Module 3F5: Computer and Network Systems

Solutions to 2015 Tripos Paper

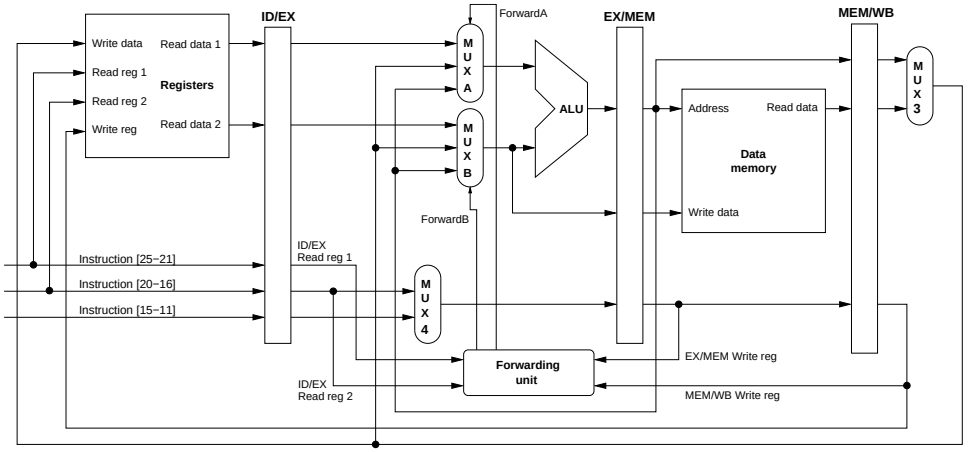
Authors: Andrew Gee and Tim Wilkinson

1. **Datapaths and pipelining**

(a) A pipelined datapath features extra registers between the principal datapath stages. In each clock cycle, instructions advance through just one stage of the datapath, writing their interim results into the pipeline registers. In this way, several instructions can be in the pipeline at the same time, one at each stage. Pipelining therefore increases instruction throughput.

The term *hazard* is used to describe dependencies between instructions which disrupt the operation of a pipelined datapath. *Data hazards* occur when an instruction requires data before a previous instruction has written it to the register file. *Branch hazards* occur when the address of the next instruction is required (for instruction fetching) before an earlier conditional branch instruction has been evaluated. [20%]

(b)



The forwarding unit compares the registers which have just been read ("read reg 1" and "read reg 2") with any registers about to be written by the two downstream instructions. If they are the same, it sets MUXA and/or MUXB to ignore the value read from the register file and instead use the appropriate forwarded value. [20%]

(c) (i) Forwarding cannot completely resolve the data hazard between the `lw` and the `add`, so there will have to be one stall here. We will also need to flush three instructions after the `bne`, apart from on the last iteration when the branch is not taken. Allowing for the initial `add` instruction and also four cycles at the end for the pipeline to clear, the code requires $1 + 9(n-1) + 6 + 4 = 9n + 2$ cycles. [10%]

(ii) We can avoid stalling between the `lw` and the `add` as follows:

```
        add $9,$0,$0        # clear $9 to zero
Loop:   lw $8,Astart($9)    # $8 loaded with data at address $9+Astart
        addi $9,$9,4        # $9 loaded with $9+4
        add $8,$8,$10       # $8 loaded with $8+$10
        sw $8,Astart-4($9)  # $8 stored at address $9+Astart-4
        bne $9,$11,Loop     # Jump back 4 instructions if $9≠$11
```

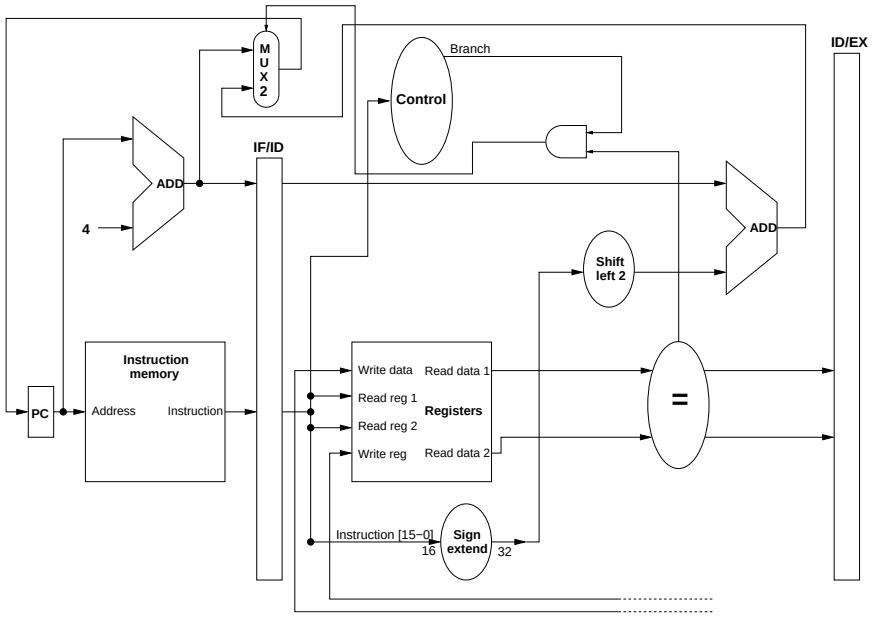
This code requires $1 + 8(n-1) + 5 + 4 = 8n + 2$ cycles. [15%]

(iii) Alternatively, we can avoid the stall between the `lw` and the `add` by unrolling the loop. This also reduces the number of branch flushes.

```
       add $9,$0,$0         # clear $9 to zero
Loop:  lw $8,Astart($9)     # $8 loaded with data at address $9+Astart
       lw $12,Astart+4($9)  # $12 loaded with data at address $9+Astart+4
       add $8,$8,$10        # $8 loaded with $8+$10
       add $12,$12,$10      # $12 loaded with $12+$10
       sw $8,Astart($9)     # $8 stored at address $9+Astart
       sw $12,Astart+4($9)  # $12 stored at address $9+Astart+4
       addi $9,$9,8         # $9 loaded with $9+8
       bne $9,$11,Loop      # Jump back 7 instructions if $9≠$11
```

This code requires $1 + 11(n/2-1) + 8 + 4 = 11n/2 + 2$ cycles. [15%]

(d)



In the modified datapath, the branch target calculation is moved from the EX stage to the ID stage. The registers are also compared at the ID stage. Only one subsequent instruction

(in the delay slot) has been fetched before the branch target is known. The code in (c) could schedule the `sw` in the delay slot:

```
       add $9,$0,$0       # clear $9 to zero
Loop:  lw $8,Astart($9)   # $8 loaded with data at address $9+Astart
       addi $9,$9,4       # $9 loaded with $9+4
       add $8,$8,$10      # $8 loaded with $8+$10
       bne $9,$11,Loop    # Delayed branch
       sw $8,Astart-4($9) # $8 stored at address $9+Astart-4
```

This code requires $1 + 5n + 4 = 5n + 5$ cycles. Loop unrolling would reduce the number of branches and hence speed up the code even more. [20%]

Assessors' remarks: This question tested the candidates' understanding of pipelining and hazards. (a) and (b) were essentially book work, (c) was similar to examples paper and past tripos questions, only (d) was somewhat unfamiliar. Almost all candidates answered (a) very well, demonstrating a sound understanding of the core concepts. Things went downhill rapidly from that point, with few candidates able to sketch how the forwarding unit connects to the rest of the datapath in (b), and only around half of the candidates realising that forwarding eliminates all but one of the data hazards in (c)(i). There were very few sensible insights into how delayed branches might work in (d).

2. **I/O and DMA**

(a) Polling, interrupt-driven I/O and direct memory access (DMA) are three different mechanisms for allowing the CPU to interact with I/O devices. Polling requires the least hardware: the CPU periodically checks to see whether the device is ready to send or receive more data, and handles the data transaction if necessary. The polling frequency must be high enough to satisfy the device's maximum data transfer rate. This can be tremendously wasteful of CPU time, especially for devices which are mostly idle. Polling may be used for low bandwidth devices which can tolerate low frequency polling, like mice.

Interrupt-driven I/O requires extra signal lines to interrupt the CPU whenever an I/O device requires attention. The CPU must still be involved in every bus transaction, so may still be heavily loaded when the device is active. But, in contrast to polling, there is no CPU load when the device is idle. Interrupt driven I/O may be used for relatively low bandwidth devices which are mostly idle, like printers.

DMA is the most expensive technique in terms of hardware, requiring a dedicated DMA controller. But it is the only viable technique for very high bandwidth devices, like hard disks, which might otherwise fully occupy the CPU with bus transfers. With DMA, the CPU hands control to the DMA controller, which deals with the individual bus transactions between the device and memory. Once the transfer is complete, the CPU is interrupted. The CPU then checks whether the transfer was completed successfully or whether there was an error. [25%]

(b) DMA poses problems for both virtual memory systems and caches. Starting with virtual memory, the DMA controller is supplied with a starting address and a number of bytes to transfer. But should the starting address be virtual or physical? If virtual, then the DMA controller will need extra hardware to store the necessary page table entries and perform the translations. If physical, then care must be taken to ensure that DMA transfers do not cross page boundaries, since contiguous virtual pages do not generally map to contiguous physical pages. Whichever approach is taken, the operating system must cooperate by not moving pages around while a DMA transfer involving that page is in progress.

Moving on to caches, there are two potential problems here. When transferring data from the I/O device to memory, there may be a copy of this chunk of memory in the cache. If the processor reads from this chunk, it will get the old value (from the cache) and not the new value (as updated by the DMA controller). Similarly, when DMA is used to transfer data from memory to the I/O device, and the cache is write-back, the DMA controller may transfer an old value from memory when there is a newer one in the cache. This is called the *stale data problem*.

There are three ways round this. One possibility is to route all DMA activity through the cache, though this is expensive and very wasteful of cache space, since the processor rarely needs to see all the I/O data immediately and, in the meantime, useful data has been displaced from the cache. The second option is to have the operating system invalidate the cache for an I/O write or force write-backs for an I/O read. This sort of *cache flushing* requires minimal hardware support but is inefficient, since the whole cache is affected even if only one block overlaps with the DMA activity. The final, most complex option is to provide hardware mechanisms to selectively flush individual cache entries. [25%]

(c) This is to do with the gradual replacement of I/O buses (e.g. parallel ATA and PCI) with serial point-to-point networks (e.g. SATA and PCIe). In the old days of the shared I/O bus, only one device could talk to memory at a time, so one DMA controller was all that was needed. Now that we have largely replaced I/O buses with serial point-to-point networks, we can envisage concurrent DMA transfers involving multiple devices *at the same time*. It would take many simultaneous DMA transfers to saturate the bandwidth of the single memory system, so this is unlikely to be a problem. However, we would require a separate DMA controller for each device, and the neatest place to locate this is on the device itself. The obvious advantage is increased DMA bandwidth between the combined set of I/O devices and memory. [25%]

(d) For a gigabit NIC, a DMA transfer size of one packet could mean a very high rate of interrupts at the CPU. A larger DMA transfer of many packets would not reduce the amount of network data the CPU ultimately has to deal with, but it would reduce the number of interrupts. The efficiency saving comes from the considerable overhead of the interrupt itself (context switching, preservation of CPU registers before jumping to the interrupt service routine, accessing the interrupt registers to determine the source of the interrupt, etc). The downside is an increased latency between data arriving at the NIC and

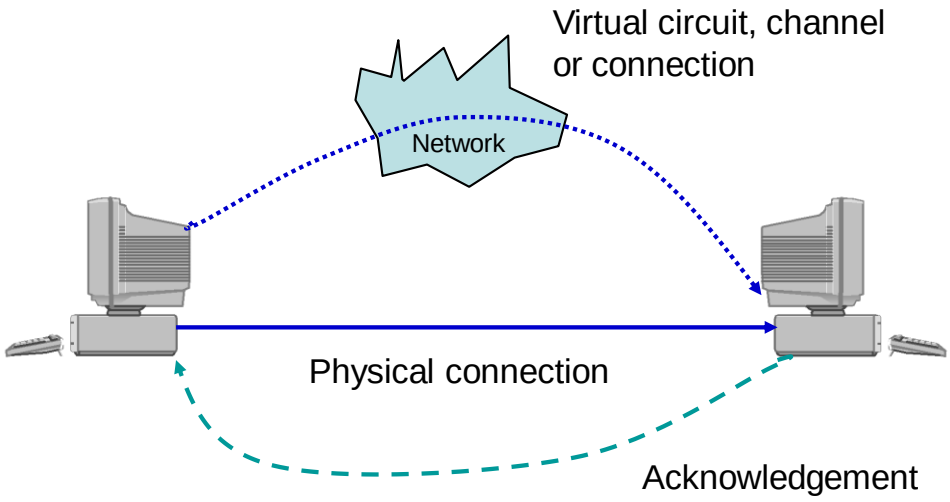
its availability for a user or kernel process. Setting the interrupt coalescing parameter T involves, essentially, a trade-off between high throughput and low latency. [25%]

Assessors' remarks: This question tested candidates' understanding of I/O and DMA. Part (a) was very well answered, with almost all candidates demonstrating a sound grasp of the basic concepts. Part (b) was also well answered, with most candidates understanding the potential problems with caches, though fewer were on top of the virtual memory issues. The more speculative parts of the question, (c) and (d), elicited a more varied set of responses, though it was pleasing to see many candidates identifying the essential throughput/latency trade-off in (d).

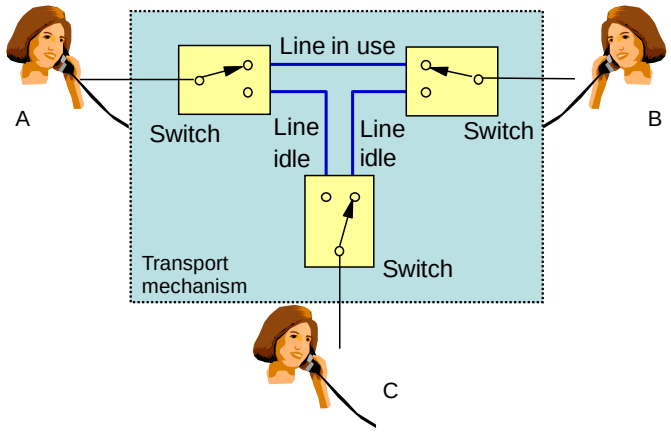
Q3 –Answers are more verbose than expected from the candidates

a) In a connection oriented technique a circuit, virtual circuit, connection or virtual connection (VC) is established between sender and receiver **before** information is transported. Thus, a telephone call is first connected by dialling the number to set up a circuit between caller and receiver. This connection ensures the readiness of the receiver to accept the information and the connection remains until the call is complete. Networks such as the SDH network and other circuit switched and most packet switched and cell switched networks are examples of connection oriented switching or connection oriented transport service (COTS).

A packet switched network is connection oriented as the users are connected by a virtual connection that is actually sent as a sequential line of packets through a physical media common with other users.



Below is an example of a COTS telephone network. If C tries to call A or B while they are connected, then C receives the engaged signal and has to wait. This is the penalty of the implementation with switches, but it suits telephony, where only one call can be easily handled by a user at one time. The switched network can be expanded by including extra telephones at each switch and more complex switching. This is an example of a circuit switched network.



In a real exchange, the numbers of exchanges and their locations are governed by the overall number and geographical density of the telephone users. The number of junctions or trunks will be

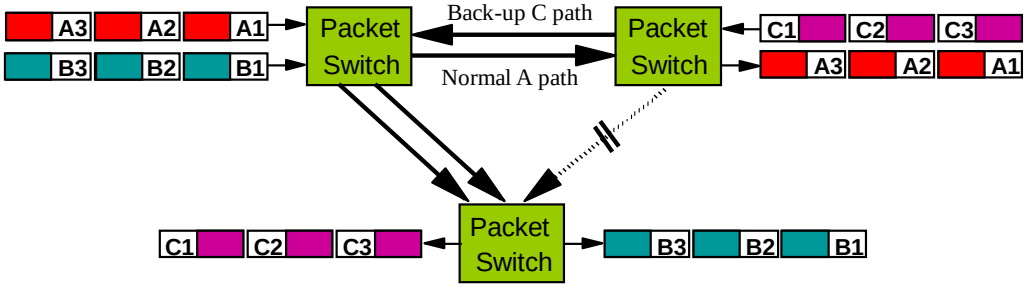
made sufficient to cater for normal telephone demand. The above network is classed as having full availability as any of the stations can be connected to any of the available junctions.

[A well answered section of mostly book work. Majority of candidates lost marks for not indicating the role of acknowledgement in COTS systems]

b) The main disadvantage of a COTS network service is the amount of time spent setting up a call or connection. Services such as packet switching are form of statistical multiplexing. In this way, the entire bandwidth can be used momentarily by any of the logical channels or virtual circuits (VC) sharing the data connection. Problems arise when more than one or all transmitters try to send packets at once. This is accommodated by buffers at each end of the connection. These delay some of the simultaneous packets in a first in first out (FIFO) system until the line becomes free. With buffers, it is possible to run links at up to 100% utilisation. At very close to full utilisation, very large buffers are required for each of the logical channels.

Queuing = buffers = delay (latency)

When switching with a COTS service, a fixed path is chosen for a given logical channel (virtual circuit) at the time of call set-up. The path itself is chosen based on the current loading of the network and the available topology. Should any link in the path become unavailable during the course of the call (say, because of transmission failure), then an alternative path is sought, without breaking the connection. Packets are stored until a new path is found. This is not very efficient use of network bandwidth. The advantage of COTS service switching is that the packets pertaining to a given local connection all take the same path, all suffering about the same amount of queuing delay in the buffers and arrive pretty much in the same order as they were sent (allowing for lost packets along the way). The disadvantage of path oriented routing is inflexibility at high traffic rates.



The Ethernet protocol evolved as a layer 2 driven LAN protocol. It used the layer 2 MAC/LLC spilt to separate the media properties from the control. The LLC as was originally defined by the 802.2 standard contained many control fields in the frame to allow frame numbering and windowing to be controlled from layer by dedicated hardware. The was very efficient and suited a COTS type service very well, however as hardware got faster and more control was taken over by layer 3 protocols like IP, the controls were disabled or ignored within the LLC. Hence Ethernet became dependent on layer 3 systems to run its services. Unfortunately IP is fundamentally a connectionless protocol, hence it no longer works a well with COTS services.

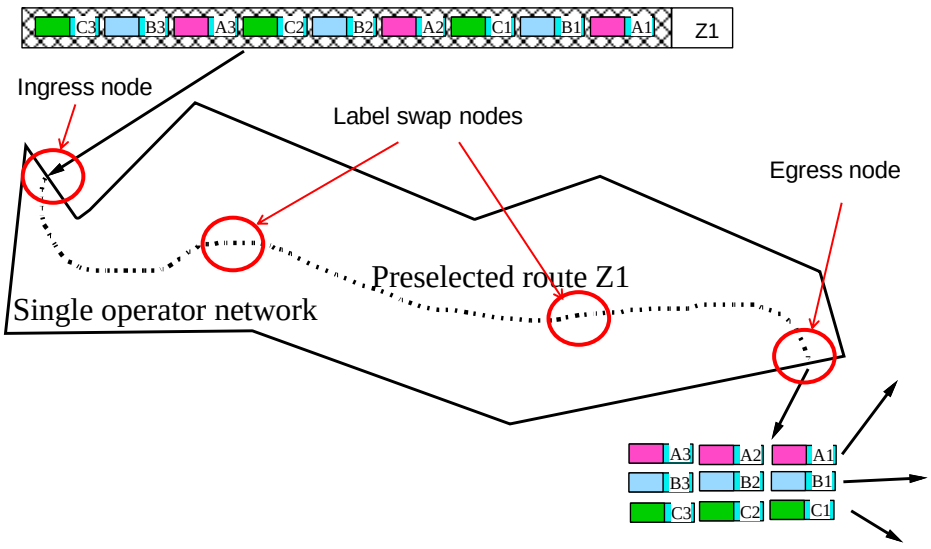
[*Also well answered in general with most indicating the advantages of COTS. All candidates missed the point of mentioning Ethernet, with most listing the loss of CSMA/CD to structured cabling as the main cause, this is a physical layer issue, not relating to COTS*]

c) Voice services are intolerant to any form of delay (fixed or variable). If there is any significant delay (more than 20-30msec) then reflections can lead to echoes being generated. Voice services are tolerant to transmission errors. Conversely, data services are highly sensitive to bit errors. Data networks are much more tolerant to delays in transmission. The real problem is how to transmit voice/video without delay or error? The vast majority of modern internet services are now in this category. VoIP, video on demand, Internet TV and convergence services etc.

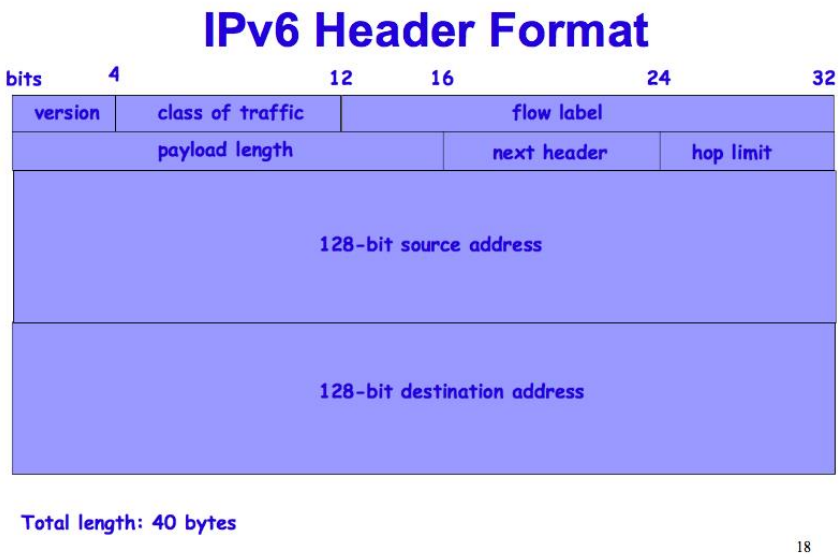
Voice services gave rise to the perfect COTS service in SDH has been designed for low latency and minimal delays, which is ideal for voice. Data tends to be bursty and has been suited to packet switched networks. This is more difficult to map onto SDH, especially compressed video streams which are bursty and delay sensitive. As a result protocols such as IP and Ethernet evolved specifically to take on the bursty nature of data traffic. This was fine for basic data services like web surfing, but more demanding services such a video are now more prevalent and modern networks are struggling to match the QoS demands for video, especially look forward to full HD and 4and 8k services. To deliver these types of services COTS oriented service is required. As a result, network providers have started developing new extension protocols to allow better QoS control. Early ideas such as Diffserv and others work well and offer the type of QoS desired, however they are all proprietary to manufacturers and require dedicated equipment across the entire network to function properly.

[*Most got the point of point of the QoS relationship with COTS but the majority of the answers used VoIP as the evolution which is purely a higher layer to IP therefore no QoS advantage. Reference to enhanced IP protocols were few and far between*.]

d) Multi-label protocol systems (MPLS). The process of routing packets cannot easily guarantee delay (latency) across the network. This is a problem for voice services across packet switched networks (such as VoIP). One way to fix this is to group packets through a common network together with a common global header (or label or tag) which gets the group of packets to the other side of the network with minimum delay. This puts a lot of pressure on the routers at the edge of the network to find suitable packet groups



The other main attempt to increase QoS was the change from IPV4 to IPV6. The IPV4 type of Service is now changed to Traffic Class and is designed to allow a propriety QoS to be implemented. The original semantics of the IPv4 Type of Service field have been superseded by the diffserv semantics per RFC 2474. However, in IPv4, both interpretations of the field are in use (although most routers either cannot or are not configured to look at the field anyway, hence intermediate nodes may not use this feature and add latency).

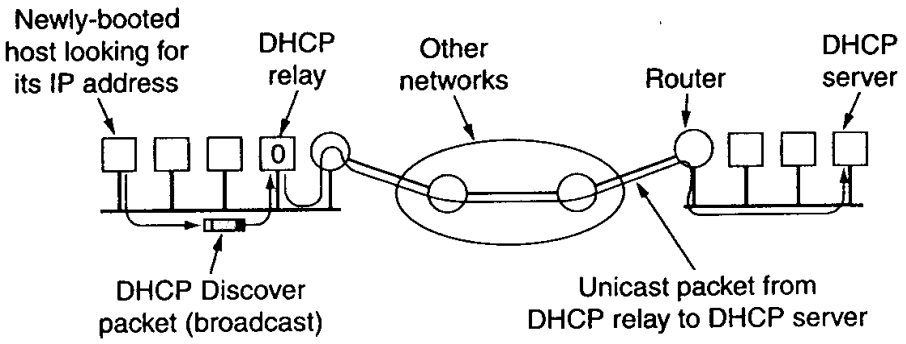


[*This was fairly poorly answered despite being in questions in previous years. The key to this section being to list 'recent' protocols. Most listed TCP (1970's) and ATM (1980's) which are not very recent. A few mentioned MPLS and no one mentioned IPV6 despite being a sub theme of the next question! One candidate listed 4G which is an interesting case, but really physical layer*.]

Q4 –Answers are more verbose than expected from the candidates

a) The source and destination addresses in IPv4 are both 32 bits, hence there is a limit on the number of things that can be connected to the IoT. Total address theoretical space is 2^{32} = 4294967296 $4.3x10^9$ addresses. Estimated world population is 6-7billion, hence there are not enough addresses for one thing per person, let alone multiple things. It is normal to write these addresses as four octets separated by dots (eg. 169.129.24.88). Each of the four decimal values may only have a value between 0 and 255. Also, the IPv4 address has two separate parts, the network identifier and the station identifier, so the address space is even further limited.

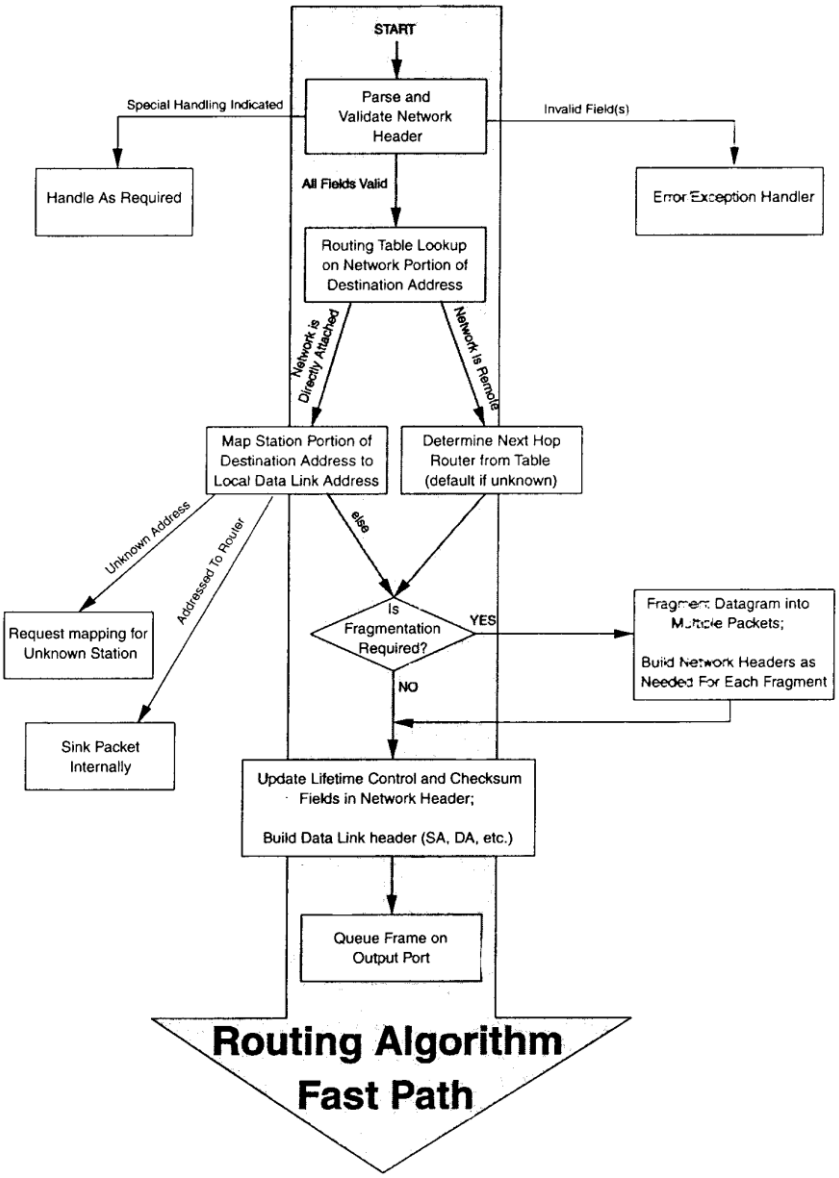
What happens when we run out of IPv4 addresses? A simple solution is to use the an updated version of the bootstrap protocol (BOOTP) which manually assigned IPv4 addresses, called the dynamic host configuration protocol (DCHP). DCHP allows both manual and automatic IP address assignment and has largely replaced both BOOTP and the reverse address resolution protocol (RARP).



This is effectively a concatenation of the MAC address with a locally assigned IP address. The network identifier still forms the global section of the overall IP address. Applications such as IoT could use DHCP to expand their address space, however the agility of the DHCP server must be very high as the number and flexibility of the users will be much larger with IoT users rather than static stations. The sort of approach used by WiFi networks would be best suited for IoT users.

[*Well answered section, a few could not remember the 32 bit length and several chose to ignore the fact that the problem was limited to IPV4. Many got DHCP and few spotted options such as BOOTP and RARP*]

b) A key concept on routing IP packets quickly is the Fast Path. Packets are vetted to see if they will delay those that follow them. Packets that use fragmentation or that involves the routing mechanism itself are taken off the fast path. ICMP packets use the options field in the IP packet to implement more sophisticated services. These are not included in the fast path. Neither are fragmented packets of any packets use for internal router and routing management.



IoT - advantage of the fast path is that it will process data with low latency, hence any applications such as voice and video will work efficiently. There will be some cost in terms of complexity as the of the processing makes each node complex.

A lot of applications will not require low latency, hence the fast path is not required and is a bit of a waste of technology. Another disadvantage which comes from the use of the fast path is that a lot of IoT traffic will only be local, making the fast path mostly redundant.

[*A little disappointing considering that the fast path has been asked before. Several confused it with datagram or adaptive routing. Not many were able to relate it to the IoT, especially the issue of local versus global traffic. One spotted the possibility of fixed paths*]

c) There are two main choices for mobile enabling devices for use with the IoT. For short range communication we can use one of the 802.12 protocols such as WiFi or Bluetooth. For longer range connectivity a mobile phone network like GSM or 3G would be suitable.

Wireless LANs (WLANs) are a type of shared media LAN similar to token ring and CSMA/CD Ethernet. A traditional configuration of a WLAN was to have as much area covered by base stations, with mobile stations positioned anywhere within the local area, this is ideal for mobile IoT applications. For small networks this works well, with a single base-station operating as a wireless access point, managing data and transmission channels between multiple stations. It is also possible to pass on some of the network management between users in an ad-hoc network where users can act as nodes in the network which would suite applications such a IoT enable devices. This type of operation is more complex, but greatly expands the range and usage of the wireless network. The IEEE 802.11 standard defines wireless LAN connections in two main frequency bands over relatively short distances, which have become part of the WiFi marketing standard and brand of products. The 802.11b is the international standard for wireless networking that operates in the 2.4 GHz frequency range (2.4 GHz to 2.4835 GHz) and provides a throughput of up to 11 Mbps. This is a very commonly used frequency. The 802.11a specification for wireless networking operates in the 5 GHz frequency range (5.725 GHz to 5.850 GHz) with a maximum of 54 Mbps data transfer rate. Another standard is the 802.11g series which offers up to 54Mbps in the 2.4GHz band (now up to 125Mbps in some cases). Now available is the 802.11n which uses multiple antenna/receiver combinations to increase the reliability and data rate further.

GSM (and 3G) are mobile network physical layer protocols which will allow IoT devices to connect together globally. GSM operates in a two frequency bands (900MHz and 1.8GHz). The available spectrum is broken up into 200kHz bands; within each band TDM is used to multiplex multiple users. GSM was originally designed for use in the 900MHz band, but later frequencies were allocated at 1.8GHz. Full GSM has up to 200 full duplex channels per cell; each channel has a down-link and an up-link frequency. Each of the 124 frequency channels supports eight separate connections using time division multiplexing (TDM). Each currently active station (mobile phone) is assigned one time-slot on one channel, theoretically 992 can be supported in each cell, but many of them are not available to avoid frequency conflicts with neighbouring cells. A single user can receive up to 9,600bps of data, but with enhancements such as GPRS and 3G, 4G then higher data rates are possible. The exact choice will depend on the IoT application.

The main advantage of using GSM is the global connectivity, without having to use intermediate nodes. The problem is that connection requires a SIMM card and some sort of network subscription. This is attractive to network providers, but not so attractive to customers due to cost. There is also a lot less flexibility in providing an IoT service over a network such as GSM.

[*Generally well answered, although several tried to compare WiFi with WLAN rather than GSM. A lot of answers had unnecessary waffle about hidden stations and RTS, MACAW. Very few spotted the IoT global versus local communication issue*]

d) This is a fairly open ended question, so any discussion made on the limits of bandwidth, addresses or other issues such a power and energy are what is expected. If addresses are chosen, then the scalability of MAC addresses could be an issue, or the problems with migrating to IPv6 and its global acceptance could also be important.

The bandwidth comments are more application dependent. If usage like video are envisaged, then it might be total bandwidth of volume of data required which is a limit. Or if the number of users is expanding, then it could be the overhead of routing many different users and their bursty traffic.

[*Quite well answered given the open nature of the section. Most stated address space or bandwidth as possible problems. Nobody thought of energy and power consumption issues*.]

Examiner's Comment:

Q1. This question tested the candidates' understanding of pipelining and hazards. (a) and (b) were essentially book work, (c) was similar to examples paper and past tripos questions, only (d) was somewhat unfamiliar. Almost all candidates answered (a) very well, demonstrating a sound understanding of the core concepts. Things went downhill rapidly from that point, with few candidates able to sketch how the forwarding unit connects to the rest of the datapath in (b), and only around half of the candidates realising that forwarding eliminates all but one of the data hazards in (c)(i). There were very few sensible insights into how delayed branches might work in (d).