**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);

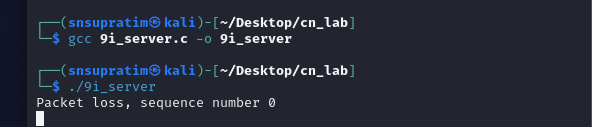
}

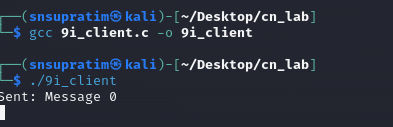
}

close(sockfd);

return 0;

}





#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;

}

CLIENT SIDE CODE :