

```

    }

    if(listen(server_fd, 5) < 0) {
        printf("Listening failed!\n");
        exit(1);
    }

    if((client_fd = accept(server_fd, (struct sockaddr*) &address, (socklen_t*) &addrlen)) <
0) {
        printf("Connection failed!\n");
        exit(1);
    } else {
        printf("Connected to client.\n");
    }

    packet p;
    int flag = -1;

    while(1) {
        int status = recv(client_fd, &p, sizeof(packet), 0);

        if(status < 0) {
            printf("Receive failed!\n");
        } else if (status == 0) {
            printf("Receive completed.\nArray: ");

            for(int i = 0; arr[i] != -1; i++) {
                printf("%d ", arr[i]);
            }

            printf("\n");

            break;
        } else {
            if(flag != p.seq) {
                arr[k] = p.data;
                k++;
            }

            printf("Received: %d (SEQ %d)\n", p.data, p.seq);
            flag = p.seq;

            p.type = 1;
            p.seq = (p.seq + 1) % 2;

            if(rand() % 5 != 2) {

```

```

        if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
            printf("Send failed!\n");
        } else {
            printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);

            if(recv(d.client_fd, d.p, sizeof(packet), 0) < 0) {
                printf("Receive failed!\n");
            } else {
                printf("Received: ACK %d\n", d.p->seq);

                d.arr[*d.i] = -1;

                *(d.i) = *(d.i) + 1;
            }
        }
    } else {
        printf("SEQ %d lost\n", d.p->seq);
    }
}

void* timeout(void* t) {
    sleep(1);
    pthread_t tid = *((pthread_t*) t);
    pthread_cancel(tid);
}

void main() {
    int client_fd;
    struct sockaddr_in serv_addr;

    printf("TCP Client\n");

    client_fd = socket(AF_INET, SOCK_STREAM, 0);

    if(client_fd < 0) {
        printf("Socket creation failed!\n");
        exit(1);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(PORT);

    if(connect(client_fd, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) < 0) {
        printf("Connection failed!\n");
        exit(1);
    }
}

```

OUTPUT:

```
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1a$ gcc server.c
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1a$ ./a.out
Stop and Wait ARQ
TCP Server
Connected to client.
Received: 1 (SEQ 0)
Sent: ACK 1
Received: 2 (SEQ 1)
Sent: ACK 0
Received: 3 (SEQ 0)
ACK 1 lost
Received: 3 (SEQ 0)
Sent: ACK 1
Received: 4 (SEQ 1)
Sent: ACK 0
Received: 5 (SEQ 0)
Sent: ACK 1
Receive completed.
Array: 1 2 3 4 5
```

```
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1a$ gcc client.c -lpthread
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1a$ ./a.out 9030
TCP Client
Connected to server.
Enter array size: 5
Enter array elements: 1 2 3 4 5
Sent: 1 (SEQ 0)
Received: ACK 1
Sent: 2 (SEQ 1)
Received: ACK 0
SEQ 0 lost
Sent: 3 (SEQ 0)
Sent: 3 (SEQ 0)
Received: ACK 1
Sent: 4 (SEQ 1)
Received: ACK 0
Sent: 5 (SEQ 0)
Received: ACK 1
Send completed.
Array: -1 -1 -1 -1 -1
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1a$
```

b) GO-BACK-N ARQ

SOURCE CODE:

SERVER:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
```

```

        exit(1);
    } else {
        printf("Connected to client.\n");
    }

    packet p;
    int exp_seq = 0, flag = 0;

    while(1) {
        int status = recv(client_fd, &p, sizeof(packet), 0);

        if(status < 0) {
            printf("Receive failed!\n");
        } else if (status == 0) {
            printf("Receive completed.\nArray: ");

            for(int i = 0; arr[i] != -1; i++) {
                printf("%d ", arr[i]);
            }

            printf("\n");

            break;
        } else {
            if(p.seq > exp_seq) {
                if(!flag) {
                    flag = 1;

                    p.type = -1;

                    p.seq = exp_seq;

                    if(send(client_fd, &p, sizeof(packet), 0) < 0) {
                        printf("Send failed!\n");
                    } else {
                        printf("Sent: NACK %d\n", p.seq);
                    }
                }

                continue;
            } else {
                flag = 0;

                exp_seq = p.seq + 1;
            }
        }
    }

```

```

    int n;
    int client_fd;
    int exp_seq;
    packet* p;
    window* w;
} data;

void recvAck(data d);

void sendWindow(data d) {
    d.p->seq = d.w->start;

    for(int i = d.w->start; i <= d.w->end && i < d.n; i++) {
        d.p->type = 0;
        d.p->data = d.arr[i];

        if(rand() % 10 != 6) {
            if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
                printf("Send failed!\n");
            } else {
                printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);
            }
        } else {
            printf("%d (SEQ %d) lost\n", d.p->data, d.p->seq);
        }

        d.p->seq = d.p->seq + 1;
    }

    recvAck(d);
}

void sendFrame(data d) {
    d.p->type = 0;
    d.p->data = d.arr[d.w->end];

    if(rand() % 10 != 6) {
        if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
            printf("Send failed!\n");
        } else {
            printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);
        }
    } else {
        printf("%d (SEQ %d) lost\n", d.p->data, d.p->seq);
    }
}

```

```

        d.exp_seq = d1.p->seq + 1;

        if(d.w->end < d.n)
            sendFrame(d);
        else
            recvAck(d);
    } else if(d1.p->type == -1) {
        printf("Received: NACK %d. Window retransmitting.\n", d1.p->seq);
        sendWindow(d);
    }
}

}

void main() {
    int client_fd;
    struct sockaddr_in serv_addr;

    printf("TCP Client\n");

    client_fd = socket(AF_INET, SOCK_STREAM, 0);

    if(client_fd < 0) {
        printf("Socket creation failed!\n");
        exit(1);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(PORT);

    if(connect(client_fd, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) < 0) {
        printf("Connection failed!\n");
        exit(1);
    } else {
        printf("Connected to server.\n");
    }

    struct timeval tv;
    tv.tv_sec = 1;
    tv.tv_usec = 0;
    setsockopt(client_fd, SOL_SOCKET, SO_RCVTIMEO, (const char*)&tv, sizeof(tv);

    int n;
    window w;

    printf("Enter window size: ");

```

OUTPUT:

```
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1b$ ./a.out
TCP Client
Connected to server.
Enter window size: 3
Enter array size:
6
Enter array elements: 1 2 3 4 5 6
Sent: 1 (SEQ 0)
2 (SEQ 1) lost
Sent: 3 (SEQ 2)
Received: ACK 0
Sent: 4 (SEQ 3)
Received: NACK 1. Window retransmitting.
Sent: 2 (SEQ 1)
Sent: 3 (SEQ 2)
4 (SEQ 3) lost
ACK 1 not received! Window retransmitting.
Sent: 2 (SEQ 1)
Sent: 3 (SEQ 2)
Sent: 4 (SEQ 3)
Received: ACK 1
Sent: 5 (SEQ 4)
Received: ACK 2
Sent: 6 (SEQ 5)
Received: ACK 3
ACK 4 not received! Window retransmitting.
Sent: 5 (SEQ 4)
Sent: 6 (SEQ 5)
Received: ACK 4
Received: ACK 5
Send completed.
Array: -1 -1 -1 -1 -1 -1
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1b$
```

```

        int seq; // Sequence number
    } packet;

int add(int* arr, int key, int index) {
    int flag = -1;

    for(int i = 0; i < index; i++) {
        if(arr[i] == -1) {
            flag = i;
            break;
        }
    }

    arr[index] = key;

    return flag;
}

void main() {
    int server_fd, client_fd;
    struct sockaddr_in address;
    int addrlen = sizeof(address);

    printf("Selective Repeat ARQ\nTCP Server\n");

    if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("Socket creation failed!\n");
        exit(1);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if(bind(server_fd, (struct sockaddr*)&address, addrlen) < 0) {
        printf("Socket binding failed!\n");
        exit(1);
    }

    if(listen(server_fd, 5) < 0) {
        printf("Listening failed!\n");
        exit(1);
    }

    if((client_fd = accept(server_fd, (struct sockaddr*)&address, (socklen_t*)&addrlen)) <
0) {

```



```

        } else {
            printf("Lost: NACK %d\n", p.seq);
        }

        p.seq = temp;
    }

    p.type = 1;

    if(rand() % 10 != 6) {
        if(send(client_fd, &p, sizeof(packet), 0) < 0) {
            printf("Send failed!\n");
        } else {
            printf("Sent: ACK %d\n", p.seq);
        }
    } else {
        printf("Lost: ACK %d\n", p.seq);
    }
}

close(server_fd);
close(client_fd);
}

```

CLIENT:

```

#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

#define PORT 9000

int count = 0;

typedef struct packet {
    int data;
    int type; // SEQ (0), ACK (1) or NACK(-1)
    int seq; // Sequence number
} packet;

typedef struct window {
    int size;
    int start;

```

```

void sendFrame(data d, int seq) {
    d.p->type = 0;
    int temp;

    if(seq == -1)
        d.p->data = d.arr[d.w->end];
    else {
        d.p->data = d.arr[seq];
        temp = d.p->seq;
        d.p->seq = seq;
    }

    if(d.p->data == -1)
        return;

    if(rand() % 10 != 6) {
        if(send(d.client_fd, d.p, sizeof(packet), 0) < 0) {
            printf("Send failed!\n");
        } else {
            printf("Sent: %d (SEQ %d)\n", d.p->data, d.p->seq);
        }
    } else {
        printf("Lost: %d (SEQ %d)\n", d.p->data, d.p->seq);
    }

    if(seq == -1)
        d.p->seq = d.p->seq + 1;
    else
        d.p->seq = temp;
}

void recvAck(data d) {
    data d1;
    packet p;
    d1.p = &p;

    if(recv(d.client_fd, d1.p, sizeof(packet), 0) < 0) {
        printf("Time out! Window retransmitting.\n");
        sendWindow(d);
        recvAck(d);
    } else {
        if(d1.p->type == 1) {
            if(d.arr[d1.p->seq] == -1) {
                recvAck(d);
            } else {
                printf("Received: ACK %d\n", d1.p->seq);
            }
        }
    }
}

```

```

void main() {
    int client_fd;
    struct sockaddr_in serv_addr;

    printf("TCP Client\n");

    client_fd = socket(AF_INET, SOCK_STREAM, 0);

    if(client_fd < 0) {
        printf("Socket creation failed!\n");
        exit(1);
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(PORT);

    if(connect(client_fd, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) < 0) {
        printf("Connection failed!\n");
        exit(1);
    } else {
        printf("Connected to server.\n");
    }

    struct timeval tv;
    tv.tv_sec = 1;
    tv.tv_usec = 0;
    setsockopt(client_fd, SOL_SOCKET, SO_RCVTIMEO, (const char*)&tv, sizeof tv);

    int n;
    window w;

    printf("Enter window size: ");
    scanf("%d", &w.size);

    w.start = 0;
    w.end = w.size - 1;

    printf("Enter array size: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter array elements: ");
    for(int i = 0; i < n; i++) {

```

```
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1c$ gcc client.c
dell@dell-Latitude-5480:~/Programs/NetworkLab/cycle2/exp1c$ ./a.out
TCP Client
Connected to server.
Enter window size: 3
Enter array size: 5
Enter array elements: 10 20 30 40 50
Sent: 10 (SEQ 0)
Lost: 20 (SEQ 1)
Sent: 30 (SEQ 2)
Received: ACK 0
Sent: 40 (SEQ 3)
Received: ACK 2
ACK 1 not received! Frame 1 retransmitting.
Sent: 20 (SEQ 1)
Sent: 50 (SEQ 4)
Received: NACK 1. Frame 1 retransmitting.
Lost: 20 (SEQ 1)
Received: ACK 3
ACK 1 not received! Frame 1 retransmitting.
Sent: 20 (SEQ 1)
Received: ACK 1
Time out! Window retransmitting.
Sent: 50 (SEQ 4)
Received: ACK 4
Send completed.
Array: -1 -1 -1 -1 -1
```