**Server:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAX\_BUFFER\_SIZE 1024

// Function to perform bit stuffing

void performBitStuffing(char \*input, char \*output)

{

int count = 0;

for (int i = 0; input[i] != '\0'; i++)

{

output[count++] = input[i];

if (input[i] == '1')

{

output[count++] = '0';

}

}

output[count] = '\0';

}

int main()

{

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[MAX\_BUFFER\_SIZE] = {0};

char stuffedBuffer[MAX\_BUFFER\_SIZE \* 2] = {0};

// Create socket

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

{

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Prepare sockaddr\_in structure

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

// Bind socket

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0)

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_fd, 3) < 0)

{

perror("listen");

exit(EXIT\_FAILURE);

}

// Accept connection from client

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t \*)&addrlen)) < 0)

{

perror("accept");

exit(EXIT\_FAILURE);

}

// Read bit stream from client

valread = read(new\_socket, buffer, MAX\_BUFFER\_SIZE);

printf("Received bit stream from client: %s\n", buffer);

// Perform bit stuffing

performBitStuffing(buffer, stuffedBuffer);

// Send stuffed bit stream back to client

send(new\_socket, stuffedBuffer, strlen(stuffedBuffer), 0);

printf("Stuffed bit stream sent to client\n");

return 0;

}

**Client:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define MAX\_BUFFER\_SIZE 1024

int main()

{

int sock = 0, valread;

struct sockaddr\_in serv\_addr;

char buffer[MAX\_BUFFER\_SIZE] = {0};

char receivedBuffer[MAX\_BUFFER\_SIZE \* 2] = {0};

// Create socket

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0)

{

printf("\nInvalid address/ Address not supported \n");

return -1;

}

// Connect to server

if (connect(sock,(struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

{

printf("\nConnection Failed \n");

return -1;

}

// Input bit stream from user

printf("Enter bit stream: ");

scanf("%s", buffer);

// Send bit stream to server

send(sock, buffer, strlen(buffer), 0);

printf("Bit stream sent to server\n");

// Receive modified bit stream from server

valread = read(sock, receivedBuffer, MAX\_BUFFER\_SIZE \* 2);

printf("Modified bit stream received from server: %s\n", receivedBuffer);

return 0;

}







