**33 BIT STUFFING**

#include <stdio.h>

#include <string.h>

#define MAX 100

void bitStuffing(int input[], int n) {

int output[MAX], i, j = 0, count = 0;

// Start Flag

printf("Flag: 01111110\n");

for (i = 0; i < n; i++) {

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

// Check for five consecutive 1s

if (input[i] == 1) {

count++;

if (count == 5) {

output[j++] = 0; // Insert stuffed bit

count = 0; // Reset count after stuffing

}

} else {

count = 0; // Reset count when a 0 is found

}

}

// Print stuffed output

printf("Bit Stuffed Data: ");

for (i = 0; i < j; i++) {

printf("%d", output[i]);

}

printf("\n");

// End Flag

printf("Flag: 01111110\n");

}

int main() {

int input[MAX], n, i;

printf("Enter the number of bits: ");

scanf("%d", &n);

printf("Enter the bit sequence (0s and 1s only):\n");

for (i = 0; i < n; i++) {

scanf("%d", &input[i]);

}

bitStuffing(input, n);

return 0;

}

**UDP**

34

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int sockfd;

char buffer[BUFFER\_SIZE];

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t addr\_len = sizeof(client\_addr);

// Create UDP socket

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Define server address

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

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

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

// Bind socket to port

if (bind(sockfd, (const struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Bind failed");

exit(EXIT\_FAILURE);

}

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

// Receive message from client

recvfrom(sockfd, buffer, BUFFER\_SIZE, 0, (struct sockaddr \*)&client\_addr, &addr\_len);

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

// Send response to client

char \*message = "Hello from Server!";

sendto(sockfd, message, strlen(message), 0, (struct sockaddr \*)&client\_addr, addr\_len);

// Close socket

close(sockfd);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int sockfd;

char buffer[BUFFER\_SIZE];

struct sockaddr\_in server\_addr;

socklen\_t addr\_len = sizeof(server\_addr);

// Create UDP socket

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Define server address

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

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

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

printf("Sending message to Server...\n");

// Send message to server

char \*message = "Hello from Client!";

sendto(sockfd, message, strlen(message), 0, (struct sockaddr \*)&server\_addr, addr\_len);

// Receive response from server

recvfrom(sockfd, buffer, BUFFER\_SIZE, 0, (struct sockaddr \*)&server\_addr, &addr\_len);

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

// Close socket

close(sockfd);

return 0;

}

**TCP**

//Server Code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[BUFFER\_SIZE] = {0};

// Create socket

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

perror("Socket failed");

exit(EXIT\_FAILURE);

}

// Define server address (IPv4, Port)

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

// Bind socket to address

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

perror("Bind failed");

exit(EXIT\_FAILURE);

}

// Listen for connections (max queue: 3)

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

perror("Listen failed");

exit(EXIT\_FAILURE);

}

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

// Accept a connection

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

perror("Accept failed");

exit(EXIT\_FAILURE);

}

printf("Client connected!\n");

// Receive data from client

read(new\_socket, buffer, BUFFER\_SIZE);

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

// Send response to client

char \*message = "Hello from Server!";

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

// Close connection

close(new\_socket);

close(server\_fd);

return 0;

}

//Client Code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 8080

#define BUFFER\_SIZE 1024

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

char buffer[BUFFER\_SIZE] = {0};

// Create socket

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

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Define server address

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

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

// Convert address (127.0.0.1 for localhost)

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

perror("Invalid address");

exit(EXIT\_FAILURE);

}

// Connect to server

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

perror("Connection failed");

exit(EXIT\_FAILURE);

}

printf("Connected to Server!\n");

// Send message to server

char \*message = "Hello from Client!";

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

// Receive response from server

read(sock, buffer, BUFFER\_SIZE);

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

// Close socket

close(sock);

return 0;

}