Friday, May 16, 2008
2:30p.m. – 4:30p.m.
Duration: 2 hrs

University of Glasgow

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

COMPUTING SCIENCE 3T:
NETWORKED SYSTEMS ARCHITECTURE 3

(Answer all 3 questions)

This examination paper is worth a total of 60 marks.

1. **(a)** At the network layer, the Internet is a connectionless packet network, providing a best-effort packet delivery service. By way of contrast, the traditional telephone network provides a reliable circuit switched service. Discuss what are the advantages and disadvantages of these two approaches to network design.

[6]

(b) Many of the key design decisions of the Internet were influenced by the *end-to-end argument*. With the aid of an example, briefly describe the end-to-end argument as it applies to network protocol design.

[4]

(c) The Transmission Control Protocol (TCP) is designed to provide end-to-end reliable delivery of data across an unreliable Internet Protocol (IP) network. To achieve this, each transmitted data packet contains a sequence number, and the receiver sends cumulative positive acknowledgements for packets as they are received. The sender uses these acknowledgements to decide which packets have been lost and should be retransmitted. Using diagrams to show transmission and retransmission of data packets and acknowledgements, illustrate how TCP behaves when:

(i) a single packet is lost in transit; and

(ii) when two consecutive packets are reordered in transit.

Describe how TCP distinguishes these events, based on the acknowledgements.

[7]

(e) TCP uses a sliding window scheme to provide congestion control, adapting the window size to match the capacity of the network. If you were to use TCP to download a large file from a server located on a high bandwidth network in New York to your home computer in Glasgow, what would be the optimal window size for TCP to choose, in bytes? Assume a network round trip time of approximately 100ms, and a home ADSL connection with 8Mbps downlink speed and 1Mbps uplink speed. Explain how you arrived at your answer.

[3]

2. **(a)** A key function of the data link layer in the OSI reference model is *framing*: converting the raw bit-stream provided by the physical layer into a structured communication channel which can be used by the upper layers. In order to determine the start of a frame, many data link layer implementations use a *start code* comprising the binary pattern 01111110. With reference to the properties of the physical layer, explain why this bit pattern is commonly chosen as the start code.

[4]

(b) It is essential that the start code only appear at the beginning of a data link layer frame, and not within the data. Discuss why this limitation exists, and explain how the data link layer can prevent the start code occurring within the data, while still allowing arbitrary data to be transferred.

[8]

(c) Before data link layer frames can be used, it is necessary to check those frames for errors that may have occurred during transmission. This check can be done in a number of different ways, for example using a *parity code*, a *checksum*, or a *cyclic redundancy check (CRC)*, depending on the likelihood of errors.

(i) Discuss what physical processes might cause transmission errors, and how these might affect the bit stream.

[2]

(ii) Explain what is a parity code, how it can be calculated, and how it can be used to detect transmission errors.

[4]

(iii) Outline the advantages and disadvantages of using a parity code to detect transmission errors, compared to using a checksum or a CRC.

[2]

3. **(a)** You have written a basic web server program in the laboratory sessions for this course, using the C programming language, and the Berkeley Sockets API. Outline the design of your web server, explaining the high-level structure of your code, and highlighting how it uses the functions in the Berkeley Sockets API.

[12]

(b) The Berkeley Sockets API makes use of the following three `structs` to pass address information to the `connect()` and `bind()` functions:

```
struct sockaddr {
   uint8_t           sa_len;
   sa_family_t       sa_family;
   char              sa_data[22];
};

struct sockaddr_in {
   uint8_t           sin_len;
   sa_family_t       sin_family;
   in_port_t         sin_port;
   struct in_addr    sin_addr;
   char              sin_pad[16];
};

struct sockaddr_in6 {
   uint8_t           sin6_len;
   sa_family_t       sin6_family;
   in_port_t         sin6_port;
   uint32_t          sin6_flowinfo;
   struct in6_addr   sin6_addr;
};
```

A program using the Sockets API will create either a `struct sockaddr_in` or a `struct sockaddr_in6`, but must pass a pointer to a `struct sockaddr` to the `connect()` or `bind()` functions, casting as appropriate. Explain how and why this works.

[6]

(c) An alternative design for the Sockets API might use a `union` instead of casting between different types of `struct`. Outline how the data structures shown in part (b) might look, if such a `union` were to be used.

[2]