COMPSCI 314 S1 C Assignment 4

Department of Computer Science

The University of Auckland

Due Wednesday 1 June 05, 11:59 pm

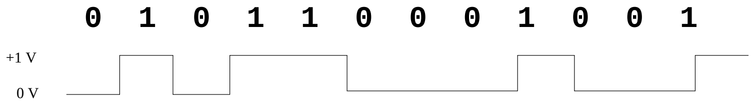
This assignment will contribute 40/am300 = 13.33% to your coursework mark, and 4% to your overall course mark.

Submit your assignment via the DropBox, either in PDF (preferred), or in MS Word format.

1. **Digital Encoding Schemes** [12 marks]

(a) Explain how the binary sequence 010110001001 would be encoded for transmission on a wire using *NRZ Encoding*. [3 marks]

High for a 1 (0 Volt on my diagram), zero for 0



My diagram was drawn with xfig – it seems to have a lack of clock synchronisation between the diagram and the text above it!

My lecture notes used High for 1, but the textbook uses low for a 1 – either is OK as a correct answer (but their explanation (in words) must match their diagram!

(b) If the sender and receiver clocks are not precisely synchronised, what effect will that have on a received NRZ signal? [3 marks]

Can’t determine where the bits start and finish, Rx will drift against Tx, resulting in incorrect reception of the data.

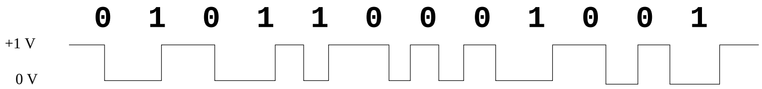
(c) Explain how the problem in (b) can be solved using a *differential* encoding scheme. [3 marks]

Differential encoding scheme uses transitions to indicate 1s or 0s, the transitions can be used to synchronise the Rx clock.

CONTINUED

(d) Give an example of a differential encoding scheme, and show how the binary sequence from (a) above would be encoded using it. [3 marks]

Manchester encoding, as used for 10Mb/s Ethernet, transitions at centre of cell to the value of that bit. Adjacent bits have a transition on the cell boundary, unlike bits have no transition between cells.



Shay describes Manchester as a self-synchronising *code. Differential codes (e.g. NRZ-I) use transitions to indicate 1s and 0s, they are more resistant to noise, and not sensitive to signal polarity. For this question, Manchester or Differential Manchester are good answers; both have plenty of transitions so that a receiver can recover the clock.*

2. **Internet Protocol** [13 marks]

(a) What is meant by *Classless* IP addressing? What problem(s) does it solve compared with *Class-based* addressing? [4 marks]

CIDR allows allocation of a network number with any number of bits, Class-based only allowed network number fields with 8,16 or 24 bits. When using CIDR, all hosts must know the number of bits, so routing protocols must carry it along with the network address, e.g.!130.216.0.0/16

CIDR solves – or at least ameliorates – the problem of our wasting IPv4 address space by inefficient allocation. If a network has 1000 hosts, we can allocate it a /22 network number; before we'd have had to give it a /16 (Class B), where only 1000 of the possible 65535 addresses would be used.

(b) Explain briefly how an IP host discovers the Ethernet address of a destination IP host. [3 marks]

Send ARP request packet containing destination IP address, and source (host's) IP address.

Destination host sees packet, sends back packet with destination host's IP address.

Hosts maintain 'ARP caches,' i.e. lists of Ethernet addresses for recently-used hosts; that minimises the number of times they need to discover Ethernet addresses using ARP.

(c) What are the *Time To Live (TTL)* and *Protocol* fields in the IPv4 header? How are they used? [3 marks]

TTL = seconds packet may continue to exist in the Internet
Protocol = higher-layer protocol used for payload of this IP packet

TTL is decremented by each router, packet is discarded when TTL reaches zero. Receiving IP stack uses Protocol as part of deciding how to get the packet's payload to the correct application program (i.e. process on the host).

(d) What are the IPv6 equivalents of the two fields in (c) above? Comment on any differences in the way IPv6 uses them, compared with IPv4. [3 marks]

Next Header, Hop Limit

Next Header is an extension of the Protocol field, used for chaining extension headers; last EH specifies protocol of IPv6 payload.
Hop Limit is simply a more accurate name for TTL.

3. **UDP and TCP** [15 marks]

(a) Give brief descriptions of the service provided by UDP and TCP. Your answer should cover the kinds of data each protocol carries, and the way in which that data is moved from one host to another. [4 marks]

UDP ⇒ Unreliable *datagram* transport TCP ⇒ Reliable *byte* transport

TCP provides congestion management, UDP does not.

(b) Which of these two protocols would be more suitable for each of the following applications? Explain (briefly) your answers.

- Domain Name lookup
- Remote Login
- File transfer
- Streamed Audio (e.g. from an Internet ‘radio station’

[4 marks]

DNS: UDP, client sends UDP request to server and waits for response. If it times out, it tries another server. Very simple exchange (2 packets), no need for TCP overhead. *DNS uses TCP for zone downloads, where there is a lot of data – but not for normal ‘name lookups.’*

Login: ssh uses TCP. Need all bytes transfereed in correct order (in both directions).

File transfer uses TCP. Again, need all bytes in correct order.

Streamed Audio (e.g. RealAudio): USP is fine, since we don;t care if we loose some data, that just makes s hort gaps in the sound, which we can tolerate. *Also, we probably don’t want to hold large buffers of data ubntil it’s ACKed!*

(c) What are *ports* in UDP and TCP? How are they used in carrying data from one application to another? [3 marks]

A ‘port’ is a number used to identify a final destination – i.e. a process within a receiving host – for an IP packet.

An IP stack uses the *Destination Port* number for each incoming packet to decide which process on the receiving host will handle the incoming TCP or UDP packet (especially it’s payload).

Similarly, the *Source Port* is used to determine which port the process should send a reply back to on the host that originated the incoming packet.

(d) How are ports identified in IP headers? What ports does FTP (the File Transfer Protocol) normally use? Could FTP use other ports? If so, how? [4 marks]

The IP header doesn't have port-number fields, they are in the transport-layer headers, i.e. the UDP and TCP headers.

Port numbers are 16-bit integers, 0 to 65535. Low-numbered ports are 'well-known,' i.e. always used as destination port by particular applications.

FTP uses port 21 for control messages (get, put, etc.), and port 20 for data transfers.

FTP can use other ports for the data transfer – this is called 'passive FTP.' In this, a client uses the *pasv* command to tell the server what port to send data back to.