Admittedly, this is rather inelegant and a bit confusing, and if you've got enough IPv4 addresses, it would be much better to assign the subnets the same number of IPv4 addresses. IPv6 largely avoids this problem; there are so many IPv6 addresses you don't have to be so careful about preserving them.

1.2.3 A Look at Windows

If you look inside Windows, you'll find that the node knows the local submask as well as the IP address. My windows IP Address control box looks like this:

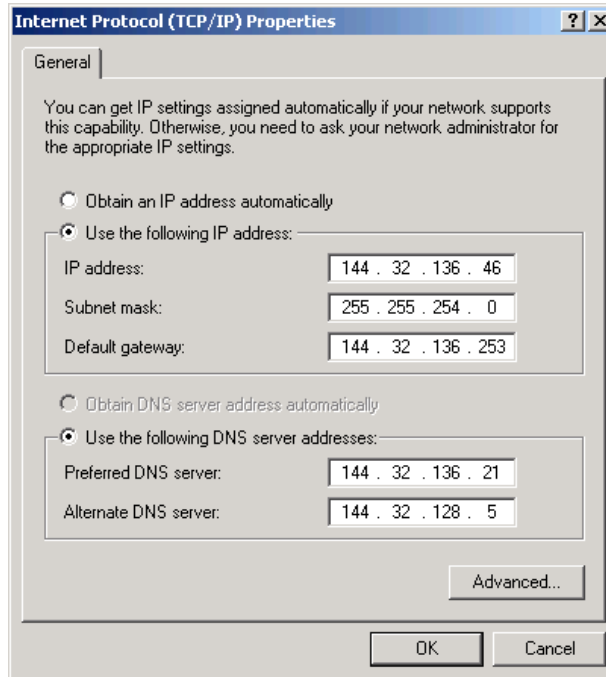


Figure 1-6 – Setting-Up IPv4 on a Windows 2000 Computer

Why would the end station in your computer need to know the local submask? Because it needs to know what the local broadcast address is. This subnet has a 23 bit mask, which means the local broadcast address ends with $32 - 23 = 9$ ones. That makes the local broadcast address 144.32.137.255. This node should receive packets addressed to 144.32.137.255, but not, for example, packets addressed to 144.32.136.255. (144.32.136.255 is a valid host address on this subnet, and could be the IPv4 address being used by another computer.)

(If I ticked the “Obtain an IP address automatically” option, the machine would try and find out its own IPv4 address using DHCP. See later in the book for more details about DHCP).

If I open a command prompt, and enter the command “route print”, then I see this:

```

C:\WINNT\system32\cmd.exe
C:\Documents and Settings\dajp1>route PRINT
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x1000003 ...00 03 47 e8 55 6e ..... Intel(R) PRO/100 VE Network Connection
=====
Active Routes:
Network Destination    Netmask          Gateway          Interface        Metric
0.0.0.0                0.0.0.0          144.32.136.253   144.32.136.46    1
127.0.0.0              255.0.0.0        127.0.0.1        127.0.0.1        1
144.32.136.0           255.255.254.0    144.32.136.46   144.32.136.46    1
144.32.136.46          255.255.255.255  127.0.0.1        127.0.0.1        1
144.32.255.255         255.255.255.255  144.32.136.46   144.32.136.46    1
224.0.0.0              224.0.0.0        144.32.136.46   144.32.136.46    1
255.255.255.255       255.255.255.255  144.32.136.46   144.32.136.46    1
Default Gateway:      144.32.136.253
=====
Persistent Routes:
None
C:\Documents and Settings\dajp1>

```

Figure 1-7 – A Routeing Table in a Windows Machine

Note that the default route is 0.0.0.0 with a network bit-mask (netmask) of 0.0.0.0 as expected, and all packets that don't match any of the other entries will be sent out of interface 144.32.136.46 (this computer's IP address, associated with the Ethernet card in my PC) towards 144.32.136.253 (the default gateway¹⁶).

Key Points

- Originally, IPv4 addresses consisted of 8-bit network identifiers (net-IDs) and 24-bit host identifiers (host-IDs), and that the demand for more networks led to class-based addressing; although this did not provide a very good solution, and these days the Internet uses classless addressing.
- IPv4 addresses are conventionally written in dotted-decimal format, and ranges of IPv4 addresses are written in the X.X.X.X/Y format where Y is the length of the network bit mask.
- For the range X.X.X.X / Y, the network address is X.X.X.X, and consists of the top Y bits of the IPv4 addresses of any node on the network, and the bottom (32 – Y) bits are zero. This network address can never be assigned to a node.
- For the range X.X.X.X / Y, the broadcast address has the bottom (32 – Y) bits all set to one, and packets sent to this destination address are received by all nodes on this network.
- The range X.X.X.X / Y, can accommodate $2^{(32 - Y)} - 2$ nodes: the highest and lowest addresses in this range cannot be used for individual nodes.

¹⁶ A gateway is a router that acts as the link between one network and another: in this case the one going outside the local subnet. The two terms ('router' and 'gateway') are often used almost interchangeably, although strictly speaking some routers are not gateways: there is a router function inside every single node on the Internet (for example, see the routing table in the PC in the figure above), although my computer is not a gateway: it doesn't provide a link between two networks.

1.4 Tutorial Questions

**1) Go and have a look at the RIPE web site, (<http://www.ripe.net>) and find out which IP addresses have been assigned to the Holy See in the Vatican (country code 'va'). How many IP addresses does the Pope control? What's the maximum number of host computers in the Vatican?

2) Consider the situation if only class A, class B or class C addresses were available. Assume everyone who wanted a class A address had one million hosts, if everyone who wanted a class B address had one thousand hosts, and everyone who wanted a class C address had ten hosts. How many hosts would there be on the Internet when all available address space had been allocated? What proportion of the total available address space does this make?

3) Which of the following addresses lies in the range 12.62.148.128 / 26?

12.62.149.15
12.62.149.24
12.62.149.204
12.62.148.160

4) Which one of the following ranges matches 237.223.131.179?

237.223.224.0 / 19
237.218.0.0 / 15
237.223.0.0 / 17
237.223.128.0 / 17

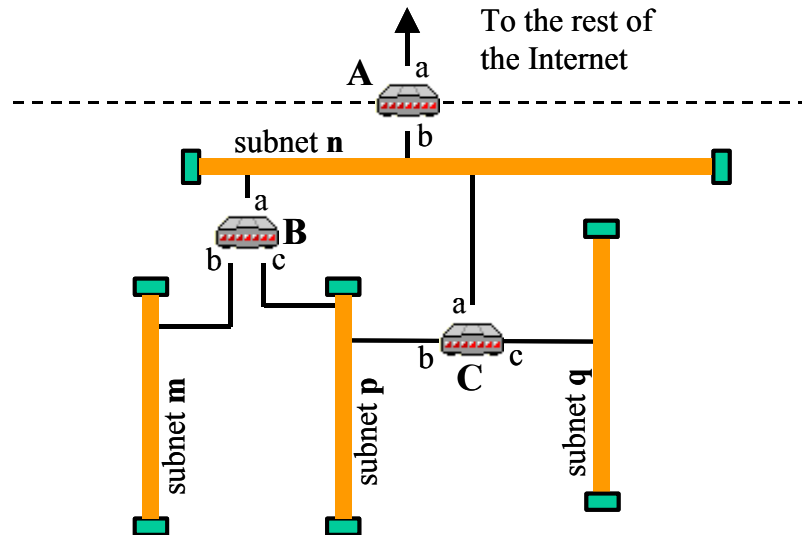
5) Which one of the following ranges contains both 139.123.124.158 and 139.123.124.128?

139.123.124.128 / 27
139.123.124.32 / 27
139.123.124.160 / 28
139.123.124.0 / 27

6) Which one of the following ranges contains 165.218 / 15?

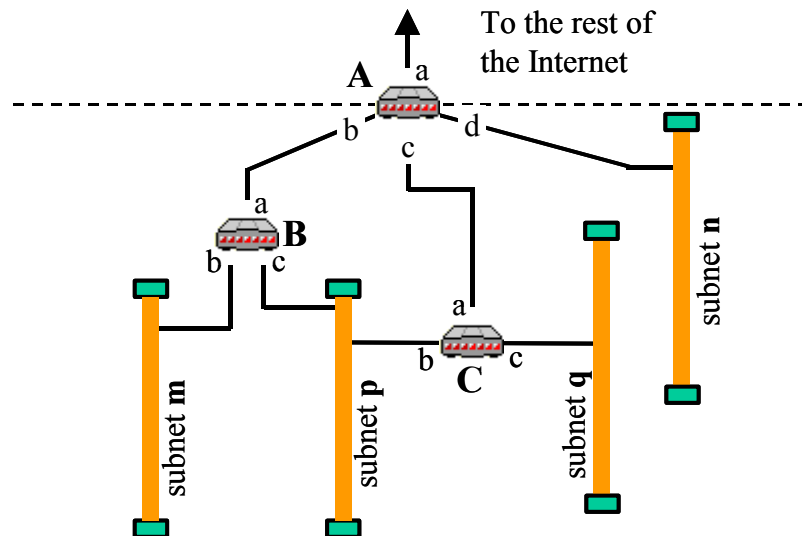
165.212 / 14
165.216 / 14
165.220 / 16
165.192 / 13

*7) A network administrator obtains the rights to use the range of IPv4 addresses 203.132.52 / 22. He builds the network shown in the diagram below, with four subnets. If there are 100 nodes on each subnet, assign the IP addresses to the various subnets in a sensible way, and give a possible set of entries for the routing tables in the three routers shown.



*8) Revisit question 7), but now consider the case where there are 180 users on subnet **m**, 280 users on subnet **p**, 200 users on subnet **q** and 73 users on subnet **n**. How could you assign IPv4 addresses to the four subnets now?

*9) A user with an allocation of 130.26.184.0 / 23 builds a network with the following topology:



If there are 180 nodes on subnet **m**, 50 on subnet **p**, 40 on subnet **q** and 60 on subnet **n**, suggest a suitable allocation of IPv4 addresses to the various subnets, and deduce a simple routing table for router **A**.

*10) The original RFC describing subnetting did not allow the first or last subnet in any network range to be used, whereas most modern routers are quite happy to assign these addresses to subnetworks. What changed? Why were these subnets not allowed in the first place?