Flow Control

Aim:

To simulate flow control technique using socket programming in C.

server:

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

#define PORT 8080
#define MAX_SEQ 4

typedef struct {
    char src;
    char dest;
    int seq_no;
    char data[4];
    char fcs; // Frame Check Sequence (dummy for simplicity)
} HDLC_Frame;

void send_ack(int sockfd, int ack_no) {
    write(sockfd, &ack_no, sizeof(ack_no));
    printf("Sent Acknowledgment: ACK%d\n", ack_no);
}

int main() {
    int sockfd, newsockfd, addr_len;
    struct sockaddr_in serv_addr, client_addr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = inet_addr("10.6.15.22"); // Server machine's IP
address


    if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Bind failed");
```

```
        exit(EXIT_FAILURE);
    }

    if (listen(sockfd, 5) < 0) {
        perror("Listen failed");
        exit(EXIT_FAILURE);
    }

    printf("Server is listening on port %d\n", PORT);
    addr_len = sizeof(client_addr);
    newsockfd = accept(sockfd, (struct sockaddr *)&client_addr, (socklen_t
*)&addr_len);
    if (newsockfd < 0) {
        perror("Accept failed");
        exit(EXIT_FAILURE);
    }

    int expected_seq_no = 0;
    HDLC_Frame frame;
    int ack_no;

    while (1) {
        int n = read(newsockfd, &frame, sizeof(frame));
        if (n <= 0) {
            break;
        }

        printf("Received: F-%c-%d-%s-FCS-F\n", frame.dest, frame.seq_no, frame.data);

        if (frame.seq_no == expected_seq_no) {
            printf("Packet %d received correctly.\n", frame.seq_no);
            ack_no = expected_seq_no + 1;
            expected_seq_no = (expected_seq_no + 1) % MAX_SEQ;
        } else {
            printf("Packet %d out of order. Expecting %d.\n", frame.seq_no,
expected_seq_no);
            ack_no = expected_seq_no; // Go-Back-N behavior
        }

        send_ack(newsockfd, ack_no);
    }

    printf("Connection closed.\n");
    close(newsockfd);
    close(sockfd);
```

```
    return 0;
}
```

client:

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

#define PORT 8080
#define MAX_SEQ 4 // Maximum sequence number for Go-Back-N

typedef struct {
    char src;
    char dest;
    int seq_no;
    char data[4];
    char fcs; // Frame Check Sequence (dummy for simplicity)
} HDLC_Frame;

void send_packet(int sockfd, HDLC_Frame frame) {
    write(sockfd, &frame, sizeof(frame));
    printf("Sent Packet: F-%c-%d-%s-FCS-F\n", frame.dest, frame.seq_no, frame.data);
}

int main() {
    int sockfd;
    struct sockaddr_in serv_addr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);
    serv_addr.sin_addr.s_addr = inet_addr("10.6.15.22"); // Server machine's IP


    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }
```

```
    char data[100];
    printf("Enter data to send: ");
    scanf("%s", data);

    char source = 'A';
    char destination = 'B';
    int len = strlen(data);
    int num_packets = (len + 3) / 4; // 4 bits per packet
    int base = 0, next_seq_no = 0, ack_no;

    while (base < num_packets) {
        HDLC_Frame frame;
        frame.src = source;
        frame.dest = destination;
        frame.seq_no = next_seq_no;
        strncpy(frame.data, &data[base * 4], 4);
        frame.data[4] = '\0'; // Null-terminate

        char send_choice;
        printf("Send Packet %d? (Y/N): ", next_seq_no);
        scanf(" %c", &send_choice);
        if (send_choice == 'Y') {
            send_packet(sockfd, frame);
        }

        read(sockfd, &ack_no, sizeof(ack_no));
        printf("Received Acknowledgement: ACK%d\n", ack_no);

        if (ack_no >= base + 1) {
            base = ack_no;
        }
        next_seq_no = (next_seq_no + 1) % MAX_SEQ;
    }

    printf("Transmission complete. Closing connection.\n");
    close(sockfd);
    return 0;
}
```

Output:

(1) server:

```
UGB2@ssn-23:~/Downloads$ ./ser
Server is listening on port 8080
Received: F-B-0-0100-FCS-F
Packet 0 received correctly.
Sent Acknowledgment: ACK1
Received: F-B-1-1001-FCS-F
Packet 1 received correctly.
Sent Acknowledgment: ACK2
Connection closed.
```

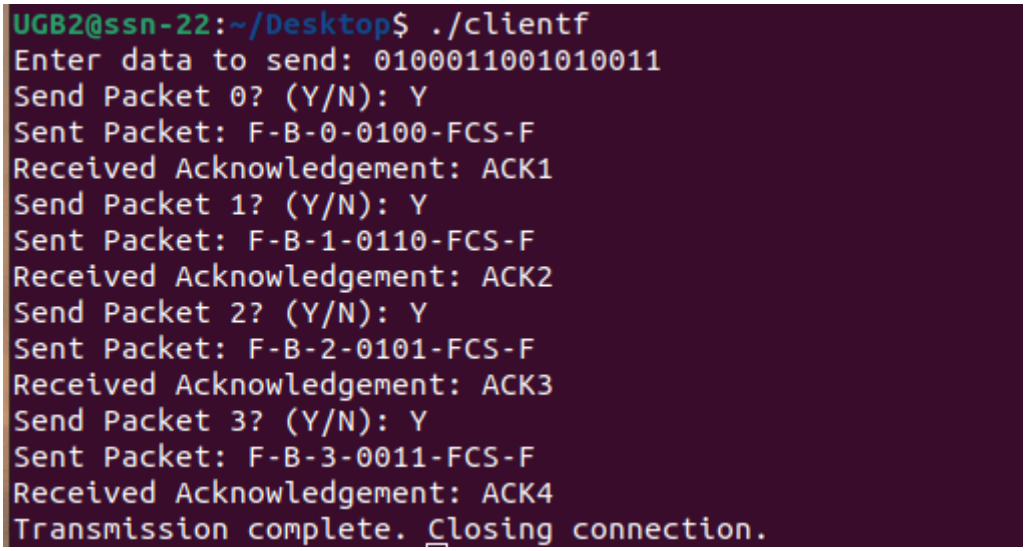
client:

```
UGB2@ssn-22:~/Desktop$ ./clientf
Enter data to send: 01001001
Send Packet 0? (Y/N): Y
Sent Packet: F-B-0-0100-FCS-F
Received Acknowledgement: ACK1
Send Packet 1? (Y/N): Y
Sent Packet: F-B-1-1001-FCS-F
Received Acknowledgement: ACK2
Transmission complete. Closing connection.
```

(2) server:

```
UGB2@ssn-23:~/Downloads$ ./ser
Server is listening on port 8080
Received: F-B-0-0100-FCS-F
Packet 0 received correctly.
Sent Acknowledgment: ACK1
Received: F-B-1-0110-FCS-F
Packet 1 received correctly.
Sent Acknowledgment: ACK2
Received: F-B-2-0101-FCS-F
Packet 2 received correctly.
Sent Acknowledgment: ACK3
Received: F-B-3-0011-FCS-F
Packet 3 received correctly.
Sent Acknowledgment: ACK4
Connection closed.
UGB2@ssn-23:~/Downloads$
```

client:


UGB2@ssn-22:~/Desktop$ ./clientf
Enter data to send: 0100011001010011
Send Packet 0? (Y/N): Y
Sent Packet: F-B-0-0100-FCS-F
Received Acknowledgement: ACK1
Send Packet 1? (Y/N): Y
Sent Packet: F-B-1-0110-FCS-F
Received Acknowledgement: ACK2
Send Packet 2? (Y/N): Y
Sent Packet: F-B-2-0101-FCS-F
Received Acknowledgement: ACK3
Send Packet 3? (Y/N): Y
Sent Packet: F-B-3-0011-FCS-F
Received Acknowledgement: ACK4
Transmission complete. Closing connection.