ECE 316 - Operating Systems and Networking Laboratory

Practical Assignment 6

General Instructions

For each Assignment, a report (.pdf) and the source-code (e.g., .c, .cpp, .m, .bat etc.) of the solution should be submitted through Microsoft Teams "ECE316 – Operating Systems and Networks Laboratory" no later than the due date of the Assignment. The report should start with a cover page that clearly contains the assignment number and the Team number and the names of the members. In your report, include only the pseudocode, not the actual code, with any comments and description you may want to add, as well as a typical scenario that you used to test your programs. Please note that the report should be as concise as possible. Caution: You are not allowed to upload executables (.exe)!

If input test files are given, you are not allowed to make any changes to the provided input files.

Report File Naming Format: "Team#_Assignment#.pdf"

1. [40%] Σκοπός αυτής της άσκησης είναι να σας δώσει μια πρώτη επαφή με τον προγραμματισμό των sockets του Unix/Linux. Για την άσκηση αυτή θα πρέπει να υλοποιήσετε τις δύο διεργασίες Server και Client που επισυνάπτονται στο τέλος. Αφού τις υλοποιήσετε, εξηγήστε με συντομία τι κάνει η κάθε μία από τις πιο κάτω εντολές.
    a. socket ()
    b. bind ()
    c. listen ()
    d. accept ()
    e. connect ()
    f. serve()
    g. read ()
    h. write ()
    i. close ()

2. [60%] Σκοπός της άσκησης αυτής είναι να υλοποιήσετε το πρωτόκολλο παύσης και αναμονής (Stop and wait protocol) στην γλώσσα προγραμματισμού C χρησιμοποιώντας sockets. Θα χρειαστεί να υλοποιήσετε τις διεργασίες του αποστολέα (Transmitter) και παραλήπτη (Receiver) που θα επικοινωνούν μέσω ενός μη αξιόπιστου καναλιού (η διεργασία του καναλιού επισυνάπτεται στο τέλος). Συγκεκριμένα, ο αποστολέας θα διαβάζει τους χαρακτήρες του μηνύματος από το πληκτρολόγιο και θα τους αποστέλλει (ένα χαρακτήρα την κάθε φορά) στον παραλήπτη μέσω του καναλιού. Ο παραλήπτης θα τυπώνει τους σωστούς χαρακτήρες στην οθόνη (πρέπει να απορρίπτει τους διπλότυπους χαρακτήρες). **Τι θα καθιστούσε ένα κανάλι μη αξιόπιστο;**

Ο αποστολέας (Socket τύπου Client–non-blocking mode) θα συνδέεται με το κανάλι (channel) στο port 4610 (server port = 4610) και το κανάλι θα συνδέεται με τον παραλήπτη (Socket τύπου Server – non-blocking mode) στο port 4613 (Receiver port= 4613).

Ο αποστολέας θα διαβάζει μια σειρά χαρακτήρων από το πληκτρολόγιο και θα τους αποστέλλει στο κανάλι, ένα χαρακτήρα την κάθε φορά. Το πλαίσιο του αποστολέα θα έχει μήκος 2 χαρακτήρων. Ο πρώτος χαρακτήρας θα είναι το sequence number(0 ή 1) και ο δεύτερος θα αντιστοιχεί στο μήνυμα. Για παράδειγμα ένα μήνυμα για αποστολή μπορεί να έχει το ακόλουθο format pkt0={0,ch} or pkt1={1,ch} όπου ch ένας χαρακτήρας.

Το κανάλι μπορεί τυχαία, είτε να καθυστερήσει την αποστολή του frame είτε να το θεωρήσει χαμένο (δηλαδή δεν χρειάζεται η υλοποίηση CRC). Η μέγιστη καθυστέρηση που μπορεί να υπάρξει είναι 1.5 δευτερόλεπτα.

Να συγκρίνεται τα ακόλουθα τρία σενάρια.

1) Το κανάλι μπορεί να καθυστερήσει την αποστολή του frame ή να το θεωρήσει χαμένο.
2) Το κανάλι μπορεί μόνο να καθυστερήσει την αποστολή του frame
3) Το κανάλι μπορεί μόνο να θεωρήσει το frame χαμένο.

Ο παραλήπτης είναι υπεύθυνος να εκτυπώσει το σωστό μήνυμα που στάλθηκε από τον αποστολέα και να απαντήσει με το ανάλογο Acknowledgment (ACK).

SERVER:

```
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static void serve(int);

#define PORT 8080
#define BUFSIZE 1024

main( int argc, char **argv )
{
        printf("hello from server\n");
        int lsd; //listening socket
        struct sockaddr_in sin; //binding struct
        int sin_size=sizeof(sin);
        int sd; //socket to accept new connection

        //create listening socket
        printf("creating listening socket\n");
        lsd=socket(AF_INET, SOCK_STREAM, 0);
        if ( lsd == -1 )
        {
                fprintf(stderr, "%s: cannot create listening socket: ", argv[0]);
                perror(0);
                exit(1);
        }else{
                printf("%s: created listening socket\n", argv[0]);
        }

   bzero(&sin, sin_size);

   //assign IP, PORT
        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = htonl(INADDR_ANY);
        printf("%s\n", sin.sin_addr.s_addr);
        sin.sin_port = htons(PORT);
        printf("%d\n", sin.sin_port);

   // Binding newly created socket to given IP and verification
        if ( bind(lsd, &sin, sin_size) < 0 )
        {
                fprintf(stderr, "%s: cannot bind listening socket: ", argv[0]);
                perror(0);
                exit(1);
        }else{
```

```
                printf("%s: listening socket binded\n", argv[0]);
        }

        //Initiate a listen queue
        if ( listen(lsd, 5) <0 )
        {
                fprintf(stderr, "%s: cannot listen on socket: ", argv[0]);
                perror(0);
                exit(1);
        }else{
                printf("%s: listening on socket\n", argv[0]);
        }


        if ( (sd=accept(lsd, &sin, &sin_size)) <0){
                fprintf(stderr, "%s: cannot accept connection: ", argv[0]);
                perror(0);
                exit(1);
    }else
       printf("server acccept the client...\nWaiting for client's message\n");

        // Function for chatting between client and server
    serve(sd);

    // After chatting close the socket
    close(sd);
}

void serve(int sd){
        char buff[BUFSIZE];
    int n;
    // infinite loop for chat
    for (;;) {
       bzero(buff, BUFSIZE);

       // read the message from client and copy it in buffer
       read(sd, buff, sizeof(buff));
       // print buffer which contains the client contents
       printf("From client: %s\t To client : ", buff);
       bzero(buff, BUFSIZE);
       n = 0;
       // copy server message in the buffer
       while ((buff[n++] = getchar()) != '\n')
          ;

       // and send that buffer to client
       write(sd, buff, sizeof(buff));

       // if msg contains "Exit" then server exit and chat ended.
       if (strncmp("exit", buff, 4) == 0) {
          printf("Server Exit...\n");
          break;
       }
    }
    return;
}
```

CLIENT:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define BUFSIZE 1024
#define SERVER_PORT 4610

void func(int sockfd)
{
    char buff[BUFSIZE];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
    return;
}

main( int argc, char **argv )
{
            printf("hello from client\n");
            int sd;                 /* Socket descriptor */
            struct sockaddr_in server;    /* Server to connect */
            struct hostent *server_host;  /* Host info */
            char buf[BUFSIZE];
            int nbytes;

            /* Create socket */
            sd=socket(AF_INET, SOCK_STREAM, 0);
            if ( sd == -1 ) {
                        fprintf(stderr, "%s: cannot create socket: ", argv[0]);perror(0);
                        exit(1);
            }else
        printf("Socket successfully created..\n");

    bzero(&server, sizeof(server));

            /* Set up struct sockaddr_in */
            server.sin_family = AF_INET;
            server.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
        server.sin_port   = htons(SERVER_PORT);

        /* Connect */
        if ( connect(sd, &server, sizeof(server)) < 0 ) {
                fprintf(stderr, "%s: cannot connect to server: ", argv[0]);
                perror(0);
                exit(1);
        }

         // function for chat
    func(sd);

    // close the socket
    close(sd);
}
```

CHANNEL:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define BUFSIZE 1024
#define SERVER_PORT 4610

void func(int sockfd)
{
    char buff[BUFSIZE];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Enter the string : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("From Server : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Client Exit...\n");
            break;
        }
    }
    return;
}

main( int argc, char **argv )
{
        printf("hello from client\n");
        int sd;                 /* Socket descriptor */
        struct sockaddr_in server;    /* Server to connect */
        struct hostent *server_host;  /* Host info */
        char buf[BUFSIZE];
```

```
        int nbytes;

        /* Create socket */
        sd=socket(AF_INET, SOCK_STREAM, 0);
        if ( sd == -1 ) {
                fprintf(stderr, "%s: cannot create socket: ", argv[0]);perror(0);
                exit(1);
        }else
    printf("Socket successfully created..\n");

  bzero(&server, sizeof(server));

        /* Set up struct sockaddr_in */
        server.sin_family = AF_INET;
        server.sin_addr.s_addr = inet_addr("127.0.0.1");
        server.sin_port   = htons(SERVER_PORT);

        /* Connect */
        if ( connect(sd, &server, sizeof(server)) < 0 ) {
                fprintf(stderr, "%s: cannot connect to server: ", argv[0]);
                perror(0);
                exit(1);
        }

         // function for chat
  func(sd);

  // close the socket
  close(sd);
}
```