DUBLIN INSTITUTE OF TECHNOLOGY

DT228 BSc. (Honours) Degree in Computer Science

DT282 BSc. (Honours) Degree in Computer Science (International)

Year 3

WINTER EXAMINATIONS 2016/2017

CLIENT SERVER PROGRAMMING [CMPU3006]

MR. DAMIAN BOURKE
DR. DEIRDRE LILLIS
MR. PAUL COLLINS

THURSDAY 12TH JANUARY 9.30 A.M. – 11.30 A.M.

TWO HOURS

INSTRUCTIONS TO CANDIDATES

ATTEMPT SECTION A
AND ANY FOUR QUESTIONS FROM SECTION B.
SECTION A CARRIES 40 MARKS AND
SECTION B CARRIES 60 MARKS.

Section A

1. Refer to Figure 1 – *A snippet of code*. There are thirteen deliberate errors introduced into this code; ten of the errors relate to the incorrect use of: arguments or incorrect return values and, three relate to the sequencing of the Socket Primitive calls.

   You are required to:

   (i) Identify which arguments and/or return values are incorrect and suggest the correct argument/return value to be used. Use the line numbers to identify where your corrections are to be made. (10 marks)

   (ii) Identify the correct line number where the "out-of-sequence" Socket Primitive calls (and any other lines normally associated with that primitive call) should be placed e.g. "line 50, the call to connect, moves to line 60". (6 marks)

   (iii) Identify the type of application this is: client or server. Justify your answer. (4 marks)

```
1       Listen(connfd, LISTENQ);
2
3       listenfd = Socket(AF_INET, SOCK_STREAM, 0);
4
5       ..Complete server address structure. No Errors here..
6
7
8       for ( ; ; )
9       {
10
11        Bind(listenfd, (SA *) &inbuff, sizeof(servaddr));
12
13        listenfd = Accept(connfd, (SA *) NULL, NULL);
14
15        Write(connfd, inbuff, strlen(inbuff));
16
17        while ( (n = read(listenfd, inbuff, MAXLINE)) > 0)
18        {
19            if (strstr(inbuff,"\r\n")>0)
20                  break;
21        }
22
23        sscanf(inbuff, "%s %s %s", cmd, path, vers);
24
25        if (!strcmp(cmd, "/index.html"))
26        {
27            snprintf(outbuff, sizeof(outbuff), HOME_PAGE);
28        }
29        else
30        {
31            snprintf(outbuff, sizeof(outbuff), ERROR_PAGE);
32        }
33
34        Close(listenfd);
35      }//end for loop
36
37  }//end main
```

Figure 1 – *A snippet of code.*

2. The organisers of the *Web Summit* event have asked you to develop a simple client-server application that stores personal details about attendees at the event. The information to be stored for each attendee is: Title, Firstname, Surname, Email Address and Company. The application should return a simple verification message that their information has been stored. Error checking of empty fields is not required. All information supplied by the attendee is simply stored in a local file on the server.

   You are required to:

   (i) Design a suitable *syntax* (using a *box digram* format) for the protocol you intend implementing for data sent from the client application to the server application and vice versa (i.e. in the reverse direction). In your diagram outline any characters or character strings that might usefully be embedded in the data that would allow the protocol to work properly in terms of reading and parsing the data. (10 marks)

   (ii) Write suitable pseudocode for the **server** application. Your code should commence at the point where the socket is created i.e. ignore variable declarations and initialisations.

   In your code identify the following:
   - The correct sequence of *socket* primitives used throughout. Precise arguments are not required to be identified nor is there any requirement for error checking,
   - Focus only on the high-level steps; detailed steps such as "read one character-at-a-time from the file" can be replaced with "read data from the file" etc.

   (10 marks)

Section B

3. The following diagram (see Figure 2 – exchange of TCP segments) shows an exchange of TCP segments sequence between a client and server application. Some of the associated states of the TCP connection have been hidden from view.

You are required to:

(i) Identify the hidden states at points U through to Y inclusive. (5 marks)

(ii) Identify which side, client or server, has performed the *active/passive* close. (5 marks)

(iii) Identify the point where the connection is in a "half-closed" state. (5 marks)

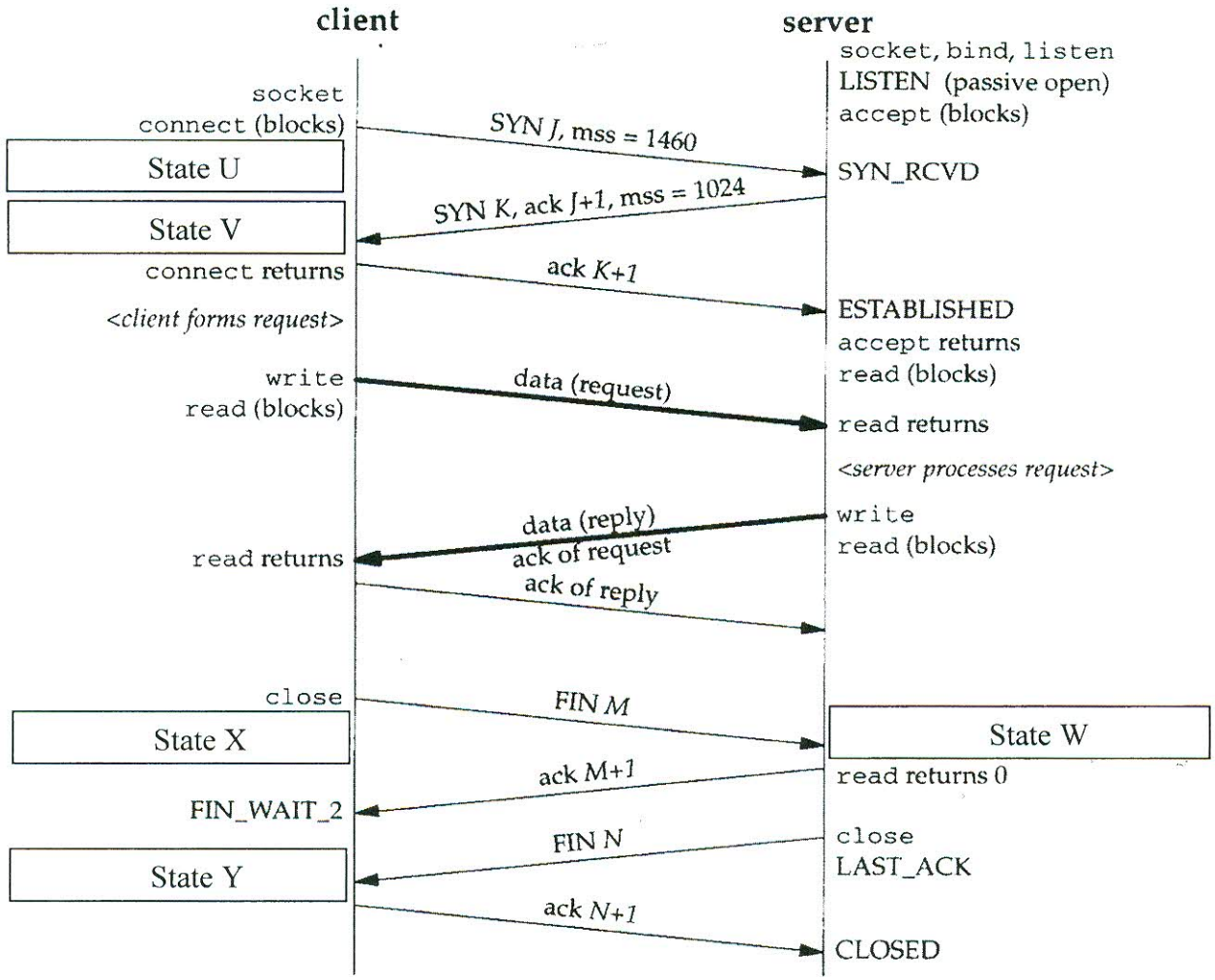

Figure 2 – *exchange of TCP segments*

4. In relation to the following printout of a *telnet* interaction with a mail server (see Figure 3: *telnet* interaction with SMTP server):

   (i) Identify the line numbers (either individual line numbers or ranges of line numbers) that relate to the three phases of communications. (5 marks)

   (ii) Identify the line numbers (either individual line numbers or ranges of line numbers) that show data destined for, and understood by the *User Agent* and those that show data destined for, and understood by the *Mail Transfer Agent*. Justify your answer. (10 marks)

1. Trying 147.252.25.102...
2. Connected to smtp.dit.ie.
3. 220 msgstore1.dit.ie -- Oracle Communications Messaging Server 7u4-27.01(7.0.4.27.0) 64bit
4. HELO student.dit.ie
5. 250 msgstore1.dit.ie OK, [147.252.234.34].
6. MAIL FROM: john.doe@dit.ie
7. 250 2.5.0 Address Ok.
8. RCPT TO: mary.jane@dit.ie
9. 250 2.1.5 mary.jane@dit.ie OK.
10. DATA
11. 354 Enter mail, end with a single ".".
12. From: john.doe@dit.ie
13. To: mary.jane@dit.ie
14. Subject: Dinner Tonight?
15. Fancy dinner tonight?
16. . (full stop)
17. 250 2.5.0 Ok, envelope id 0MVJ001@msgstore1.dit.ie
18. QUIT
19. 221 2.3.0 Bye received. Goodbye.
20. Connection closed.

Figure 3: *telnet* interaction with SMTP server.

5. One of the actions performed by the call to the *listen* socket primitive is to create a queue system inside the TCP entity for the related socket.

   (i) Describe the operation of this queue system. (5 marks)

   (ii) Explain how the TCP entity responds to incoming requests when the queue is full. (5 marks)

   (iii) Explain the other action performed by the call to *listen*. (5 marks)

6. In relation to a **client** component of a typical email application.

   (i) Illustrate the main architectural components (i.e. the MTA Client, MAA Client and User Agent) associated with this client application and explain the main role of each of the components. (7 marks)

   (ii) Identify the protocols (if any) used by these architectural components when communicating with their equivalent server applications (if any). (8 marks)

7. In relation to TCP's *Error Control* technique:

   (i) Explain how a sending TCP entity deals with "lost" data segments when sending data across a connection. (7 marks)

   (ii) The underlying *Network Architecture* and the traffic on the *Network* at any given moment in time necessitate TCP using a non-fixed (variable) *retransmission timer.* Explain? (8 marks)