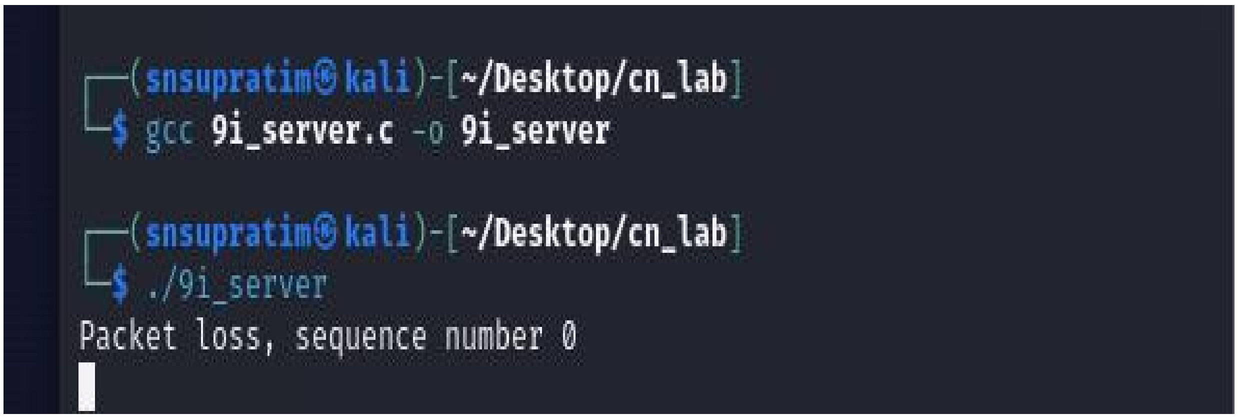
COMPUTER NETWORK LAB ASSIGNMENTS :
SUPRATIM NAG/CSE-AIM/22/57 :

Implementation of Data Link Layer Flow Control Mechanism:

Write a program to implement Sliding Window protocol.

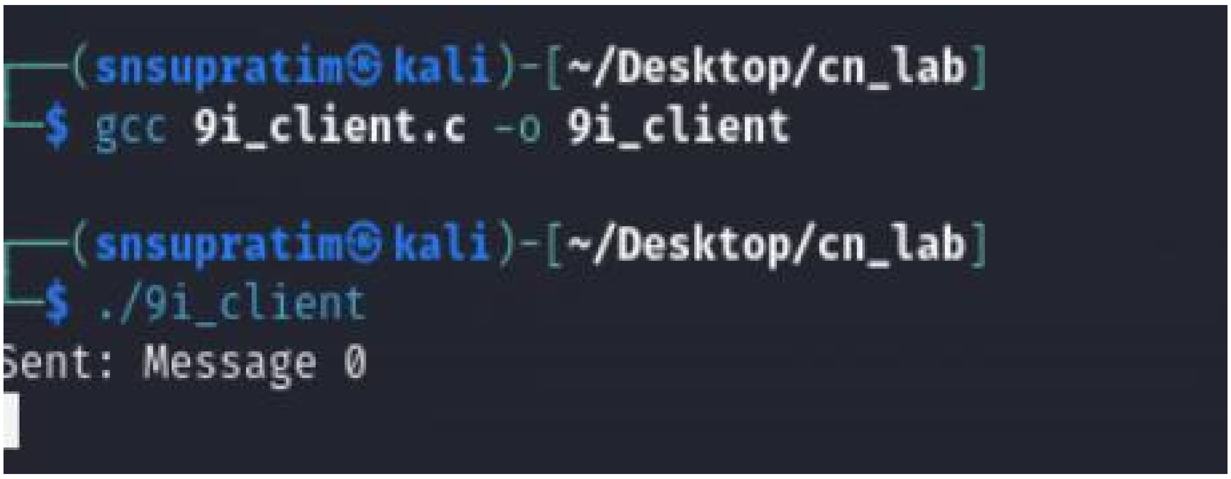
SERVER SIDE CODE :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define WINDOW_SIZE 4
#define BUFFER_SIZE 1024
#define PORT 8080
// Structure to hold packet information
struct packet {
int seq_num;
char data[BUFFER_SIZE];
};
int main() {
int sockfd;
struct sockaddr_in server_addr, client_addr;
socklen_t addr_len = sizeof(struct sockaddr_in);
char buffer[BUFFER_SIZE];
int expected_seq_num = 0;
struct packet ack_packet;
// Create UDP socket
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) <  0) {
perror("Socket creation failed");
exit(EXIT_FAILURE);
}
memset(&server_addr, 0, sizeof(server_addr));
memset(&client_addr, 0, sizeof(client_addr));
// Server information
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(PORT);
// Bind the socket with the server address
if (bind(sockfd, (const struct sockaddr *)&server_addr, sizeof(server_addr))
<  0) {
perror("Bind failed");
exit(EXIT_FAILURE);
}
while (1) {
// Receive data from client
struct packet recv_packet;
int len = recvfrom(sockfd, &recv_packet, sizeof(recv_packet), 0, (struct
sockaddr *)&client_addr, &addr_len);
if (len <  0) {
perror("Receive failed");
exit(EXIT_FAILURE);
}
// Simulate packet loss
if (rand() % 10 >= 3) {
printf("Packet loss, sequence number %d\n", recv_packet.seq_num);
continue;
}
// If packet is in expected sequence number
if (recv_packet.seq_num == expected_seq_num) {
printf("Received: %s\n", recv_packet.data);
// Send acknowledgment
ack_packet.seq_num = expected_seq_num;
sendto(sockfd, &ack_packet, sizeof(ack_packet), 0, (const struct sockaddr
*)&client_addr, addr_len);
// Move window
expected_seq_num++;
} else {
printf("Received out-of-order packet: %d\n", recv_packet.seq_num);
}
}
close(sockfd);
return 0;
}
```



```
┌──(snsupratim㉿kali)-[~/Desktop/cn_lab]
└─$ gcc 9i_server.c -o 9i_server

┌──(snsupratim㉿kali)-[~/Desktop/cn_lab]
└─$ ./9i_server
Packet loss, sequence number 0
```



```
──(snsupratim㉿kali)-[~/Desktop/cn_lab]
─$ gcc 9i_client.c -o 9i_client

──(snsupratim㉿kali)-[~/Desktop/cn_lab]
─$ ./9i_client
ent: Message 0
```

CLIENT SIDE CODE :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#define WINDOW_SIZE 4
#define BUFFER_SIZE 1024
#define PORT 8080
#define SERVER_IP "127.0.0.1"
// Structure to hold packet information
struct packet {
int seq_num;
char data[BUFFER_SIZE];
};
int main() {
int sockfd;
struct sockaddr_in server_addr;
socklen_t addr_len = sizeof(struct sockaddr_in);
char buffer[BUFFER_SIZE];
int seq_num = 0;
// Create UDP socket
if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
perror("Socket creation failed");
exit(EXIT_FAILURE);
}
memset(&server_addr, 0, sizeof(server_addr));
// Server information

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);
server_addr.sin_port = htons(PORT);
while (1) {
// Create packet
struct packet send_packet;
sprintf(send_packet.data, "Message %d", seq_num);
send_packet.seq_num = seq_num;
// Send packet
sendto(sockfd, &send_packet, sizeof(send_packet), 0, (const struct
sockaddr *)&server_addr, addr_len);
printf("Sent: %s\n", send_packet.data);
// Receive acknowledgment
struct packet ack_packet;
int len = recvfrom(sockfd, &ack_packet, sizeof(ack_packet), 0, NULL,
NULL);
if (len <  0) {
perror("Receive failed");
exit(EXIT_FAILURE);
}
// Check if acknowledgment matches sent sequence number
if (ack_packet.seq_num == seq_num) {
printf("Received acknowledgment for sequence number %d\n",
ack_packet.seq_num);
seq_num++;
} else {
printf("Received incorrect acknowledgment: %d\n", ack_packet.seq_num);
}
// Simulate delay
sleep(1);
}
close(sockfd);
return 0;
}
```