Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelanarayanan V

S.No	Experiment Name	Date	Page No.	Marks
1.	**Basic Network Configuration Commands**	**10-01-2024**		
2.	**Client-Server Application Echo**	**17-01-2024**		
3.	**IP Address Validation and Simple application of ATM using TCP**	**24-01-2024**		
4.	**CRC code generator using socket programming**	**07-02-2024**		
5. a)	**Echo programming using UDP**	**21-02-2024**		
5. b)	**IP address validation using UDP**	**21-02-2024**		

S.No	Experiment Name	Date	Page No.	Marks
5. c)	**ATM simulation using UDP**	**21-02-2024**		
6.	**Stop and wait ARQ**	**28-02-2024**		

Experiment No. 6

Experiment Name: Stop and wait ARQ using TCP and UDP programmings

Date: 28-2-2024

Problem Statement

Design a simple stop and wait ARQ protocol in UDPand TCP protocols and execute in Linux.

Aim

To write a c program for stop and wait ARQ protocol (UDP protocol) and execute in Linux environment. Here the feedback is not obtained and message transfer is faster than TCP.

Algorithm

Sender:
Rule 1) Send one data packet at a time.
Rule 2) Send the next packet only after receiving acknowledgement for the previous.

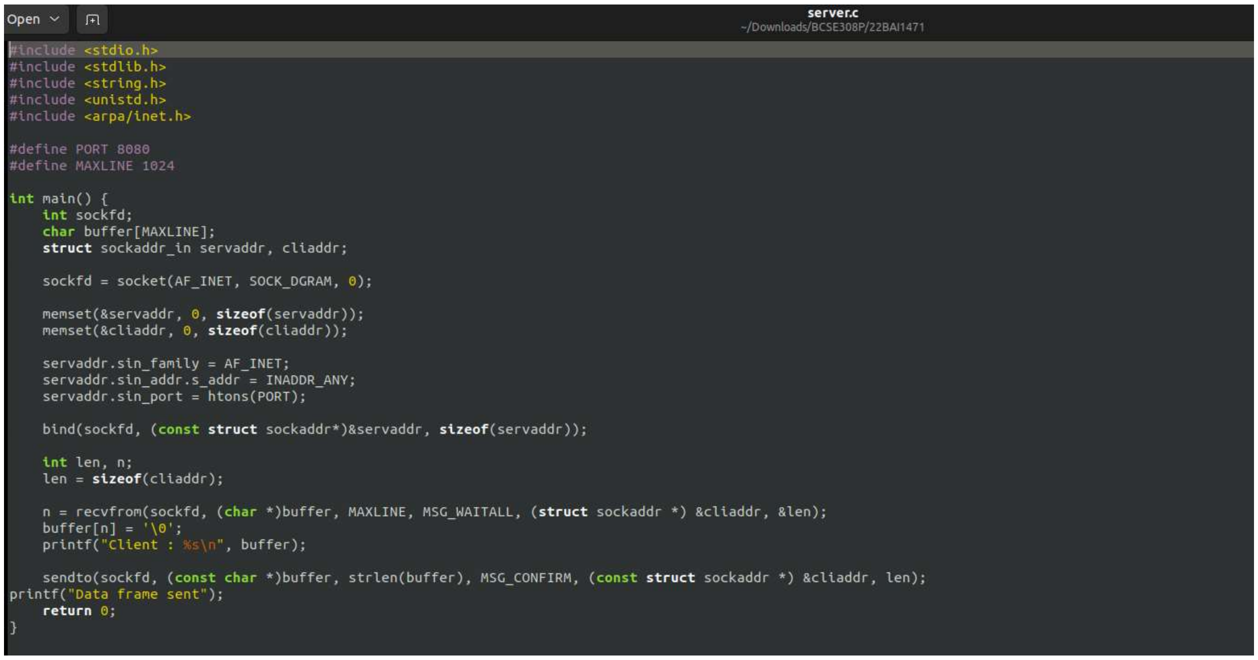
After sending printing the message data frame sent

Receiver:

Rule 1) Send acknowledgement after receiving and consuming a data packet.
Rule 2) After consuming packet acknowledgement need to be sent (Flow Control)

After receiving the frame printing the message acknowledgement number 0 or 1.. Received

Server side code

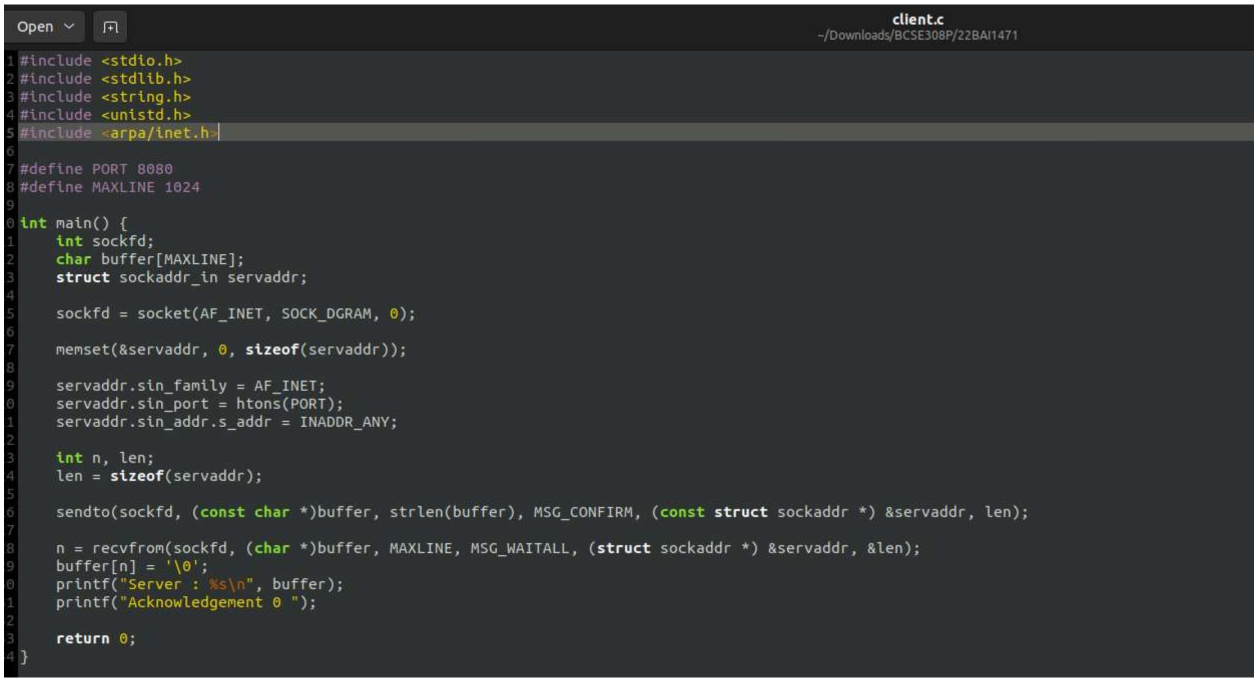


Open

server.c
~/Downloads/BCSE308P/22BAI1471

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    bind(sockfd, (const struct sockaddr*)&servaddr, sizeof(servaddr));

    int len, n;
    len = sizeof(cliaddr);

    n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *) &cliaddr, &len);
    buffer[n] = '\0';
    printf("Client : %s\n", buffer);

    sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct sockaddr *) &cliaddr, len);
printf("Data frame sent");
    return 0;
}
```

Client side code

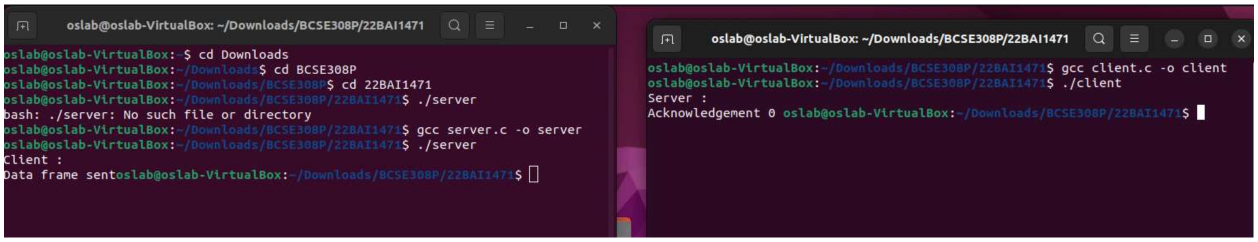


Open

client.c
~/Downloads/BCSE308P/22BAI1471

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&servaddr, 0, sizeof(servaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;
    len = sizeof(servaddr);

    sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct sockaddr *) &servaddr, len);

    n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *) &servaddr, &len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);
    printf("Acknowledgement 0 ");

    return 0;
}
```

Output on Linux terminal



```
oslab@oslab-VirtualBox:~$ cd Downloads
oslab@oslab-VirtualBox:~/Downloads$ cd BCSE308P
oslab@oslab-VirtualBox:~/Downloads/BCSE308P$ cd 22BAI1471
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./server
bash: ./server: No such file or directory
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./server
Client :
Data frame sentoslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$
```

```
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc client.c -o client
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./client
Server :
Acknowledgement 0 oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$
```

Aim

To write a c program for stop and wait ARQ protocol (TCP protocol) and execute in Linux environment.Here the feedback is obtained and message transfer is slower than UDP.

SERVER SIDE CODE

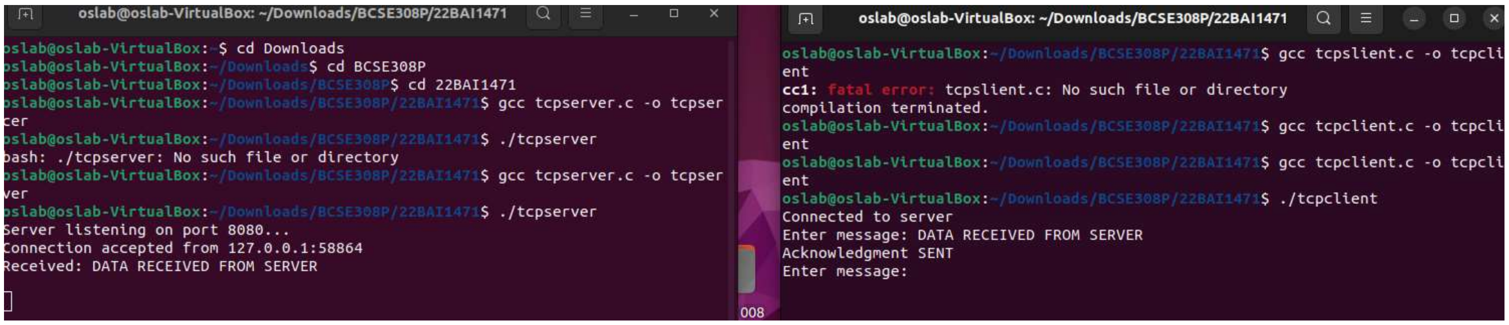
```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>


#define PORT 8080

#define BUFFER_SIZE 1024


void error(const char *msg) {

    perror(msg);

    exit(1);

}


int main() {

    int server_fd, client_fd, addr_len, recv_len;
    struct sockaddr_in server_addr, client_addr;

    char buffer[BUFFER_SIZE];

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {

        error("Socket creation failed");

    }

    memset(&server_addr, '0', sizeof(server_addr));

    server_addr.sin_family = AF_INET;

    server_addr.sin_addr.s_addr = INADDR_ANY;

    server_addr.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {

        error("Bind failed");

    }

    if (listen(server_fd, 3) < 0) {

        error("Listen failed");

    }


    printf("Server listening on port %d...\n", PORT);


    addr_len = sizeof(client_addr);


   if ((client_fd = accept(server_fd, (struct sockaddr *)&client_addr, (socklen_t*)&addr_len)) <
0) {

        error("Accept failed");

    }
    printf("Connection accepted from %s:%d\n", inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port));

    while (1) {

        recv_len = recv(client_fd, buffer, BUFFER_SIZE, 0);

        if (recv_len <= 0) {

            break;

        }

        buffer[recv_len] = '\0';

        printf("Received: %s\n", buffer);

        send(client_fd, "ACK", 3, 0);

    }


    close(server_fd);

    return 0;

}
```

CLIENT SIDE

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>


#define PORT 8080

#define SERVER_IP "127.0.0.1"

#define BUFFER_SIZE 1024
```

```
void error(const char *msg) {
    perror(msg);
    exit(1);
}

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        error("Socket creation failed");
    }
    memset(&server_addr, '0', sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
        error("Invalid address/ Address not supported");
    }
    if (connect(client_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        error("Connection failed");
    }

    printf("Connected to server\n");
    while (1) {
        printf("Enter message: ");
        fgets(buffer, BUFFER_SIZE, stdin);


        send(client_fd, buffer, strlen(buffer), 0);
        if (recv(client_fd, buffer, BUFFER_SIZE, 0) <= 0) {
            error("Acknowledgment not SENT");
        } else {
            printf("Acknowledgment SENT\n");
        }
    }


    close(client_fd);
    return 0;
}
```

OUTPUT


oslab@oslab-VirtualBox: ~/Downloads/BCSE308P/22BAI1471
oslab@oslab-VirtualBox: ~/Downloads/BCSE308P/22BAI1471

```
slab@oslab-VirtualBox:~$ cd Downloads
slab@oslab-VirtualBox:~/Downloads$ cd BCSE308P
slab@oslab-VirtualBox:~/Downloads/BCSE308P$ cd 22BAI1471
slab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc tcpserver.c -o tcpser
er
slab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./tcpserver
ash: ./tcpserver: No such file or directory
slab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc tcpserver.c -o tcpser
ver
slab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./tcpserver
erver listening on port 8080...
onnection accepted from 127.0.0.1:58864
eceived: DATA RECEIVED FROM SERVER
```

```
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc tcpslient.c -o tcpcli
ent
cc1: fatal error: tcpslient.c: No such file or directory
compilation terminated.
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc tcpclient.c -o tcpcli
ent
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ gcc tcpclient.c -o tcpcli
ent
oslab@oslab-VirtualBox:~/Downloads/BCSE308P/22BAI1471$ ./tcpclient
Connected to server
Enter message: DATA RECEIVED FROM SERVER
Acknowledgment SENT
Enter message:
```

Conclusion

Hence the stop and wait ARQ method is executed using both tcp and udp protocols.It is the simplest **flow control method** in which the sender will send the packet and then wait for the acknowledgement by the receiver that it has received the packet then it will send the next packet.

But if the no of data frames are more this process is not much reliable and takes a lot of time.