**HW 4. Retransmission 알고리즘 구현**컴퓨터네트워크 02분반

컴퓨터공학과

2142851 김형준

**목차**

1. **Go-Back-N**
   1. 코드
   2. 결과 출력
   3. 어려웠던 점
2. **Selective repeat**
   1. 코드
   2. 결과 출력
   3. 어려웠던 점
3. **Go-Back-N**
4. **코드**

Sender.c

*// 2142851 컴퓨터공학과 김형준*

*// Go-Back-N Sender*

*#include* "../Common.h"

*#include* <pthread.h>

*#define* SERVERPORT 9000

*#define* BUFSIZE    512

char ACKs[6][10] = { "ACK 0", "ACK 1", "ACK 2", "ACK 3", "ACK 4", "ACK 5" };

char packets[6][10] = { "packet 0", "packet 1", "packet 2", "packet 3", "packet 4", "packet 5" };

*// 가장 최근 받은 ACK*

char lastACK[10];

*// 사용하진 않음. => 잘 받았는지 확인*

bool isReceived[6] = { false, false, false, false, false, false };

int arrow = 0;

int send\_arrow = 0;

int send\_area = 4;

clock\_t startTime[6], endTime[6];

double elapsedTime[6];

*// 서버로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

        retval = recv(sock, buf, BUFSIZE, 0);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        }

        buf[retval] = '\0';

*if* (strcmp(buf, ACKs[arrow]) == 0) {

            printf("\"%s\" is received.\n", buf);

            isReceived[arrow] = true;

            arrow++;

            send\_area++;

        }

*else* {

            printf("\"%s\" is received and ignored.\n", buf);

        }

        strcpy(lastACK, buf);

    }

    close(sock);

    pthread\_exit(NULL);

}

*// 서버로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*/\**

*if (fgets(buf, BUFSIZE + 1, stdin) == NULL)*

*break;*

*int len = (int)strlen(buf);*

*if (buf[len - 1] == '\n')*

*buf[len - 1] = '\0';*

*if (strlen(buf) == 0)*

*break;*

*\*/*

*if* (send\_area > 0 && send\_arrow < 6) {

            strcpy(buf, packets[send\_arrow]);

            retval = send(sock, buf, (int)strlen(buf), 0);

*if* (retval == SOCKET\_ERROR) {

                err\_display("send()");

*break*;

            }

            printf("\"%s\" is transmitted.\n", buf);

*// 0.1 second delay*

            usleep(100000);

            startTime[send\_arrow] = clock();

            send\_arrow++;

            send\_area--;

        }

*if* ((strcmp(buf, packets[5]) == 0) && (strcmp(lastACK, ACKs[5]) == 0)) {

*/\**

*printf("===== END =====\n");*

*printf("buf : %s\n", buf);*

*printf("packets[5] : %s\n", packets[5]);*

*printf("lastACK : %s\n", lastACK);*

*printf("ACKs[5] : %s\n", ACKs[5]);*

*printf("strcmp(buf, packets[5]) : %d\n", strcmp(buf, packets[5]));*

*printf("strcmp(lastACK, ACKs[5]) : %d\n", strcmp(lastACK, ACKs[5]));*

*printf("===== END =====\n");*

*\*/*

*break*;

        }

*for* (int i = arrow; i < send\_arrow; i++) {

            endTime[i] = clock();

            elapsedTime[i] = (double)(endTime[i] - startTime[i]) / CLOCKS\_PER\_SEC;

        }

*// Time Out*

*for* (int i = 0; i < send\_arrow; i++) {

*// 전송 후 5초가 지나면 타임아웃*

*if* (elapsedTime[i] > 5) {

                printf("%s is timeout.\n", packets[i]);

*// 보낼 수 있는 공간 및 arrow 초기화*

                send\_area = 4;

                send\_arrow = i;

*break*;

            }

        }

    }

    close(sock);

    pthread\_exit(NULL);

}

int main(int *argc*, char \**argv*[]) {

    int retval;

*// 명령행 인수가 있으면 IP 주소로 사용*

    char \*SERVERIP = (argc > 1) ? argv[1] : (char \*)"127.0.0.1";

*// 소켓 생성*

    SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);

*if* (sock == INVALID\_SOCKET) err\_quit("socket()");

*// connect()*

    struct sockaddr\_in serveraddr;

    memset(&serveraddr, 0, sizeof(serveraddr));

    serveraddr.sin\_family = AF\_INET;

    inet\_pton(AF\_INET, SERVERIP, &serveraddr.sin\_addr);

    serveraddr.sin\_port = htons(SERVERPORT);

    retval = connect(sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

*if* (retval == SOCKET\_ERROR) err\_quit("connect()");

    pthread\_t r\_tid, s\_tid;

*// 서버로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 수신 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

*// 서버로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 전송 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(sock);

*return* 0;

}

Receiver.c

*// 2142851 컴퓨터공학과 김형준*

*// Go-Back-N Receiver*

*#include* "../Common.h"

*#include* <pthread.h>

*#define* BUFSIZE    512

*#define* SERVERPORT 9000

char ACKs[6][10] = { "ACK 0", "ACK 1", "ACK 2", "ACK 3", "ACK 4", "ACK 5" };

char packets[6][10] = { "packet 0", "packet 1", "packet 2", "packet 3", "packet 4", "packet 5" };

int receive\_arrow = 0;

bool isReceived = false;

bool firstPacket2Drop = true;

*// 클라이언트로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

        retval = recv(client\_sock, buf, BUFSIZE, 0);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        } *else* *if* (retval == 0) {

            printf("[disconnected]\n");

*break*;

        }

*if* (firstPacket2Drop && (strcmp(buf, packets[2]) == 0)) {

            firstPacket2Drop = false;

*continue*;

        }

*// 받은 데이터 출력*

        buf[retval] = '\0';

*// printf("[TCP 서버] %d바이트를 받았습니다.\n", retval);// 받은 데이터 출력*

*if* (strcmp(buf, packets[receive\_arrow]) == 0) { *// 같다면 (잘 왔다면)*

*// 0.1 second delay*

            usleep(100000);

            printf("\"%s\" is received. ", buf);

            receive\_arrow++;

        }

*else* { *// 다르다면*

*// 0.1 second delay*

            usleep(100000);

            printf("\"%s\" is received and dropped. ", buf);

        }

        isReceived = true;

*//printf("isReceived : %d\n\n", isReceived);*

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

*// 클라이언트로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// if (fgets(buf, BUFSIZE + 1, stdin) == NULL)*

*//     break;*

*// int len = (int)strlen(buf);*

*// if (buf[len - 1] == '\n')*

*//     buf[len - 1] = '\0';*

*// if (strlen(buf) == 0)*

*//     break;*

*if* (isReceived) {

*// 1 second delay*

*// usleep(1000000);*

            strcpy(buf, ACKs[receive\_arrow - 1]);

            retval = send(client\_sock, buf, (int)strlen(buf), 0);

*if* (retval == SOCKET\_ERROR) {

                err\_display("send()");

*break*;

            }

            printf("\"%s\" is transmitted. \n", buf);

            isReceived = false;

        }

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

int main(int *argc*, char \**argv*[]) {

*// 소켓 생성*

    SOCKET listen\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

*if* (listen\_sock == INVALID\_SOCKET) err\_quit("socket()");

    struct sockaddr\_in serveraddr;

    memset(&serveraddr, 0, sizeof(serveraddr));

    serveraddr.sin\_family = AF\_INET;

    serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    serveraddr.sin\_port = htons(SERVERPORT);

*if* (bind(listen\_sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr)) == SOCKET\_ERROR)

        err\_quit("bind()");

*if* (listen(listen\_sock, SOMAXCONN) == SOCKET\_ERROR)

        err\_quit("listen()");

    struct sockaddr\_in clientaddr;

    socklen\_t addrlen = sizeof(clientaddr);

    SOCKET client\_sock = accept(listen\_sock, (struct sockaddr \*)&clientaddr, &addrlen);

*if* (client\_sock == INVALID\_SOCKET) {

        err\_display("accept()");

    }

    pthread\_t r\_tid, s\_tid;

*// 클라이언트로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 수신 스레드 생성 실패\n");

        close(client\_sock);

    }

*// 클라이언트로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 전송 스레드 생성 실패\n");

        close(client\_sock);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(listen\_sock);

*return* 0;

}

1. **결과 출력**

Sender

텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

자동 생성된 설명

Receiver

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. **어려웠던 점**

처음에는 자동으로 보내는 것이 아닌, 직접 “packet n”을 입력하여 보내는 식으로 작성하였습니다. 이 부분에서 막히는 부분은 없었습니다. 하지만 입력없이 자동으로 보낼 때, timeout을 위한 usleep 함수를 사용하였는데, Receiver에서 받는 부분에 지연을 주었더니 “packet 0packet 1packet 2”와 같이 묶여서 나오는 부분을 고치는 것이 어려웠습니다.

또한, 자주 사용하지 않는 지연 함수를 사용하니 packet 각각의 RTT를 구하는 부분이 어려웠습니다. 구한다 하더라도, 시간을 체크하는 부분에서 막혔었습니다. 하지만 for문의 조건 및 초기값 설정을 잘 해주었더니, 성공적으로 코딩할 수 있었습니다.

최대로 보낼 수 있는 수를 4개로 설정하기 위한 send\_area 변수를 선언하는 부분이 살짝 헷갈렸습니다.

1. **Selective repeat**
2. **코드**

Sender

*// 2142851 컴퓨터공학과 김형준*

*// Selective repeat Sender*

*#include* "../Common.h"

*#include* <pthread.h>

*#include* <ctype.h>

*#define* SERVERPORT 9000

*#define* BUFSIZE    512

char ACKs[6][10] = { "ACK 0", "ACK 1", "ACK 2", "ACK 3", "ACK 4", "ACK 5" };

char packets[6][10] = { "packet 0", "packet 1", "packet 2", "packet 3", "packet 4", "packet 5" };

*// 가장 최근 받은 ACK*

char lastACK[10];

*// 잘 받았는지 확인*

bool isReceived[6] = { false, false, false, false, false, false };

int arrow = 0;

int send\_arrow = 0;

int send\_area = 4;

clock\_t startTime[6], endTime[6];

double elapsedTime[6];

*// 서버로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

        retval = recv(sock, buf, BUFSIZE, 0);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        }

        buf[retval] = '\0';

*if* (strcmp(buf, ACKs[arrow]) == 0) {

            printf("\"%s\" is received.\n", buf);

            isReceived[arrow] = true;

            arrow++;

            send\_area++;

        }

*else* {

            printf("\"%s\" is received and recorded.\n", buf);

*// "ACK" 문자열 제거*

            char \*ptr = buf;

*while* (\*ptr && !isdigit(\*ptr)) {

                ++ptr;

            }

            int number;

*// 숫자 부분을 추출하여 정수로 변환*

*if* (\*ptr) {

                number = atoi(ptr);

            }

*// printf("number: %d\n\n", number);*

            isReceived[number] = true;

        }

        strcpy(lastACK, buf);

    }

    close(sock);

    pthread\_exit(NULL);

}

*// 서버로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

    bool loop = true;

*while* (1) {

*/\**

*if (fgets(buf, BUFSIZE + 1, stdin) == NULL)*

*break;*

*int len = (int)strlen(buf);*

*if (buf[len - 1] == '\n')*

*buf[len - 1] = '\0';*

*if (strlen(buf) == 0)*

*break;*

*\*/*

*if* (send\_area > 0 && send\_arrow < 6) {

*if* (isReceived[send\_arrow]) {

                send\_arrow++;

*continue*;

            }

            strcpy(buf, packets[send\_arrow]);

            retval = send(sock, buf, (int)strlen(buf), 0);

*if* (retval == SOCKET\_ERROR) {

                err\_display("send()");

*break*;

            }

            printf("\"%s\" is transmitted.\n", buf);

*// 0.1 second delay*

            usleep(100000);

            startTime[send\_arrow] = clock();

            send\_arrow++;

            send\_area--;

        }

*// 전부 다 받았다면 stop*

        int true\_count = 0;

*for* (int i = 0; i < 6; i++) {

*//printf("\nisReceived[%d]: %d\n", i, isReceived[i]);*

*if* (isReceived[i])

                true\_count++;

        }

*if* (true\_count == 6) *break*;

*for* (int i = arrow; i < send\_arrow; i++) {

            endTime[i] = clock();

            elapsedTime[i] = (double)(endTime[i] - startTime[i]) / CLOCKS\_PER\_SEC;

        }

*// Time Out*

*for* (int i = 0; i < send\_arrow; i++) {

*// 이미 받은 ACK라면, 건너뛰기*

*if* (isReceived[i] == true) *continue*;

*if* (elapsedTime[i] > 5) {

                printf("%s is timeout.\n", packets[i]);

*// 보낼 수 있는 공간 및 arrow 초기화*

                send\_area = 4;

                send\_arrow = i;

*break*;

            }

        }

    }

    close(sock);

    pthread\_exit(NULL);

}

int main(int *argc*, char \**argv*[]) {

    int retval;

*// 명령행 인수가 있으면 IP 주소로 사용*

    char \*SERVERIP = (argc > 1) ? argv[1] : (char \*)"127.0.0.1";

*// 소켓 생성*

    SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);

*if* (sock == INVALID\_SOCKET) err\_quit("socket()");

*// connect()*

    struct sockaddr\_in serveraddr;

    memset(&serveraddr, 0, sizeof(serveraddr));

    serveraddr.sin\_family = AF\_INET;

    inet\_pton(AF\_INET, SERVERIP, &serveraddr.sin\_addr);

    serveraddr.sin\_port = htons(SERVERPORT);

    retval = connect(sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr));

*if* (retval == SOCKET\_ERROR) err\_quit("connect()");

    pthread\_t r\_tid, s\_tid;

*// 서버로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 수신 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

*// 서버로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &sock) != 0) {

        printf("[TCP 클라이언트] 전송 스레드 생성 실패\n");

        close(sock);

        exit(1);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(sock);

*return* 0;

}

Receiver

*// 2142851 컴퓨터공학과 김형준*

*// Selective repeat Receiver*

*#include* "../Common.h"

*#include* <pthread.h>

*#include* <ctype.h>

*#define* BUFSIZE    512

*#define* SERVERPORT 9000

char ACKs[6][10] = { "ACK 0", "ACK 1", "ACK 2", "ACK 3", "ACK 4", "ACK 5" };

char packets[6][10] = { "packet 0", "packet 1", "packet 2", "packet 3", "packet 4", "packet 5" };

int receive\_arrow = 0;

int receive\_number = 0;

int last\_receive\_number = 0;

bool receive\_flag = false;

bool isReceived[7] = { false, false, false, false, false, false, false };

*// 받은 게 뭔가 이상한지 확인*

bool something\_wrong = false;

bool buffered = false;

bool firstPacket2Drop = true;

*// 클라이언트로부터 메시지를 수신하는 함수*

void \*receive\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// 데이터 받기*

        retval = recv(client\_sock, buf, BUFSIZE, 0);

*if* (retval == SOCKET\_ERROR) {

            err\_display("recv()");

*break*;

        } *else* *if* (retval == 0) {

            printf("[disconnected]\n");

*break*;

        }

*if* (firstPacket2Drop && (strcmp(buf, packets[2]) == 0)) {

            firstPacket2Drop = false;

*continue*;

        }

*// 받은 데이터 출력*

        buf[retval] = '\0';

*// printf("[TCP 서버] %d바이트를 받았습니다.\n", retval);// 받은 데이터 출력*

*if* (strcmp(buf, packets[receive\_arrow]) == 0) { *// 같다면 (잘 왔다면)*

*// 0.1 second delay*

            usleep(100000);

            printf("\"%s\" is received. ", buf);

            receive\_arrow++;

        }

*else* { *// 다르다면*

*// 0.1 second delay*

            usleep(100000);

            printf("\"%s\" is received and buffered. ", buf);

            something\_wrong = true;

        }

*// "packet" 문자열 제거*

        char \*ptr = buf;

*while* (\*ptr && !isdigit(\*ptr)) {

            ++ptr;

        }

*// 숫자 부분을 추출하여 정수로 변환*

*if* (\*ptr) {

            receive\_number = atoi(ptr);

        }

*if* (receive\_number < last\_receive\_number) {

            printf("%s, ", packets[receive\_number]);

            int last\_i;

*for* (int i = receive\_number + 1; isReceived[i + 1] == false; i++) {

                printf("%s, ", packets[i]);

                last\_i = i;

            }

            printf("and %s are delivered. ", packets[last\_i]);

            something\_wrong = false;

        }

        last\_receive\_number = receive\_number;

        isReceived[receive\_arrow] = true;

        receive\_flag = true;

*//printf("receive\_flag : %d\n\n", receive\_flag);*

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

*// 클라이언트로 메시지를 보내는 함수*

void \*send\_func(void \**arg*) {

    int client\_sock = \*((int \*)arg);

    char buf[BUFSIZE + 1];

    int retval;

*while* (1) {

*// if (fgets(buf, BUFSIZE + 1, stdin) == NULL)*

*//     break;*

*// int len = (int)strlen(buf);*

*// if (buf[len - 1] == '\n')*

*//     buf[len - 1] = '\0';*

*// if (strlen(buf) == 0)*

*//     break;*

*if* (receive\_flag) {

*// 1 second delay*

*// usleep(1000000);*

            strcpy(buf, ACKs[receive\_number]);

            retval = send(client\_sock, buf, (int)strlen(buf), 0);

*if* (retval == SOCKET\_ERROR) {

                err\_display("send()");

*break*;

            }

*if* (something\_wrong) {

                printf("\"%s\" is retransmitted. \n", buf);

            }

*else* {

                printf("\"%s\" is transmitted. \n", buf);

            }

            receive\_flag = false;

        }

    }

    close(client\_sock);

    pthread\_exit(NULL);

}

int main(int *argc*, char \**argv*[]) {

*// 소켓 생성*

    SOCKET listen\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

*if* (listen\_sock == INVALID\_SOCKET) err\_quit("socket()");

    struct sockaddr\_in serveraddr;

    memset(&serveraddr, 0, sizeof(serveraddr));

    serveraddr.sin\_family = AF\_INET;

    serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    serveraddr.sin\_port = htons(SERVERPORT);

*if* (bind(listen\_sock, (struct sockaddr \*)&serveraddr, sizeof(serveraddr)) == SOCKET\_ERROR)

        err\_quit("bind()");

*if* (listen(listen\_sock, SOMAXCONN) == SOCKET\_ERROR)

        err\_quit("listen()");

    struct sockaddr\_in clientaddr;

    socklen\_t addrlen = sizeof(clientaddr);

    SOCKET client\_sock = accept(listen\_sock, (struct sockaddr \*)&clientaddr, &addrlen);

*if* (client\_sock == INVALID\_SOCKET) {

        err\_display("accept()");

    }

    pthread\_t r\_tid, s\_tid;

*// 클라이언트로부터 메시지를 수신하는 스레드 생성*

*if* (pthread\_create(&r\_tid, NULL, receive\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 수신 스레드 생성 실패\n");

        close(client\_sock);

    }

*// 클라이언트로 메시지를 보내는 스레드 생성*

*if* (pthread\_create(&s\_tid, NULL, send\_func, &client\_sock) != 0) {

        printf("[TCP 서버] 전송 스레드 생성 실패\n");

        close(client\_sock);

    }

    pthread\_join(s\_tid, NULL);

*// 소켓 닫기*

    close(listen\_sock);

*return* 0;

}

1. **결과 출력**

Sender

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Receiver

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. **어려웠던 점**

Go-Back-N 함수에서 조금만 수정하였더니, 잘 실행되었습니다. 다만, Go-Back-N에서는 마지막 ACK(ACK 5)를 받았을 때 종료하는 것으로 설정하였지만, Select Repeat 알고리즘에서는 이런 식으로 코딩할 경우 코드가 일찍 종료되어버리는 문제점이 생겼습니다. 이 부분에서는 Go-Back-N 함수에서 사용하진 않지만 혹시 몰라 만들어 놓았던 isReceived(bool) 변수를 사용하여 전부 받았는지 확인 후 전부 받았다면 종료시키는 방식으로 변경하였습니다.

또한, Receiver에서도 isReceived 함수를 설정하며 buffer에 저장이 되었는지를 확인하는 기능을 만들었습니다. Go-Back-N 알고리즘에서 쌩코딩으로 많은 시간을 쏟았던 만큼, 변형 부분에서는 많은 시간이 걸리지는 않았습니다.