

The University of Queensland
School of Information Technology and Electrical Engineering
Semester Two, 2009

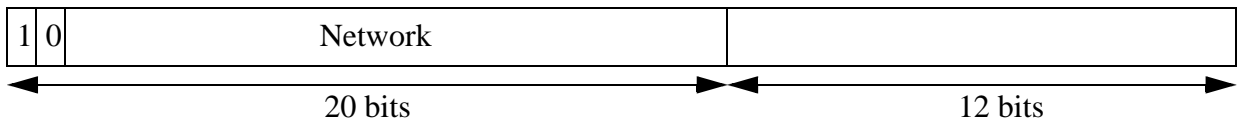
COMS3200 – Tutorial 8 - Solutions

Question 1

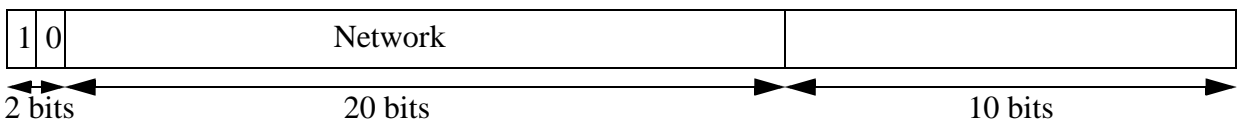
Suppose that instead of using 16 bits for the network part of a class B address, 20 bits had been used. How many class B networks would there have been? How many hosts could be placed on such a class B network?

In reality there are only 14 bits for the network part of class B address - the first two bits are always 10. Given this ambiguity, there are two possible ways to answer this question - either including or excluding the initial two bits.

If we include the initial two bits in the 20, there would be $2^{18}-2$ (=262142) possible networks with $2^{12}-2$ (=4094) possible hosts:



If we exclude the initial two bits, there would be $2^{20}-2$ (=1048574) possible networks with $2^{10}-2$ (=1022) possible hosts:



Remember, we subtract two from the number of possibilities because addresses with all 0's and all 1's have special meaning.

Question 2

Convert the IP address whose hexadecimal representation is C22F1582 to dotted decimal notation. What class of network does this belong to?

$$C22F1582 = C2.2F.15.82 = 194.47.21.130$$

This address is in a class C network. (Class C networks are those between 192.0.0.0 and 223.255.255.255). The network is 194.47.21.0 and the host number is 130.

Question 3

A class B network on the Internet has a subnet mask of 255.255.240.0. What is the maximum number of hosts per subnet?

In binary, this mask corresponds to: 11111111.11111111.11110000.00000000. We have 12 bits available to represent the host number, therefore the maximum number of hosts per subnet is $2^{12}-2 = 4094$.

Question 4

Describe a way to do reassembly of IP fragments at the destination.

(This is very similar to the method used by the BSD kernel - and probably others.)

A data structure is kept of arriving datagram fragments. For efficiency, this is a hash-table keyed on some combination of the fields from the datagram header which make the datagram unique.

These are

- source address

- destination address
- identification
- protocol

(BSD hashes on source address and identification field.)

Each entry in the hash table is a list of fragment queues (remember fragments from different original datagrams may hash to the same entry in the hash table).

When a fragment arrives, the matching list of fragment queues in the hash table is checked for a fragment queue that matches on **all** the identifying information (as listed above). If one is not found, a new fragment queue is created and added to that hash table entry. If a matching fragment queue is found, this fragment is added to the queue (in an order as determined by the fragment offset). The queue is checked to see whether all fragments of that datagram have now arrived - if so, they are reassembled in an appropriate buffer and passed to the appropriate piece of software (as determined by the protocol field, e.g. TCP).

If all fragments don't arrive before some timeout, then the fragment queue will be deleted (see Question 9).

IP datagrams that arrive without fragmentation can be passed directly to the higher level software (e.g. TCP).

Question 5

Most IP datagram reassembly algorithms have a timer to avoid having a lost fragment tie up reassembly buffers forever. Suppose a datagram is fragmented into four fragments. The first three fragments arrive, but the last one is delayed. Eventually the timer goes off and the three fragments in the receiver's memory are discarded. A little later, the last fragment stumbles in. What should be done with it?

The obvious answer is that it should be thrown away immediately. Since the other fragments have already been discarded there is no way we can reconstruct the complete datagram.

In reality, the fragment is likely to be kept initially. The IP reassembly code will most likely assume that this is the first fragment to arrive of some fragmented datagram. It will keep the fragment and wait for the others to arrive. When they do not, and the timer goes off, the fragment will be discarded.

(This question serves to illustrate that IP datagram identification numbers are not treated the same way as TCP segment sequence numbers. Whilst IP datagram identification numbers may be generated sequentially, they need not be, and the receiver does not check for this. See Question 8 for an explanation of how this might work.)

Question 6

What aspect of IP addresses makes it necessary to have one address per network interface, rather than just one per host?

IP addresses include a network/subnet number so interfaces on different networks *must* have different network/subnet portions of the address.

Question 7

Why does the Offset field in the IP header measure the offset in 8-byte units? (Hint: recall that the Offset field is 13 bits long.)

The IPv4 header allocates only 13 bits to the Offset field, but a packet's length can be up to $2^{16} - 1$. In order to support fragmentation of a maximum-sized packet, we must count offsets in multiples of $2^{16-13} = 2^3 = 8$ bytes.

The only concerns with counting fragmentation offsets in 8-byte units are that we would waste space on a network with an MTU not a multiple of 8, or that alignment on 8-byte boundaries would prove inconvenient. 8-byte chunks are small enough that neither of these is a significant

concern.

Question 8

Suppose that a TCP message that contains 2048 bytes of data and 20 bytes of TCP header is passed to IP for delivery across two networks on the Internet (i.e., from the source host to a router to the destination host). The first network has an MTU of 1024 bytes; the second has an MTU of 512 bytes. (The MTU is the largest IP frame payload). Give the sizes and offsets of the sequence of fragments delivered to the network layer at the destination host. Assume all IP headers are 20 bytes.

Consider the first network. The frame payload can be up to 1024 bytes (the MTU), which means up to 1004 bytes of IP datagram data (because we have 20 bytes of IP header). Because 1004 is not a multiple of 8, the maximum that a fragment¹ can contain is 1000 bytes. We need to transfer $2048 + 20 = 2068$ bytes of data. (The IP datagram data is the complete TCP packet - 20 bytes of TCP header plus 2048 bytes of data.) 2068 bytes of data can be broken into fragments of 1000, 1000 and 68. The sizes and offsets of the 3 fragments would be

Fragment 1: 1020 bytes, offset field value = 0

Fragment 2: 1020 bytes, offset field value = 125

Fragment 3: 88 bytes, offset field value = 250

(Remember, the packet size is data plus 20 byte of IP header. Fragment offsets are divided by 8 to fit in the 13 bit fragment offset field.)

Over the second network, we can transmit up to 492 bytes of IP data ($512 - 20$). Because 492 is not a multiple of 8, the maximum a fragment can contain is 488 bytes. The fragment of 88 bytes will not be fragmented further. The fragments of 1020 bytes WILL be fragmented further. Their 1000 bytes of data will be placed in fragments of size 488, 488 and 24.

The fragments arriving at the destination should be (not necessarily in this order of course):

Fragment 1: $488 + 20 = 508$ bytes, offset field value = $0/8 = 0$

Fragment 2: $488 + 20 = 508$ bytes, offset field value = $488/8 = 61$

Fragment 3: $24 + 20 = 44$ bytes, offset field value = $61 + 488/8 = 122$

Fragment 4: $488 + 20 = 508$ bytes, offset field value = $122 + 24/8 = 125$

Fragment 5: $488 + 20 = 508$ bytes, offset field value = $125 + 488/8 = 186$

Fragment 6: $24 + 20 = 44$ bytes, offset field value = $186 + 488/8 = 247$

Fragment 7: 88 bytes, offset field value = 250 (as above, not fragmented further)

Question 9

What is the maximum bandwidth at which an IP host can send 576-byte packets without having the Ident field wrap around within 60 seconds? Suppose IP's maximum segment lifetime (MSL) is 60 seconds; that is, delayed packets can arrive up to 60 seconds late but no later. What might happen if this bandwidth were exceeded?

The Ident field is 16 bits so we have $2^{16} = 65536$ possible Ident values. If each packet within 60 seconds has a unique Ident value, we can send at most 65536 packets in 60 seconds, i.e. we can send 576×65536 bytes per 60 seconds, or about 5Mbps. (Remember, there are 8 bits in a byte!) If we send more than this, then fragments of one packet could conceivably have the same Ident value as fragments of another packet and could be incorrectly reassembled.

1. Note that the **last** fragment could contain 1004 bytes of data (because you don't have to represent an offset of 1004 in the next fragment because there isn't one!). Earlier fragments data sizes must be a multiple of 8.