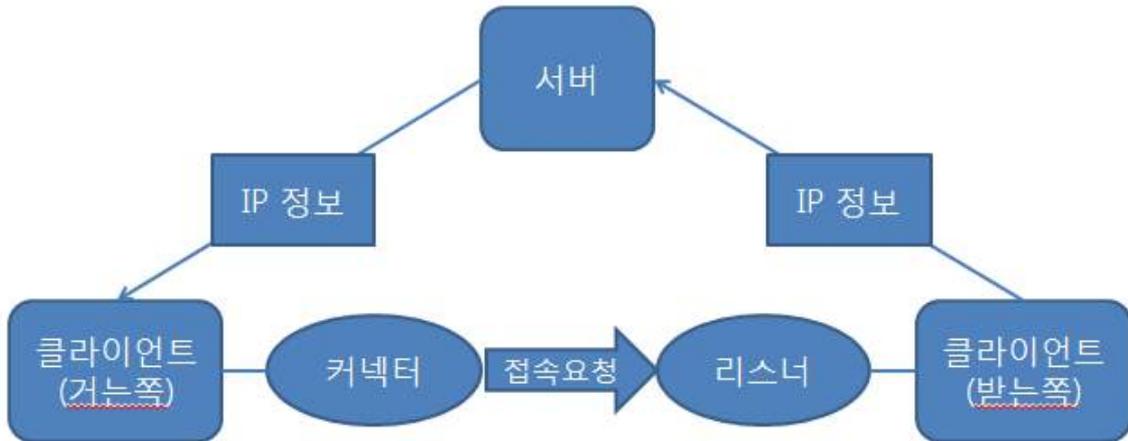


Graduation Project Mid-Report(Spring)

| | | | | |
|--|--------------------|------------|-------------------------|---------------|
| Title | 인터넷을 통한 음성통화(Voip) | | | |
| Members | Name | Student ID | E-mail | Cell Phone |
| | 김국영 | 200011436 | webmaster@game.re.kr | 010-5305-9471 |
| | 장기웅 | 200511349 | wkdrldnd09@nate.com | 010-7184-8499 |
| | 김재홍 | 200010639 | muscovite81@hotmail.com | 010-7760-3360 |
| Advisor | 유준범 교수님 | | | |
| Main Function | | | | |
| <p>개발 내용 및 최종 목표</p> <p>○ 개발 내용</p> <ul style="list-style-type: none"> - 윈도우즈 기반의 어플리케이션 - 서버 및 클라이언트를 구현하여 서버는 각 클라이언트들간의 접속에 필요한 정보를 제공할 수 있게 구현한다. - 클라이언트는 서버에서 접속할 다른 클라이언트(들)의 정보를 얻고 클라이언트와 접속하여 음성통화를 할 수 있게 구현한다. - 음성압축기술을 구현하여 이를 통한 음성데이터 전송을 구현한다. <p>○ 최종 목표</p> <ul style="list-style-type: none"> - 인터넷을 통한 무료 통화 구현 - 안정적인 접속이 가능한 어플리케이션 구현 - 기존 유선통화보다 더 좋은 음질의 통화가 가능한 어플리케이션 구현 - 보안을 위한 인증서를 통한 인증 기능(계획서 외 추가된 기능) | | | | |

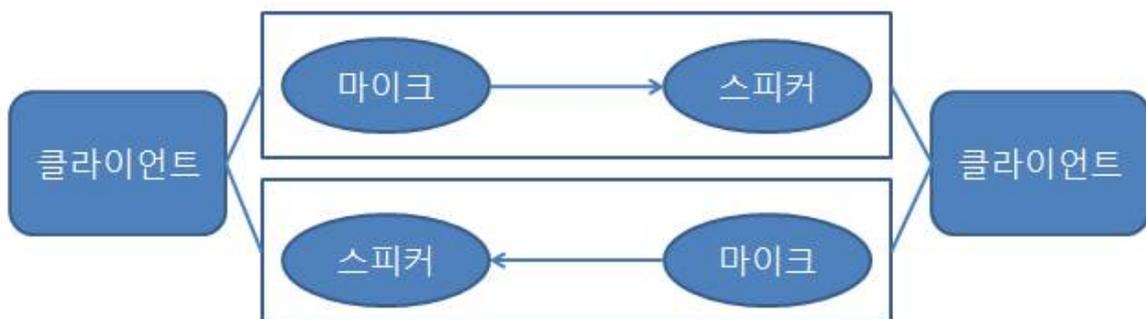
설계 구조 내용

○ 클라이언트 간의 접속



- 클라이언트의 IP 정보는 서버가 가지고 있을 수 있도록 한다.
- 클라이언트는 접속할 다른 클라이언트의 IP 정보를 서버에게 요청하고 그 정보를 서버로부터 제공 받는다.
- 클라이언트의 커넥터는 그 IP 정보를 가지고 접속할 클라이언트에게 접속 요청을 한다.
- 접속요청을 받는 클라이언트의 리스너가 이 접속 요청을 받고 클라이언트 간의 접속을 시행한다.
- 클라이언트의 리스너는 항상 접속에 대비하여 대기하고 있도록 한다.

○ 접속 후 클라이언트 간의 관계



- 각 클라이언트는 통화 중에 혼선이 일어나면 안되므로 마이크와 스피커를 각각 쓰레드로 구현한다.
- 각 클라이언트간의 스피커와 마이크 사이에서 40kbyte/s의 속도로 음성정보에 대한 패킷을 각각 전달받을 수 있게 구현한다.
- 각 클라이언트의 마이크와 스피커는 윈도우의 저수준 API를 사용하여 구현한다.

○ 서버의 구조

- 서버는 mssql을 이용한 데이터베이스를 사용한다.
- 데이터 베이스는 각 클라이언트의 id 정보와 IP 정보와 각종 개인정보를 가진다.
- 데이터 베이스의 구조 - 서버는 다음과 같은 구조의 데이터베이스를 갖는다.

Table - chat

| 열 이름 | 데이터 형식 | 정보 |
|----------|--------------|----------------------------|
| Id | Varchar(16) | Id 정보는 개인소지한 핸드폰 번호를 사용한다. |
| Pwd | Varchar(16) | Password |
| Name | Varchar(20) | 이름 |
| Nickname | Varchar(100) | 별명 |
| email | Varchar(100) | 이메일 주소 |

Table - pass

| 열 이름 | 데이터 형식 | 정보 |
|-------|-------------|----------------------------|
| Id | Varchar(16) | Id 정보는 개인소지한 핸드폰 번호를 사용한다. |
| Ip | char(15) | Ip 정보 |
| State | char(20) | 현재접속상태 |

- Id는 각 사용자들의 개인 핸드폰 번호를 사용한다. 이를 위해 Id 등록시 openssl 라이브러리를 사용해서 인증서를 핸드폰을 통해 인증하게끔 구현한다. (아직 미구현된 내용)

○ 클라이언트 헤더 파일 구조

```

Phone.h
class CPhone
{
public:
    CPhone();
    virtual ~CPhone();
    CNetwork *_m_Network;
    CDevice *_m_Device;
    void Test();
    int VoiceSend(char *buf, int len);
    int VoiceRecv(char *buf, int len);

    int Connect(CString Ip, int Port, int Is);
    int StartServer(int Port);
    int StartDevice();
    int StartTalk();
    int StopTalk();
    FILE *_m_LogFp;
    int OpenDebugLog(int Level);
    int _m_DebugLevel; // 0: debug 안함 1:
    void WriteLog(int Level, char *Fmt, ...);

    CString _m_PhoneNumber;
    tagOption _m_Option;
    tagOption & GetOption();
    int LoadOption();
    int SaveOption();
    //전 화번호로 서버에 조회해서 아이피를
    int GetIpByPhoneNumber(CString PhoneNu
    //전 화번호와 포트 아이피를 서버에 등록
    int RegisterPhoneNumber(tagOption &Inf
};

Network.h
class CNetwork
{
public:
    CNetwork();
    virtual ~CNetwork();
    CPhone *_m_Phone;
    void Init(CPhone *p){
        _m_Phone=p;
    }
    int PhoneConnect(CString Ip, int Port, i
    int PhoneSendPacket(char *packet, int l
    int PhoneRecvPacket(char *packet, int m
    int ServThread();

    //recv send 블럭모드로 패키징
    int SendPacket(char *packet, int len);
    int RecvPacket(char *packet, int maxlen

    int CommSendThread();
    int CommRecvThread();

    int StartServer(int Port);
    int StartComm(unsigned int Sock, int Is
    int StopComm();

    unsigned int _m_ConSock;

    CDataQueue _m_SendDataQueue;
    CDataQueue _m_RecvDataQueue;
    int _m_IsSecure; //0:아님 1: 보안모드 켜
    int _m_ServPort;
    int _m_PhoneState; //0: 서버리스닝중 1:
    int _m_IsCon; //0 연결안됨 1: 연결됨
};

Device.h
class CDevice
{
public:
    CDevice();
    virtual ~CDevice();
    CPhone *_m_Phone;
    void Init(CPhone *p);
    int _m_nSamplesPerSec;
    int _m_nChannels;
    int _m_wBitsPerSample;
    int _m_DeviceNum;

    int RecordThread();
    int SpeakerThread();

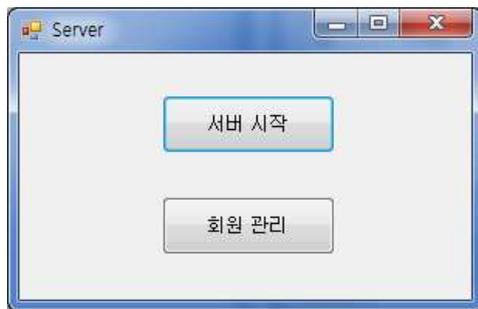
    int StartRecord();
    int StopRecord();

    int StartSpeaker();
    int StopSpeaker();

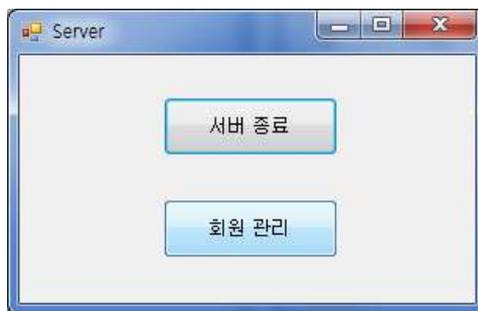
    int RecvMic(char *buf, int len);
    int SendSpeaker(char *buf, int len);
    int GetDevice(tagDeviceData *Device, ir
    void SetDevice(int DeviceNum, int nSamc
    void waveInProc(HWAVEIN hwi, UINT uMsg,
    void waveOutProc(HWAVEOUT hwi, UINT uMs
};
    
```

구현된 내용 및 사항

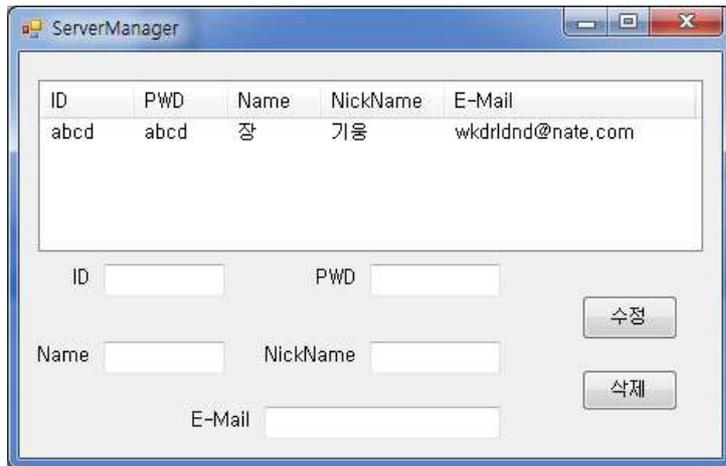
○ 서버



- 서버 프로그램 실행 후 서버시작 버튼을 누르면 서버가 동작한다.

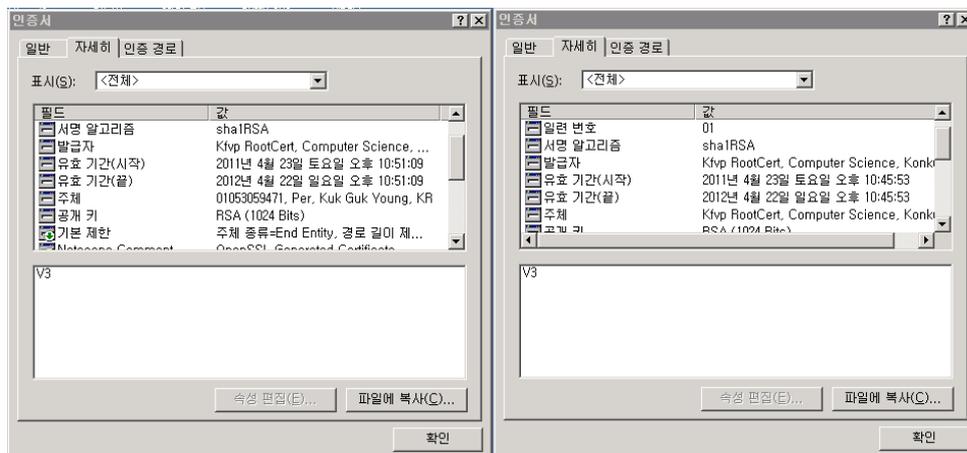


- 회원 관리 버튼을 누르면 아래와 같은 창이 뜨게 되고 회원 정보를 관리할 수 있다.

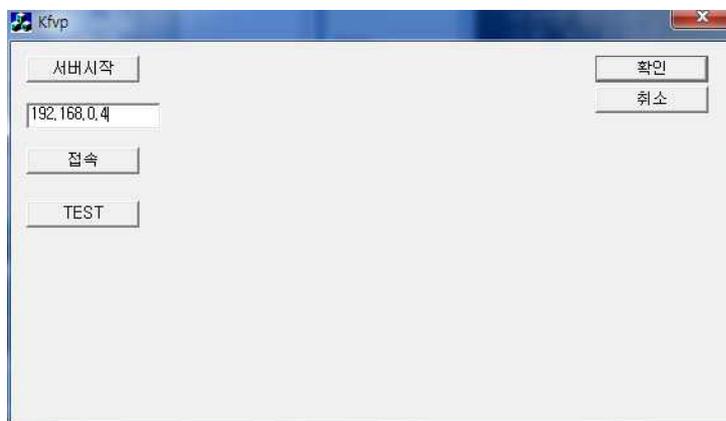


- 현재 임의의 값을 입력하였고 추후 ID에는 개인 휴대폰 번호 정보를 저장할 계획이다.

- Id 정보 저장시에 사용될 인증서에 대한 내용(미구현단계)

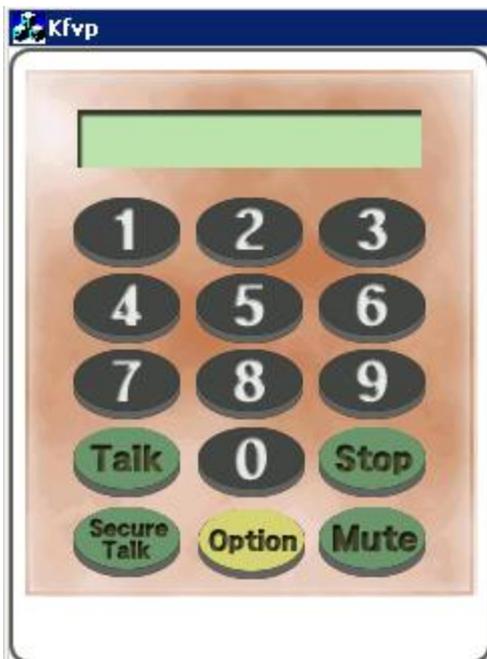


- 테스트 용 클라이언트 실행 화면



- 이 프로그램은 서버와 관계없이 클라이언트 간의 접속을 통해 통화기능에 대한 체크를 위한 테스트 프로그램이다.
- 전화를 받는쪽은 서버시작 버튼을 누르고 대기한다.
- 전화를 거는쪽은 받는쪽의 ip 주소를 입력후 접속을 누르고 Test 버튼을 누른다.
- 테스트시 양쪽다 원활한 통화가 가능 하였다.
- 사용하는 컴퓨터에 마이크가 두개이상 설정되어있으면 원하는 마이크를 잡지못하는 문제가 발생하였다. (추후 기능 추가 필요)

○ 최종 클라이언트 실행시 화면



- 위와 같은 UI를 이용하여 구현할 예정(미구현단계)
- 녹색 창에 접속 정보와 접속할 상대의 전화번호에 대한 정보를 출력할 예정(미구현단계)

최종 기한까지 구현할 내용 및 개선할 사항

○ 서버

- id 등록시 인증서를 통한 등록 구현
- 클라이언트와의 연동

○ 클라이언트

- 서버와의 연동
- 새로운 UI를 이용한 구현
- 마이크 장치와의 연결 문제점 해결