ARUN M
R6B 76

Networking lab assignment

I. a) Implement Stop-and-Wait ARQ flow control protocol.

server

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

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
   int server_fd, new_socket;
   struct sockaddr_in address;
   int addrlen = sizeof(address);
   char buffer[BUFFER_SIZE];
   int expected_seq = 0;

   server_fd = socket(AF_INET, SOCK_STREAM, 0);
   if (server_fd == -1) {
      perror("Socket failed");
      exit(EXIT_FAILURE);
   }

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

   if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
      perror("Bind failed");
      exit(EXIT_FAILURE);
   }

   if (listen(server_fd, 3) < 0) {
      perror("Listen failed");
```

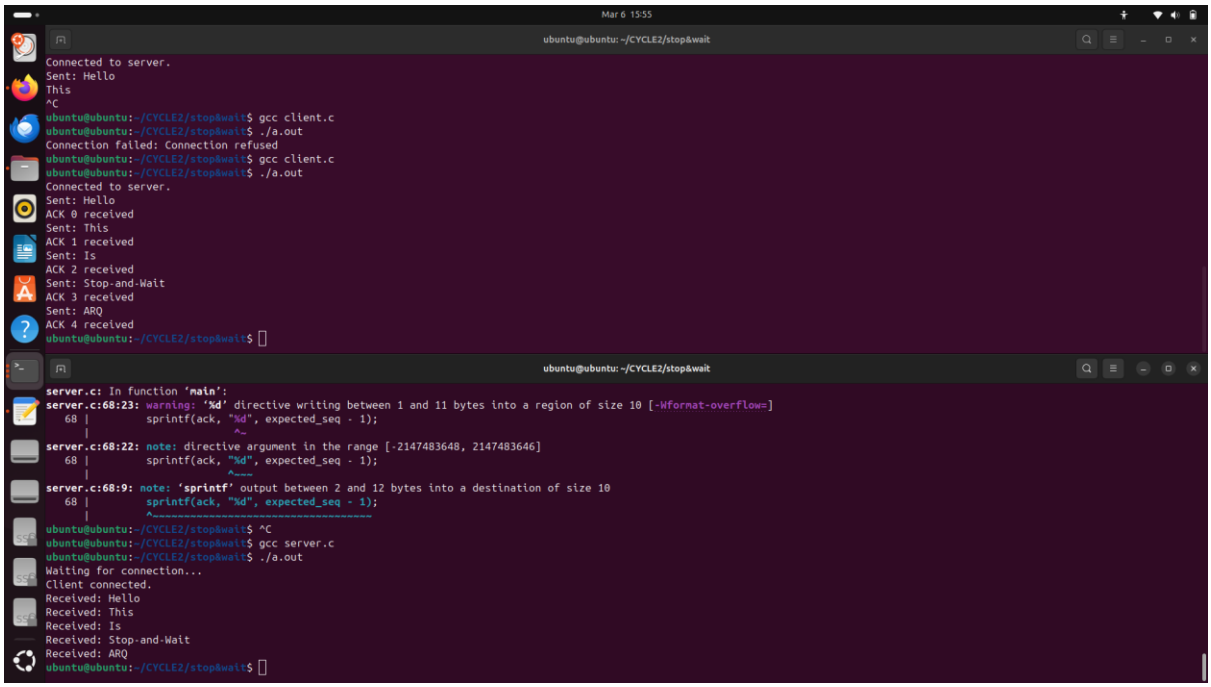
```
        exit(EXIT_FAILURE);
    }

    printf("Waiting for connection...\n");
    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen);
    if (new_socket < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }
    printf("Client connected.\n");

    while (1) {
        memset(buffer, 0, BUFFER_SIZE);
        int bytes_received = recv(new_socket, buffer, BUFFER_SIZE, 0);
        if (bytes_received <= 0) {
            break;
        }

        int seq_num;
        char message[BUFFER_SIZE];
        sscanf(buffer, "%d:%s", &seq_num, message);

        if (seq_num == expected_seq) {
            printf("Received: %s\n", message);
            expected_seq++;
        } else {
            printf("Duplicate packet detected, resending last ACK.\n");
        }

        char ack[12];
        sprintf(ack, "%d", expected_seq - 1);
        send(new_socket, ack, strlen(ack), 0);
    }

    close(new_socket);
    close(server_fd);
    return 0;
}
```

client

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

```
#define PORT 12345
#define SERVER_IP "127.0.0.1"
#define BUFFER_SIZE 1024

int main() {
    int sock;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    int seq_num = 0;
    char *messages[] = {"Hello", "This", "Is", "Stop-and-Wait", "ARQ"};
    int num_messages = sizeof(messages) / sizeof(messages[0]);


    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr);

    if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }
    printf("Connected to server.\n");

    for (int i = 0; i < num_messages; i++) {
        while (1) {
            sprintf(buffer, "%d:%s", seq_num, messages[i]);
            send(sock, buffer, strlen(buffer), 0);
            printf("Sent: %s\n", messages[i]);

            memset(buffer, 0, BUFFER_SIZE);
            recv(sock, buffer, BUFFER_SIZE, 0);

            int ack;
            sscanf(buffer, "%d", &ack);

            if (ack == seq_num) {
                printf("ACK %d received\n", ack);
                seq_num++;
                break;
            } else {
                printf("ACK mismatch, resending...\n");
```

```
            sleep(1);
        }
    }
}

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



b) Implement Go-Back--N ARQ flow control protocol.
server

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

#define PORT 12346
#define SERVER_IP "127.0.0.1"
#define BUFFER_SIZE 1024
#define WINDOW_SIZE 3

int main() {
    int sock;
```

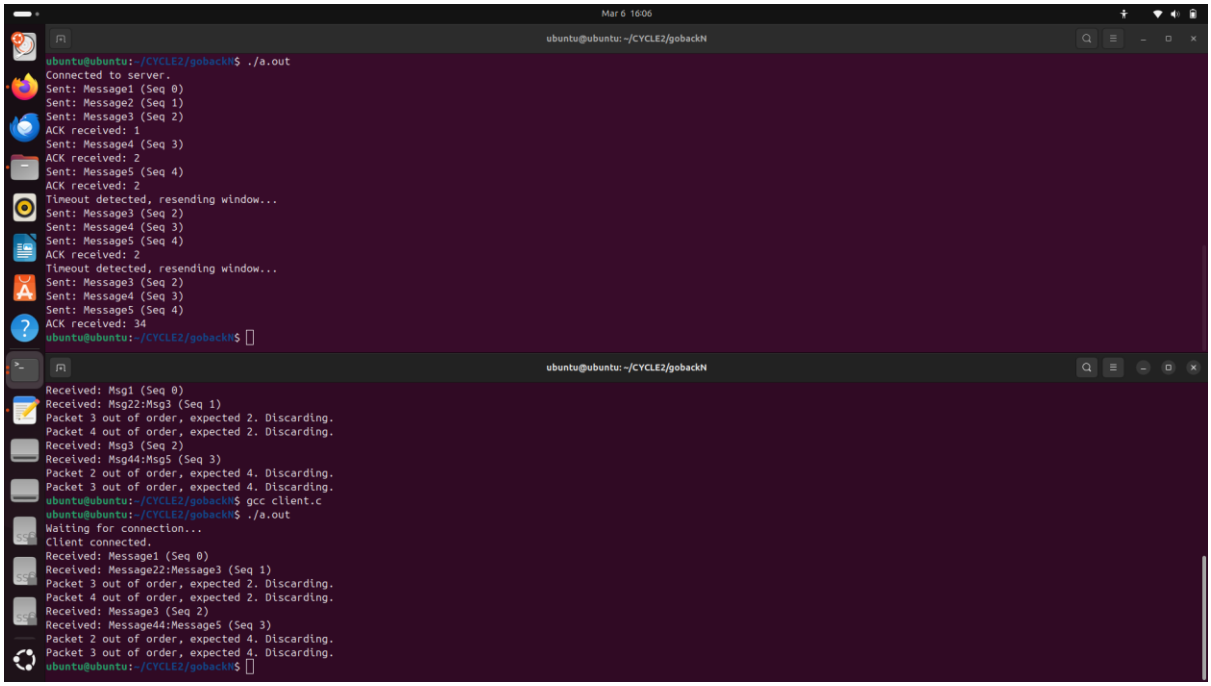
```
struct sockaddr_in server_addr;
char buffer[BUFFER_SIZE];
int base = 0, next_seq = 0;
char *messages[] = {"Message1", "Message2", "Message3", "Message4", "Message5"};
int num_messages = sizeof(messages) / sizeof(messages[0]);


sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1) {
   perror("Socket creation failed");
   exit(EXIT_FAILURE);
}


server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr);


if (connect(sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
   perror("Connection failed");
   exit(EXIT_FAILURE);
}
printf("Connected to server.\n");

while (base < num_messages) {
   // Send packets within the window
   while (next_seq < base + WINDOW_SIZE && next_seq < num_messages) {
      sprintf(buffer, "%d:%s", next_seq, messages[next_seq]);
      send(sock, buffer, strlen(buffer), 0);
      printf("Sent: %s (Seq %d)\n", messages[next_seq], next_seq);
      next_seq++;
   }

   // ereceive ACK
   memset(buffer, 0, BUFFER_SIZE);
   recv(sock, buffer, BUFFER_SIZE, 0);
   int ack;
   sscanf(buffer, "%d", &ack);
   printf("ACK received: %d\n", ack);

   if (ack > base) {
      base = ack;
   } else {
      printf("Timeout detected, resending window...\n");
      next_seq = base; // retransmit from base
   }
   sleep(1);
```

```
    }

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

client

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

#define PORT 12346
#define BUFFER_SIZE 1024

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE];
    int expected_seq = 0;


    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd == -1) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }


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

    // Bind the socket
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }


    if (listen(server_fd, 3) < 0) {
        perror("Listen failed");
        exit(EXIT_FAILURE);
```

```
    }

    printf("Waiting for connection...\n");
    new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen);
    if (new_socket < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }
    printf("Client connected.\n");

    while (1) {
        memset(buffer, 0, BUFFER_SIZE);
        int bytes_received = recv(new_socket, buffer, BUFFER_SIZE, 0);
        if (bytes_received <= 0) {
            break;
        }

        int seq_num;
        char message[BUFFER_SIZE];
        sscanf(buffer, "%d:%s", &seq_num, message);

        if (seq_num == expected_seq) {
            printf("Received: %s (Seq %d)\n", message, seq_num);
            expected_seq++;
        } else {
            printf("Packet %d out of order, expected %d. Discarding.\n", seq_num, expected_seq);
        }

        char ack[10];
        sprintf(ack, "%d", expected_seq);
        send(new_socket, ack, strlen(ack), 0);
    }

    close(new_socket);
    close(server_fd);
    return 0;
}
```


Mar 6 16:06
ubuntu@ubuntu: ~/CYCLE2/gobackN
ubuntu@ubuntu:~/CYCLE2/gobackN$ ./a.out
Connected to server.
Sent: Message1 (Seq 0)
Sent: Message2 (Seq 1)
Sent: Message3 (Seq 2)
ACK received: 1
Sent: Message4 (Seq 3)
ACK received: 2
Sent: Message5 (Seq 4)
ACK received: 2
Timeout detected, resending window...
Sent: Message3 (Seq 2)
Sent: Message4 (Seq 3)
Sent: Message5 (Seq 4)
ACK received: 2
Timeout detected, resending window...
Sent: Message3 (Seq 2)
Sent: Message4 (Seq 3)
Sent: Message5 (Seq 4)
ACK received: 34
ubuntu@ubuntu:~/CYCLE2/gobackN$
ubuntu@ubuntu: ~/CYCLE2/gobackN
Received: Msg1 (Seq 0)
Received: Msg22:Msg3 (Seq 1)
Packet 3 out of order, expected 2. Discarding.
Packet 4 out of order, expected 2. Discarding.
Received: Msg3 (Seq 2)
Received: Msg44:Msg5 (Seq 3)
Packet 2 out of order, expected 4. Discarding.
Packet 3 out of order, expected 4. Discarding.
ubuntu@ubuntu:~/CYCLE2/gobackN$ gcc client.c
ubuntu@ubuntu:~/CYCLE2/gobackN$ ./a.out
Waiting for connection...
Client connected.
Received: Message1 (Seq 0)
Received: Message22:Message3 (Seq 1)
Packet 3 out of order, expected 2. Discarding.
Packet 4 out of order, expected 2. Discarding.
Received: Message3 (Seq 2)
Received: Message44:Message5 (Seq 3)
Packet 2 out of order, expected 4. Discarding.
Packet 3 out of order, expected 4. Discarding.
ubuntu@ubuntu:~/CYCLE2/gobackN$