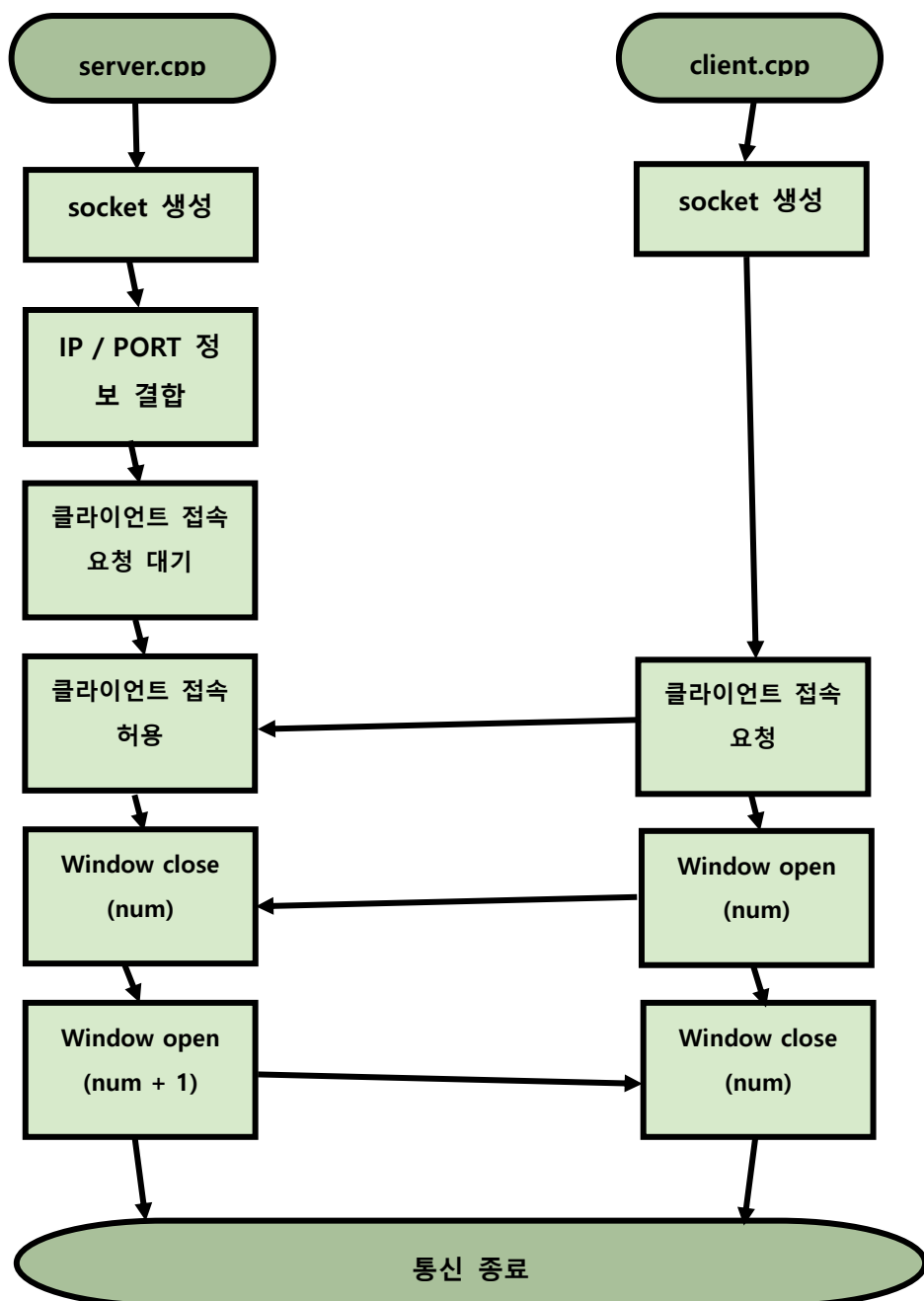


<실행 환경 설명>

AWS EC2 인스턴스를 활용하여 두 코드중 server.cpp는 AWS 인스턴스에서, client.cpp는 cmd를 통해 ubuntu로 만든 서버에 접속하여 실행하려 하였으나, cpp 파일을 인스턴스나 ubuntu 서버에 옮기는 과정이 의도대로 되지 않아 cmd창을 두 개 열어 놓고, 제 PC의 IP주소로 접속하여 같은 PORT를 입력하여 통신이 실행되는 환경을 만들었습니다.

<실행도>



<server.cpp>

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<WinSock2.h>
```

<stdio.h> - printf()

<stdlib.h> - exit()

<string.h>

- memset() : 배열이나 구조체 등을 초기화
- strlen() : 문자열 길이 반환
- strcmp() : 문자가 같은지 비교

<WinSock2.h>

- WSASocket() : 소켓 라이브러리 초기화
- socket() : 소켓 생성
- bind() : 소켓에 IP 주소와 PORT를 바인딩
- listen() : 소켓을 대기 모드로 설정하고 클라이언트의 연결을 대기
- accept() : 클라이언트의 연결을 수락
- send() : 소켓을 통해 데이터를 전송
- recv() : 소켓을 통해 데이터를 수신
- closesocket() : 소켓을 닫음
- WSACleanup() : 소켓 라이브러리를 정리하고 사용한 자원을 해제
- WSADATA : Windows 소켓 초기화에 사용되는 구조체
- SOCKET : 소켓을 나타내는 데이터 타입
- SOCKADDR_IN : 소켓의 주소 정보를 저장하는 구조체

```

void ErrorHandler(const char* message) {
    printf("%s\n", message);
}

int CheckNum(int n) {
    if (n == -1) {
        // 수신 실패 처리
        printf("Checking Error!");
        exit(0);
    }
    else {
        // 수신한 데이터 크기가 int 크기와 일치하는 경우
        n = *reinterpret_cast<int*>(buffer);
        return n;
    }
}

int ModNum(int n) {
    n = n % 8;
    return n;
}

```

ErrorHandler() : 특정 함수가 정상적으로 시행되지 않았을 때 에러가 발생했음을 알리는 출력 함수

CheckNum() : 상대 측에서 데이터를 char 형태에서 int 형으로 바꾸며, 수신 처리 실패시 프로그램을 종료하는 함수

ModNum : 총 window size = 8 로 가정하여 8 이상일 때 나머지 연산자를 사용하여 반환하는 함수

```

struct frame {
    int seq;           // 프레임의 시퀀스 번호
    int ack = 0;       // 프레임의 응답 번호
    int pack_num = 0;  // 프레임의 패킷 번호
    char m[msize];     // 프레임의 메시지
};

int main(int argc, char* argv[]) {
    WSADATA wsaData;           // Winsock의 초기화 정보를 저장하는 구조체
    SOCKET hServSock, hClntSock; // 서버 소켓과 클라이언트 소켓의 핸들
    SOCKADDR_IN servAddr, clntAddr; // 서버 주소 정보와 클라이언트 주소 정보

    int strLen;                // 클라이언트로부터 온 메시지 길이를 담는 변수
    int szClntAddr;            // 클라이언트 주소 정보의 크기를 담는 변수
    char msg[] = "This message is ack\n"; // 응답 메시지
    char message[] = "Hello World!"; // 클라이언트와 연결 성공 시 보낼 메시지
    int recv_seq;              // 수신한 프레임의 시퀀스 번호
    int recv_ack;              // 수신한 프레임의 응답 번호
    int recv_pack_num = 0;     // 수신한 프레임의 패킷 번호
    frame server_f;
}

```

```

// 소켓 라이브러리 초기화
if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    ErrorHandling("WSAStartup() error!");

hServSock = socket(PF_INET, SOCK_STREAM, 0);
if (hServSock == INVALID_SOCKET)
    ErrorHandling("socket() error!");

memset(&servAddr, 0, sizeof(servAddr));
servAddr.sin_family = AF_INET;
servAddr.sin_port = htons(atoi(argv[1]));
servAddr.sin_addr.s_addr = htonl(INADDR_ANY);

if (bind(hServSock, (SOCKADDR*)&servAddr, sizeof(servAddr)) == SOCKET_ERROR)
    ErrorHandling("bind() error!");

if (listen(hServSock, 5) == SOCKET_ERROR)
    ErrorHandling("listen() error!");

szClntAddr = sizeof(clntAddr);
hClntSock = accept(hServSock, (SOCKADDR*)&clntAddr, &szClntAddr);
if (hClntSock == INVALID_SOCKET) {
    ;
    ErrorHandling("Accept Error!");
}

send(hClntSock, message, sizeof(message), 0);

```

1. 소켓 라이브러리 초기화
2. hServSock 이름의 소켓 생성
3. IP 주소와 PORT 번호를 받을 servAddr 설정
4. hServSock 소켓에 servAddr 데이터를 적용
5. 클라이언트 측으로부터 연결을 대기
6. szClntAddr 변수에 클라이언트 주소 정보를 담고 연결을 수락
7. 클라이언트에 연결이 성공했다는 메시지 전송

```

while (1) {
    strLen = recv(hClntSock, server_f.m, sizeof(server_f.m), 0);
    if (strcmp(server_f.m, "exit\n") == 0 || strLen <= 0) { // "exit"과 일치하는지 확인 or 오류 발생
        printf("Close client Connection...\n");
        break;
    }
    if (strLen > 0) {
        recv_seq = CheckNum(recv(hClntSock, buffer, sizeof(int), 0));
        recv_ack = CheckNum(recv(hClntSock, buffer, sizeof(int), 0));
        recv_pack_num = CheckNum(recv(hClntSock, buffer, sizeof(int), 0));

        printf("-----\n");
        printf("| window's range : %d ~ %d |\n", ModNum(recv_pack_num), ModNum(recv_pack_num + 4));
        printf("-----\n");

        printf("[Client] : %s", server_f.m);
        printf("Client's frame = (%d, %d, A%d)\n\n", recv_seq, recv_ack, recv_pack_num);

        send(hClntSock, msg, sizeof(msg), 0);
    }
}

```

1. 클라이언트로부터 특정 메시지를 메시지를 받음
2. strLen 변수를 통해 정상적으로 데이터를 받았는지 확인
3. 정상적일 경우 클라이언트의 frame 의 seq, ack, pack_num 정보를 받음
4. 현재 열려 있는 window의 범위를 출력
5. 클라이언트로부터 어떤 메시지가 왔고 받은 프레임의 seq, ack, pack_num 을 출력
(pdf의 A one-bit Sliding Window Protocol 모델을 기준으로 프레임의 정보를 출력함)
6. 클라이언트 측에 데이터가 성공적으로 도착했다는 ack를 보냄

```

closesocket(hClntSock);
closesocket(hServSock);
WSACleanup();
return 0;

```

연결이 종료되면 각 소켓을 닫고 자원을 반환 후 프로그램 종료

<client.cpp>

```
#include<WinSock2.h>
#include<Ws2tcpip.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

『server.cpp에서 설명한 부분은 생략』

<Ws2tcpip.h> - inet_pton() : IP 주소를 텍스트에서 바이너리 형식으로 변환

```
void ErrorHandler(const char* message) {
    printf("%s", message);
}

int CheckNum(int n) {
    if (n == -1) {
        // 수신 실패 처리
        printf("Checking Error!");
        exit(0);
    }
    else {
        // 수신한 데이터 크기가 int 크기와 일치하는 경우
        n = *reinterpret_cast<int*>(buffer);
        return n;
    }
}

int ChangeNum(int n) {
    if (n == 1) return 0;
    else return 1;
}

int ModNum(int n) {
    n = n % 8;
    return n;
}
```

『server.cpp에서 설명한 부분은 생략』

ChangeNum() : 데이터를 보낸 후 seq와, 받은 후의 ack 값을 변환시키기 위한 함수

```

struct frame{
    int seq = 0;
    int ack = 1;
    int pack_num = 0;
    char m[msize];
};

int main(int argc, char* argv[]) {
    WSADATA wsaData;           // winsock 초기화에 필요한 정보를 저장하는 구조체
    SOCKET hSocket;            // 소켓 핸들
    SOCKADDR_IN servAddr;      // 서버 주소 정보를 담는 구조체

    int strLen;                // 클라이언트로부터 온 메시지 길이를 담는 변수
    char message[30];          // 서버로부터 받은 메시지를 저장하는 배열

    int recv_seq;              // 수신한 프레임의 시퀀스 번호를 저장하는 변수
    int recv_ack;              // 수신한 프레임의 ACK 번호를 저장하는 변수
    int recv_pack_num;         // 수신한 프레임의 패킷 번호를 저장하는 변수
    frame client_f;            // 클라이언트 프레임을 나타내는 구조체

```

```

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    ErrorHandling("WSAStartup() error!");

hSocket = socket(PF_INET, SOCK_STREAM, 0);
if (hSocket == INVALID_SOCKET)
    ErrorHandling("socket() error!");

memset(&servAddr, 0, sizeof(servAddr));
servAddr.sin_family = AF_INET;
if (inet_pton(AF_INET, argv[1], &(servAddr.sin_addr)) <= 0)
    ErrorHandling("inet_pton() error!");    servAddr.sin_port = htons(atoi(argv[2]));
if (connect(hSocket, (SOCKADDR*)&servAddr, sizeof(servAddr)) == SOCKET_ERROR)
    ErrorHandling("Connect() error!");

strLen = recv(hSocket, message, sizeof(message) - 1, 0);
if (strLen == -1) ErrorHandling("read() error!");
printf("Messasge from server : %s\n", message);

```

『server.cpp에서 설명한 부분은 생략』

inet_pton() 함수를 통해 IP주소와 PORT 번호를 받아 servAddr에 데이터를 삽입

```

while (1) {
    printf("input message : ");
    fgets(client_f.m, 256, stdin);
    if (strcmp(client_f.m, "exit\n") == 0) {
        printf("\nClose Server Connection...\n");
        break;
    }

    send(hSocket, client_f.m, sizeof(client_f.m), 0);
    if (send(hSocket, reinterpret_cast<char*>(&client_f.seq), sizeof(int), 0) == -1 ||
        send(hSocket, reinterpret_cast<char*>(&client_f.ack), sizeof(int), 0) == -1 ||
        send(hSocket, reinterpret_cast<char*>(&client_f.pack_num), sizeof(int), 0) == -1) {
        // 전송 실패 처리
        ErrorHandling("Sending error!");
        break;
    }

    client_f.ack = ChangeNum(client_f.ack);
    // recv(hSocket, message, sizeof(message), 0);
    printf("-----\n");
    printf("| Open window%d |\n", ModNum(client_f.pack_num));
    if (recv(hSocket, message, sizeof(message), 0) > 0) {
        printf("[Server] : %s", message);
        client_f.seq = ChangeNum(client_f.seq);
        client_f.pack_num += 1;
    }

    printf("| Close window%d|\n", ModNum(client_f.pack_num));
    printf("-----\n");

    // 의도대로 기능하지 않아 주석 처리한 부분입니다
    // recv_seq = CheckNum(recv(hSocket, buffer, sizeof(int), 0));
    // recv_ack = CheckNum(recv(hSocket, buffer, sizeof(int), 0));
    // recv_pack_num = CheckNum(recv(hSocket, buffer, sizeof(int), 0));
    // client_f.ack = ChangeNum(client_f.ack);

    // printf("Server's frame = (%d, %d, B%d)\n\n", recv_seq, recv_ack, recv_pack_num);
}

```

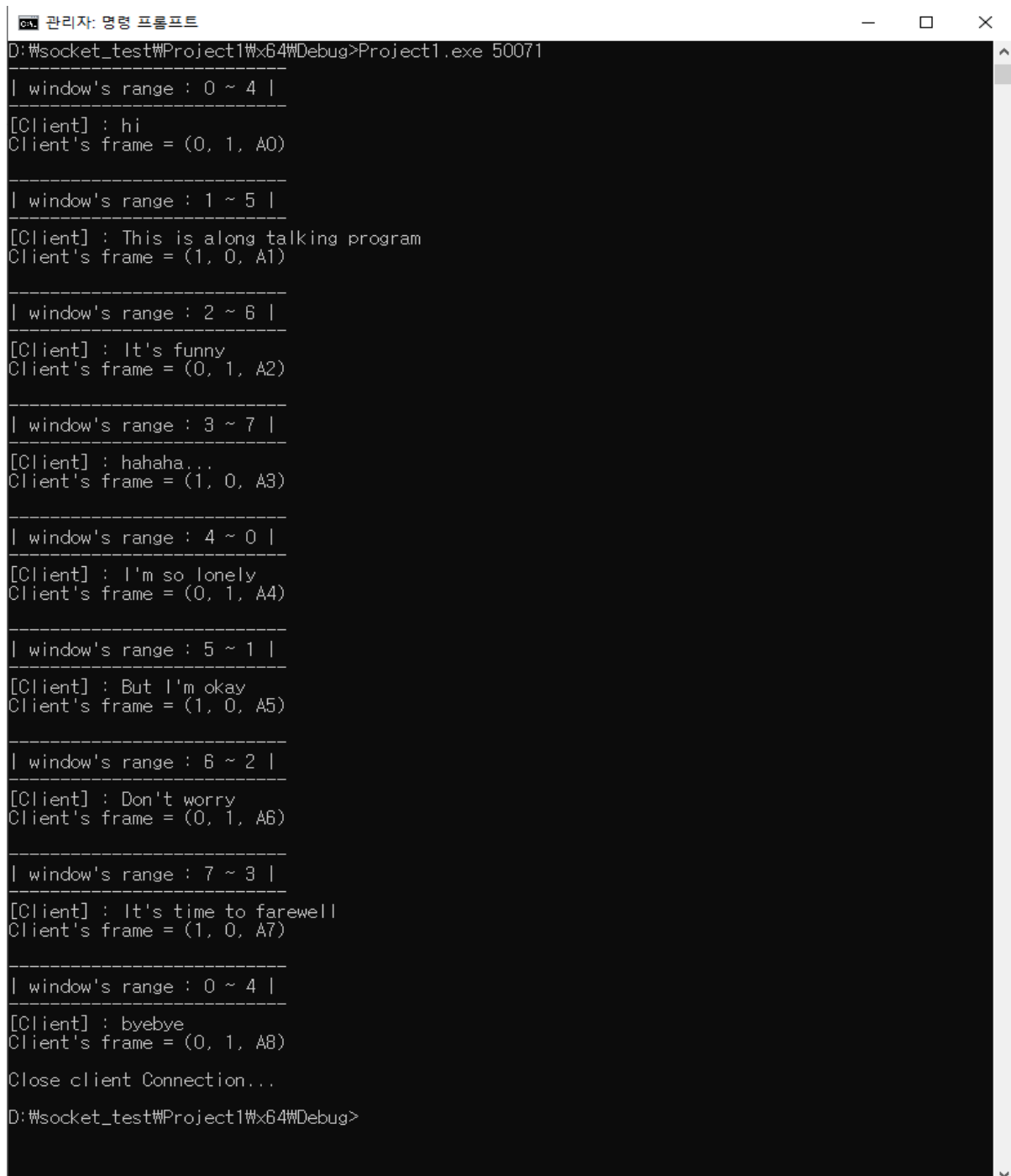
1. fgets() 함수를 통해 클라이언트의 frame 에 메시지를 저장하고 해당 메시지가 "exit" 인지 확인
2. "exit"라면 연결을 끊고 프로그램 종료
3. 아닐 경우 해당 메시지를 서버로 전송 + 클라이언트의 seq, ack, pack_num을 char 형식으로 변환해 전송
4. 전송 후, ChangeNum() 함수를 통해 ack를 변경
5. Window open / close 상태를 출력하고 서버로부터 ack를 받았을 경우 seq와 pack_num 값 변화 및 증가
- # 6. 원래 계획은 서버에서 보낸 frame의 seq, ack, pack_num 을 출력하는 과정을 거치려 하였으나 정상적으로 작동하지 않아 주석처리 함


```
closeSocket(hSocket);  
WSACleanup();  
return 0;
```

server.cpp 와 동일

<결과 화면>

- server.cpp



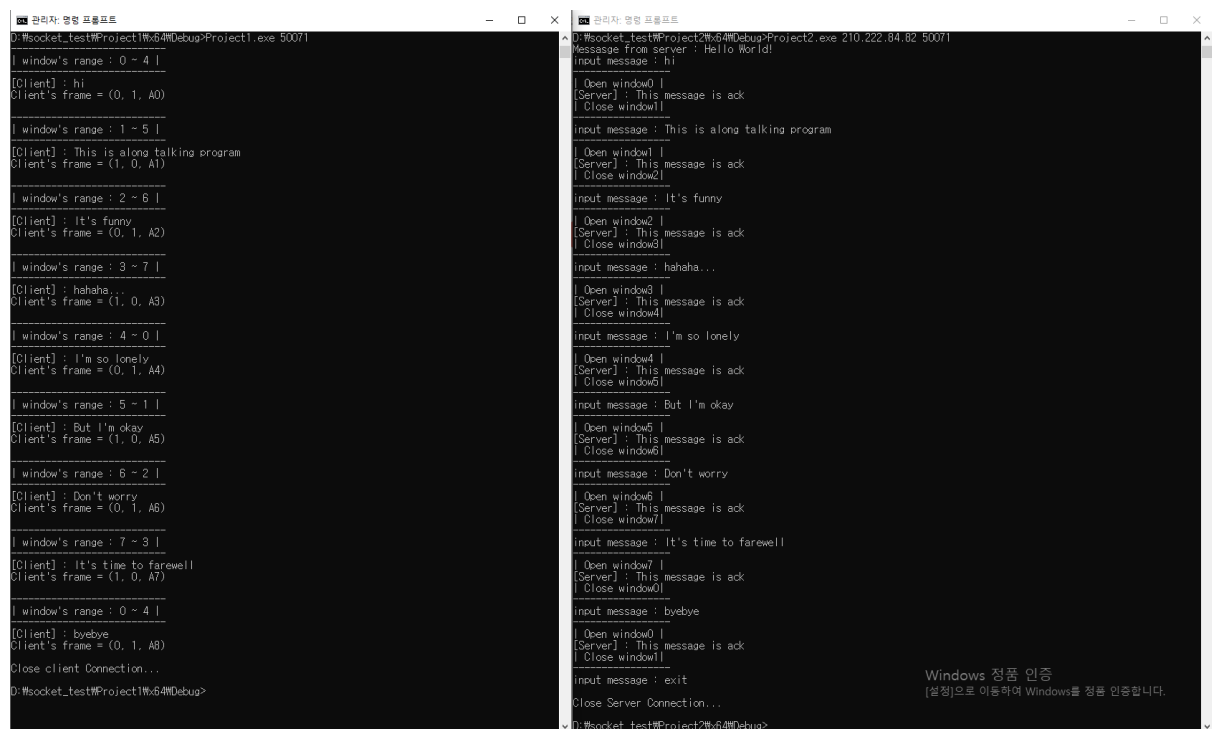
```
관리자: 명령 프롬프트  
D:\socket_test\Project1\x64\Debug>Project1.exe 50071  
-----  
| window's range : 0 ~ 4 |  
-----  
[Client] : hi  
Client's frame = (0, 1, A0)  
-----  
| window's range : 1 ~ 5 |  
-----  
[Client] : This is along talking program  
Client's frame = (1, 0, A1)  
-----  
| window's range : 2 ~ 6 |  
-----  
[Client] : It's funny  
Client's frame = (0, 1, A2)  
-----  
| window's range : 3 ~ 7 |  
-----  
[Client] : hahaha...  
Client's frame = (1, 0, A3)  
-----  
| window's range : 4 ~ 0 |  
-----  
[Client] : I'm so lonely  
Client's frame = (0, 1, A4)  
-----  
| window's range : 5 ~ 1 |  
-----  
[Client] : But I'm okay  
Client's frame = (1, 0, A5)  
-----  
| window's range : 6 ~ 2 |  
-----  
[Client] : Don't worry  
Client's frame = (0, 1, A6)  
-----  
| window's range : 7 ~ 3 |  
-----  
[Client] : It's time to farewell  
Client's frame = (1, 0, A7)  
-----  
| window's range : 0 ~ 4 |  
-----  
[Client] : byebye  
Client's frame = (0, 1, A8)  
Close client Connection...  
D:\socket_test\Project1\x64\Debug>
```

- client.cpp

```
관리자: 명령 프롬프트
D:\#socket_test#\Project2#\x64#\Debug>Project2.exe 210.222.84.82 50071
Message from server : Hello World!
input message : hi
-----
| Open window0 |
[Server] : This message is ack
| Close window1 |
-----
input message : This is along talking program
-----
| Open window1 |
[Server] : This message is ack
| Close window2 |
-----
input message : It's funny
-----
| Open window2 |
[Server] : This message is ack
| Close window3 |
-----
input message : hahaha...
-----
| Open window3 |
[Server] : This message is ack
| Close window4 |
-----
input message : I'm so lonely
-----
| Open window4 |
[Server] : This message is ack
| Close window5 |
-----
input message : But I'm okay
-----
| Open window5 |
[Server] : This message is ack
| Close window6 |
-----
input message : Don't worry
-----
| Open window6 |
[Server] : This message is ack
| Close window7 |
-----
input message : It's time to farewell
-----
| Open window7 |
[Server] : This message is ack
| Close window0 |
-----
input message : byebye
-----
| Open window0 |
[Server] : This message is ack
| Close window1 |
-----
input message : exit
Close Server Connection...
D:\#socket_test#\Project2#\x64#\Debug>
```

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.

- 동시화면



```
D:\socket_test\Project1\Win64\Debug>Project1.exe 50071
| window's range : 0 ~ 4 |
[Client] : hi
Client's frame = (0, 1, A0)

| window's range : 1 ~ 5 |
[Client] : This is along talking program
Client's frame = (1, 0, A1)

| window's range : 2 ~ 6 |
[Client] : It's funny
Client's frame = (0, 1, A2)

| window's range : 3 ~ 7 |
[Client] : hahaha...
Client's frame = (1, 0, A3)

| window's range : 4 ~ 0 |
[Client] : I'm so lonely
Client's frame = (0, 1, A4)

| window's range : 5 ~ 1 |
[Client] : But I'm okay
Client's frame = (1, 0, A5)

| window's range : 6 ~ 2 |
[Client] : Don't worry
Client's frame = (0, 1, A6)

| window's range : 7 ~ 3 |
[Client] : It's time to farewell
Client's frame = (1, 0, A7)

| window's range : 0 ~ 4 |
[Client] : goodbye
Client's frame = (0, 1, A8)
Close client Connection...
D:\socket_test\Project1\Win64\Debug>
```

```
D:\socket_test\Project2\Win64\Debug>Project2.exe 210.222.84.82 50071
Message from server : Hello World!
input message : hi
| Open window0 |
[Server] : This message is ack
| Close window0 |

input message : This is along talking program
| Open window1 |
[Server] : This message is ack
| Close window1 |

input message : It's funny
| Open window2 |
[Server] : This message is ack
| Close window2 |

input message : hahaha...
| Open window3 |
[Server] : This message is ack
| Close window3 |

input message : I'm so lonely
| Open window4 |
[Server] : This message is ack
| Close window4 |

input message : But I'm okay
| Open window5 |
[Server] : This message is ack
| Close window5 |

input message : Don't worry
| Open window6 |
[Server] : This message is ack
| Close window6 |

input message : It's time to farewell
| Open window7 |
[Server] : This message is ack
| Close window7 |

input message : goodbye
| Open window0 |
[Server] : This message is ack
| Close window0 |

input message : exit
Close Server Connection...
D:\socket_test\Project2\Win64\Debug>
```

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.

<과제 후기>

처음 AWS를 접했을 때에는 사용법을 몰라 여러 자료들을 찾아보았다. 그 과정을 통해 AWS라는 서비스가 어떤 것인지 알게 되었고 굉장히 유용한 서비스인 것을 알게 되었다. 인스턴스 생성도 해보고 인증키 방식에 대해서도 알게 되었다. 비록 AWS를 통해 프로그램을 실행시켜 확인하지 못 하였지만 클라우드 서버 프로그래밍, 소켓 통신에 대해 알게 된 시간을 가질 수 있었다. Sliding window protocol의 작동 원리에 대해 생각해 보는 시간도 가졌으며, 이를 구현해 보는 재미도 느낄 수 있어 의미있는 시간이었다고 생각한다.