데이터 통신 00분반

Hw05 Chat & File

201402447 한원희

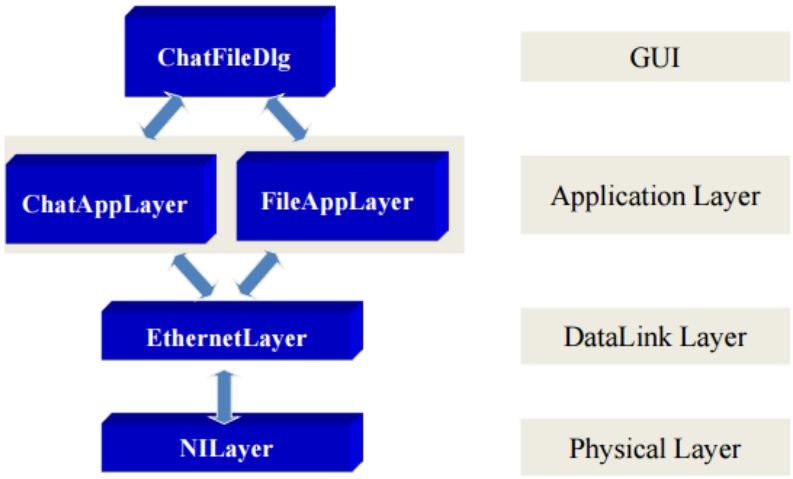
실습 일시 및 장소, 실험 환경

vm웨어 2개를 사용해 가상 ethernet사용했습니다.

실습 시나리오

- 두개의 가상 vmware를 사용해 가상 이더넷 주소를 통해 호스트가 연결되어 있음
- 두개의 vmware에서 각각 이클립스 실행
- 목적지에 서로의 주소를 적음
- 채팅과 파일을 보내 도착했는지 확인
- Wireshark를 통해 패킷 캡쳐

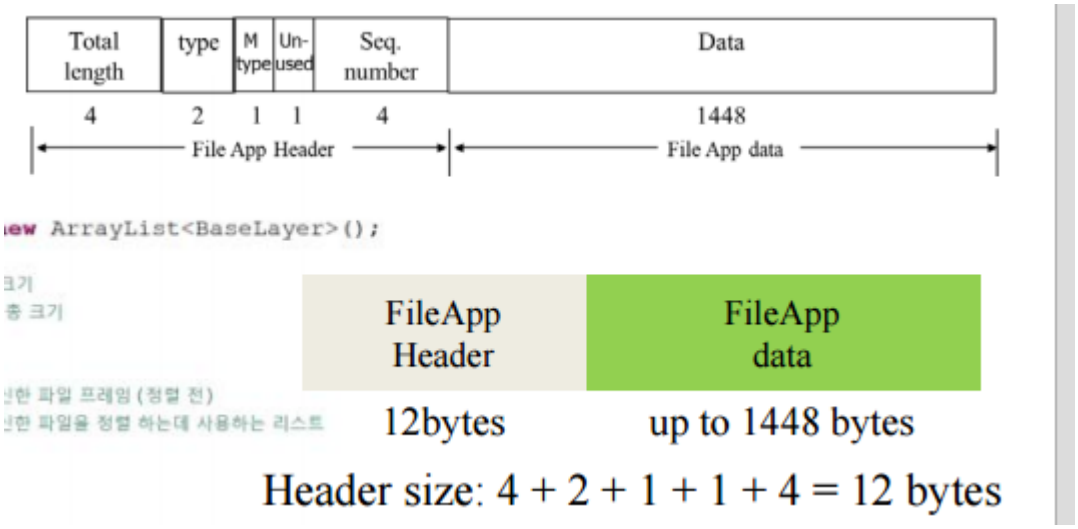
프로토콜 스택



계층은 stop & wait 때와의 차이가 크지 않습니다. ChatAppLayer와 같은 계층인 FileAppLayer가 추가되었습니다. File을 송수신할 때 사용하는 계층입니다.

```
public class _FAPP_HEADER {
    byte[] fapp_totlen;
    byte[] fapp_type;
    byte fapp_msg_type;
    byte fapp_unused;
    byte[] fapp_seq_num;
    byte[] fapp_data;

    public _FAPP_HEADER() {
        this.fapp_totlen = new byte[4];
        this.fapp_type = new byte[2];
        this.fapp_msg_type = 0x00;
        this.fapp_unused = 0x00;
        this.fapp_seq_num = new byte[4];
        this.fapp_data = null;
    }
}
```

FileAppLayer의 헤더는



이런식으로 되어 있습니다. 헤더의 길이는 12bytes이고 그 뒤는 파일의 data가 들어 있습니다.

```
private void setFragmentation(int type){
    if(type == 0) { // 처음
        m_sHeader.fapp_type[0] = (byte) 0x0;
        m_sHeader.fapp_type[1] = (byte) 0x0;
    }
    else if(type == 1) { // 중간
        m_sHeader.fapp_type[0] = (byte) 0x0;
        m_sHeader.fapp_type[1] = (byte) 0x1;
    }
    else if(type == 2) { // 끝
        m_sHeader.fapp_type[0] = (byte) 0x0;
        m_sHeader.fapp_type[1] = (byte) 0x2;
    }
}
```

현재 파일 fragment의 타입을 지정해 줍니다. 0이라면 첫번째 frag 1이라면 중간의 frag, 3이라면 마지막 frag입니다.

```
public int calcSeqNum(byte[] input) { // 몇 번째 Frame인지 계산 (Frame은 0번부터 시작)
    int seqNum = 0;
    seqNum += (input[8] & 0xff) << 24;
    seqNum += (input[9] & 0xff) << 16;
    seqNum += (input[10] & 0xff) << 8;
    seqNum += (input[11] & 0xff);

    return seqNum;
}
```

파일이 frag를 몇번 해주어야 하는지 계산하는 메소드 입니다.

```
public boolean fileInfoSend(byte[] input, int length) { // 파일 정보 송신 함수
    this.setFileMsgType(0); // 파일 정보 송신임을 나타냄
    this.Send(input, length); // 파일 정보 송신

    return true;
}
```

파일의 정보만을 보내주는 함수입니다.

```
// 프레임을 다 받았는지 확인 후, 모두 정확히 수신했으면 정렬을 진행하는 함수
public boolean sortFileList(int lastFrameNumber) {
    // 모든 프레임을 받았는지 확인
    if((fileByteList.size() - 1 != lastFrameNumber) || (receivedLength != targetLength))
        ((ChatFileDlg)this.GetUpperLayer(0)).ChattingArea.append("파일 수신 실패\n");
        return false;
    }

    // ArrayList에 SeqNum을 Index로 가지도록 삽입하여 정렬 진행
    fileSortList = new ArrayList();
    for(int checkSeqNum = 0; checkSeqNum < (lastFrameNumber + 1); ++checkSeqNum) {
        byte[] checkByteArray = fileByteList.remove(0);
        int arraySeqNum = this.calcSeqNum(checkByteArray);
        fileSortList.add(arraySeqNum, checkByteArray);
    }

    return true;
}
```

먼저 파일을 모두 받았는지 확인해 주고 그 이후로 seqnum을 통해 배열에 정렬해서 넣어주는 역할을 해 줍니다.

```
public void setAndStartSendFile() {
    ChatFileDlg upperLayer = (ChatFileDlg) this.GetUpperLayer(0);
    File sendFile = upperLayer.getFile();
    int sendTotalLength; // 보내야하는 총 크기
    int sendedLength; // 현재 보낸 크기
    this.resetSeqNum();
```

ChatFileDlg에 존재하는 getFile 함수를 통해서 파일을 받아오고 send할 준비를 해줍니다.

```
try (FileInputStream fileInputStream = new FileInputStream(sendFile)) {
    sendedLength = 0;
    BufferedInputStream fileReader = new BufferedInputStream(fileInputStream);
    sendTotalLength = (int)sendFile.length();
    this.setFileSize(sendTotalLength);
    byte[] sendData =new byte[1448];
    ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setMaximum(sendTotalLength);
    if(sendTotalLength <= 1448) {
        // 파일 정보 송신
        setFragmentation(0);
        this.setFileMsgType(0);
        this.fileInfoSend(sendFile.getName().getBytes(), sendFile.getName().getBytes().length
        // 파일 데이터 송신
        this.setFileMsgType(1);
        fileReader.read(sendData);
        this.Send(sendData, sendData.length);
        sendedLength += sendData.length;
        ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setValue(sendedLength);
```

먼저 파일을 받아온 다음 progressbar의 크기를 파일의 총 크기에 비례해서 Maximumnum을 설정해 줍니다. 그 이후 파일의 크기 1448 이하라면 frag가 없이 그냥 진행됩니다. 때문에 파일을 바로 송신해 주면 됩니다. 파일의 작업을 진행할 때마다 progressbar의 value를 수정해 줍니다.

FileMsgType(1)은 파일 데이터입니다. FileMsgType(0)은 파일의 정보입니다.

```
} else {
    sendedLength = 0;
    // 파일 정보 송신
    this.setFragmentation(0);
    this.setFileMsgType(0);
    this.fileInfoSend(sendFile.getName().getBytes(), sendFile.getName().getBytes().length);

    // 파일 데이터 송신
    this.setFileMsgType(1);
    this.setFragmentation(1);
    while(fileReader.read(sendData) != -1 && (sendedLength + 1448 < sendTotalLength)) {
        this.Send(sendData, 1448);
        try {
            Thread.sleep(4);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        sendedLength += 1448;
        this.increaseSeqNum();
        ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setValue(sendedLength);
    }

    byte[] getRealDataFrame = new byte[sendTotalLength - sendedLength];
    this.setFragmentation(2);
    fileReader.read(sendData);


            for(int index = 0; index < getRealDataFrame.length; ++index) {
                getRealDataFrame[index] = sendData[index];
            }

            this.Send(getRealDataFrame, getRealDataFrame.length);
            sendedLength += getRealDataFrame.length;
            count = 0;
            ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setValue(sendedLength);
        }
        fileInputStream.close();
        fileReader.close();
    } catch(IOException e) {
        e.printStackTrace();
    }
}
```

그 이후는 파일이 1448의 크기를 넘어 frag가 필요할 때의 함수 입니다.

첫번째 frag에게 파일의 정보를 넣어서 송신합니다. 그 이후 두번째 frag부터 (setFragmentation(1) 부분) +1448 이 전체 길이를 넘게 될 때까지 중간 frag입니다. Thread.sleep을 통하여 동기화를 진행해 줍니다. 이후 마지막은

frag 3입니다.

```
public synchronized boolean Receive(byte[] input) { // 데이터를 수신 처리 함수
    byte[] data;

    if(checkReceiveFileInfo(input)) { // 파일의 정보를 받은 경우
        data = RemoveCappHeader(input, input.length); // Header없애기
        String fileName = new String(data);
        fileName = fileName.trim();
        targetLength = calcFileFullLength(input); // 받아야 하는 총 크기 초기화
        file = new File("./" + fileName); //받는 경로..

        // Progressbar 초기화
        ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setMinimum(0);
        ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setMaximum(targetLength);
        ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setValue(0);

        // 받은 크기) 초기화
        receivedLength = 0;
```

데이터를 받았을 때 실행되는 함수입니다.

```
public boolean checkReceiveFileInfo(byte[] input) {
    if(input[6] == (byte)0x00)
        return true;

    return false;
}
```

파일의 정보를 받았다면 즉 FileMsgType(0) 이라면 받아야하는 총 크기를 받아주고 받는경로와 progressbar를 초기화 해줍니다.

```
    receivedLength = 0;
} else {
    // 단편화를 하지 않은 데이터를 받은 경우
    if (checkNoFragmentation(input)) {
        data = RemoveCappHeader(input, input.length);
        fileByteList.add(this.calcSeqNum(input), data);
        try(FileOutputStream fileOutputStream = new FileOutputStream(file)) {
            fileOutputStream.write(fileByteList.get(0));
        } catch (IOException e) {
            e.printStackTrace();
        }
```

FileMsgType(0)이 아니라면 파일의 데이터를 받아온 경우입니다. 먼저 단편화를 하지 않은 데이터의 크기가 1448 미만이라면 헤더를 제거해주고 fileByteList에 add해주고 이후 fileOutputStream에 write해줍니다.

```
    ,
} else {
    // 단편화를 진행한 데이터를 받은 경우

    // 데이터 프레임 수신
    fileByteList.add(input);
    receivedLength += (input.length - 12); // 헤더의 길이는 제외

    // 마지막 프레임 수신
    if(checkLastDataFrame(input)) {
        int lastFrameNumber = this.calcSeqNum(input);

        if(sortFileList(lastFrameNumber)) {
            try(FileOutputStream fileOutputStream = new FileOutputStream(file)) {
                for (int frameCount = 0; frameCount < (lastFrameNumber + 1); ++frameCount) {
                    data = RemoveCappHeader(fileSortList.get(frameCount), fileSortList.get(frameCount).length);
                    fileOutputStream.write(data);
                }
                ((ChatFileDlg)this.GetUpperLayer(0)).ChattingArea.append("파일 수신 및 생성 완료\n");
                fileByteList = new ArrayList();
            } catch (FileNotFoundException e) {
                ((ChatFileDlg)this.GetUpperLayer(0)).ChattingArea.append("파일 수신 실패\n");
                e.printStackTrace();
            } catch (IOException e) {
                ((ChatFileDlg)this.GetUpperLayer(0)).ChattingArea.append("파일 수신 실패\n");
                e.printStackTrace();
            }
        }
    }
    ((ChatFileDlg)this.GetUpperLayer(0)).progressBar.setValue(receivedLength); // Progressbar 갱신
}
```

파일의 크기가 1448 이상이라면 frag가 진행된 데이터 입니다.

마지막 프레임까지 전부 받았다면 sortFileList를 통해서 파일들을 리스트에 순서대로 넣어 저장해주어야 합니다.

```
public boolean Send(byte[] input, int length) { // 데이터 송신 함수
    byte[] bytes = this.ObjToByte(m_sHeader, input, length);
    ((EthernetLayer)this.GetUnderLayer()).fileSend(bytes, length + 12);
    return true;
}
```

ehternetLayer에 있는 fileSend 함수를 통해 파일을 송신해 줍니다.

```
private class _ETHERNET_Frame {
    _ETHERNET_ADDR enet_dstaddr;
    _ETHERNET_ADDR enet_srcaddr;
    byte[] enet_type;
    byte[] enet_data;

    public _ETHERNET_Frame() {
        this.enet_dstaddr = new _ETHERNET_ADDR();
        this.enet_srcaddr = new _ETHERNET_ADDR();
        this.enet_type = new byte[2];
        this.enet_data = null;
    }
}
```

Ehternet Layer의 헤더는 stop & wait과 같습니다.

```
public boolean Send(byte[] input, int length) {
    m_sHeader.enet_type = intToByte2(0x2080);

    byte[] data = ObjToByte(m_sHeader, input, length);
    this.GetUnderLayer().Send(data, data.length);

    return true;
}

public boolean fileSend(byte[] input, int length){
    m_sHeader.enet_type = intToByte2(0x2090);

    byte[] data = ObjToByte(m_sHeader, input, length);
    this.GetUnderLayer().Send(data, data.length);

    return true;
}
```

데이터의 송신은 send와 filesend로 나뉩니다. 파일이라면 enet_type을 0x2080으로 수정해주고, file이라면 0x2090으로 수정해 주었습니다.

그 이후 헤더에 수신될 주소와 자신의 주소, 타입을 저장하고 하위 계층에 보내줍니다.

```
public synchronized boolean Receive(byte[] input) {
    byte[] data = {};
    boolean isok = false;

    // broadcast 인 경우
    if(isBroadcast(input)){
        isok = true;
    }
    // 목적지 주소가 자신인지
    else if(chkAddr(input)){
        isok = true;
    }

    if(isok){
        if(input[12]==(byte)0x20 && input[13]==(byte)0x80){ // 채팅
            data = RemoveEthernetHeader(input, input.length);
            this.GetUpperLayer(0).Receive(data);
            return true;
        }
        else if(input[12]==(byte)0x20 && input[13]==(byte)0x90){
            data = RemoveEthernetHeader(input, input.length);
            this.GetUpperLayer(1).Receive(data);
            return true;
        }
    }

    return false;
}
```

먼저 데이터를 수신한다면 브로드캐스트 주소인지, 목적지 주소가 자신의 주소인지 판별해주고 아니라면 false를 리턴해 주어야합니다. Isok변수를 통해 확인해 주었습니다. 그 이후에 파일의 타입을 검사해 줍니다. 0x2080이라면 chat이고 0x2090이라면 file입니다. Input[12]와 input[13]을 통해 확인해 주었습니다.

```
private class _CHAT_APP {
    byte[] capp_totlen;
    byte capp_type;
    byte capp_unused;
    byte[] capp_data;

    public _CHAT_APP() {
        this.capp_totlen = new byte[2];
        this.capp_type = 0x00;
        this.capp_unused = 0x00;
        this.capp_data = null;
    }
}
```

ChatAppLayer의 헤더는 stop & wait 과 같습니다.

```
public boolean Send(byte[] input, int length) {
    byte[] bytes;
    m_sHeader.capp_totlen = intToByte2(length);
    m_sHeader.capp_type = (byte) (0x00);

    if (length > 1456) {    // 배열이 1456보다 클경우
        fragSend(input, length);
    } else {
        m_sHeader.capp_type = (byte) (0x00);    // 단편화 아니라면 00
        bytes = this.objToByte(m_sHeader, input, length);
        this.GetUnderLayer().Send(bytes, length + 4);
    }

    return true;
}
```

송신은 메시지의 길이가 1456 초과라면 단편화를 하고, 아니라면 타입을 00으로 해준뒤에 send해 줍니다.

```
private void fragSend(byte[] input, int length) {
    byte[] bytes = new byte[1456];
    int i = 0;
    m_sHeader.capp_totlen = intToByte2(length);
    m_sHeader.capp_type = (byte) (0x01);

    // 첫번째 전송
    byte[] tempbyte1 = new byte[1456];

    System.arraycopy(input, 0, tempbyte1, 0, 1456);
    bytes = objToByte(m_sHeader, tempbyte1, 1456);
    this.GetUnderLayer().Send(bytes, bytes.length);
    int maxLen = length / 1456;

    for(i = 0; i<=maxLen; i++){
        if(i == maxLen){
            m_sHeader.capp_type = (byte) (0x03);

            byte[] tempbyte = new byte[length%1456];

            System.arraycopy(input, i*1456, tempbyte, 0, length % 1456);
            byte[] data = objToByte(m_sHeader, tempbyte, length % 1456);
            this.GetUnderLayer().Send(data, data.length);
        }
        else if (i != 0) {

            m_sHeader.capp_type = (byte) (0x02);
            byte[] tempbyte = new byte[1456];

            System.arraycopy(input, i*1456, tempbyte, 0, 1456);
            byte[] data = objToByte(m_sHeader, tempbyte, 1456);
            this.GetUnderLayer().Send(data, data.length);
        }
    }
}
```

단편화 과정입니다. 첫번째 단편화일 때 채팅 전체의 길이를 담아서 보내줍니다. 그 이후는 1456 단위로 끊어서 보내줍니다. 마지막 단편화의 타입은 03이고, 중간 단편화는 02입니다.

```
public static void main(String[] args) {
    m_LayerMgr.AddLayer(new NILayer("NI"));
    m_LayerMgr.AddLayer(new EthernetLayer("Ethernet"));
    m_LayerMgr.AddLayer(new ChatAppLayer("ChatApp"));
    m_LayerMgr.AddLayer(new FileAppLayer("File"));
    m_LayerMgr.AddLayer(new ChatFileDlg("GUI"));

    m_LayerMgr.ConnectLayers( " NI ( *Ethernet ( *ChatApp ( *GUI ) *File ( *GUI ) ) )" );
}
```

ChatFileDlg의 메인입니다. ChatApp과 같은 층에 file이 생겼으므로 하나 더 생성해 줍니다.

```
File_send_Button.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub
        String[] split = filePath.split("\\\\");

        ChattingArea.append("[SEND] : " + split[split.length-1] + " 전송 시작 \n");
        ((FileAppLayer)m_LayerMgr.GetLayer("File")).setAndStartSendFile();
    }

});
```

채팅창 생성과 비슷하게 만들어 준 뒤에 send를 눌러 파일을 전송할 때의 처리를 해줍니다.

이것 역시 채팅창의 그것과 비슷하게 만들었습니다.

FileAppLayer의 setAndStartSendFile 함수를 실행합니다.

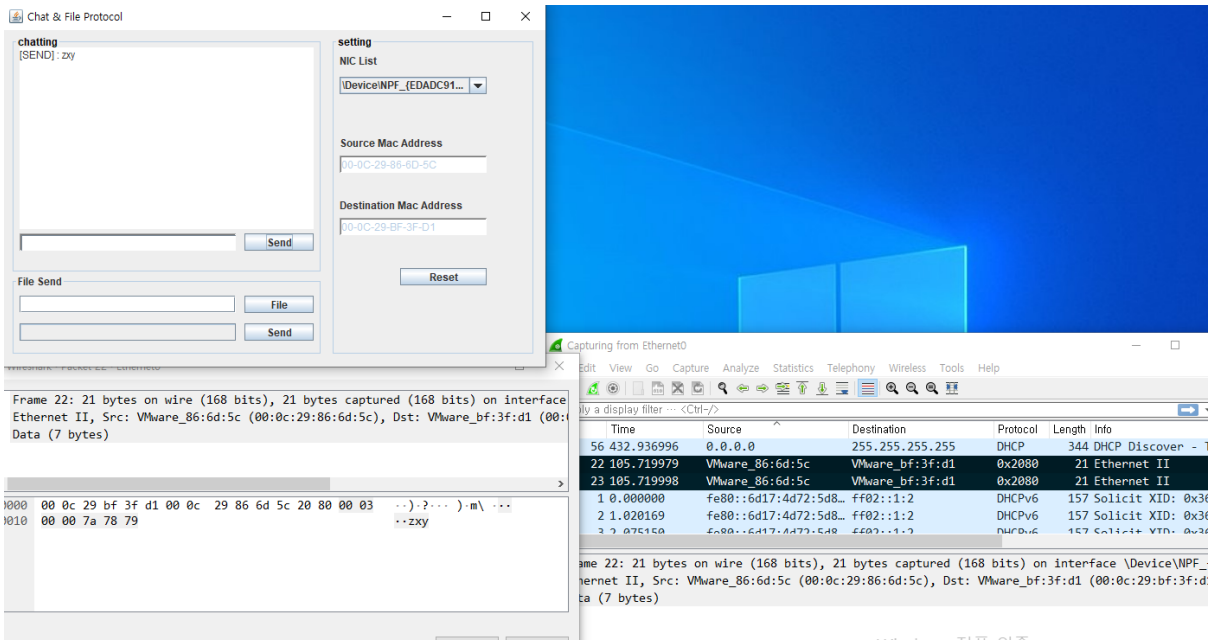
```
public File getFile() {
    return file;
}
```

getFile함수는 전역변수인 file을 리턴해 줍니다. 전역변수의 값을 준 곳은

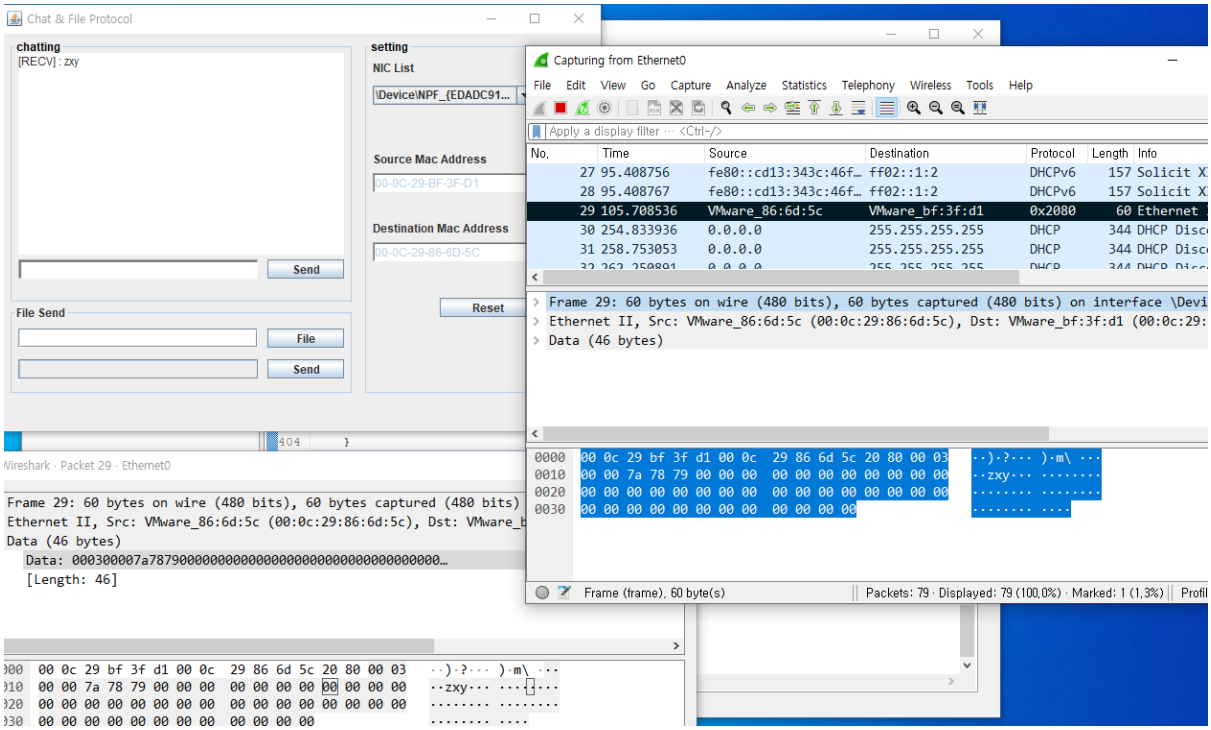
```
file_Choice_Button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int val = fileChooser.showOpenDialog(getParent());

        if(val == fileChooser.APPROVE_OPTION) {
            file = fileChooser.getSelectedFile();
            filePath = fileChooser.getSelectedFile().getAbsolutePath();
            filePathField.setText(filePath);
            System.out.println("directory :" + filePath);
        }
    }
});
```

파일 선택 창에서 파일을 선택했다면 그 파일을 받아서 file 전역변수에 넣어 줍니다.

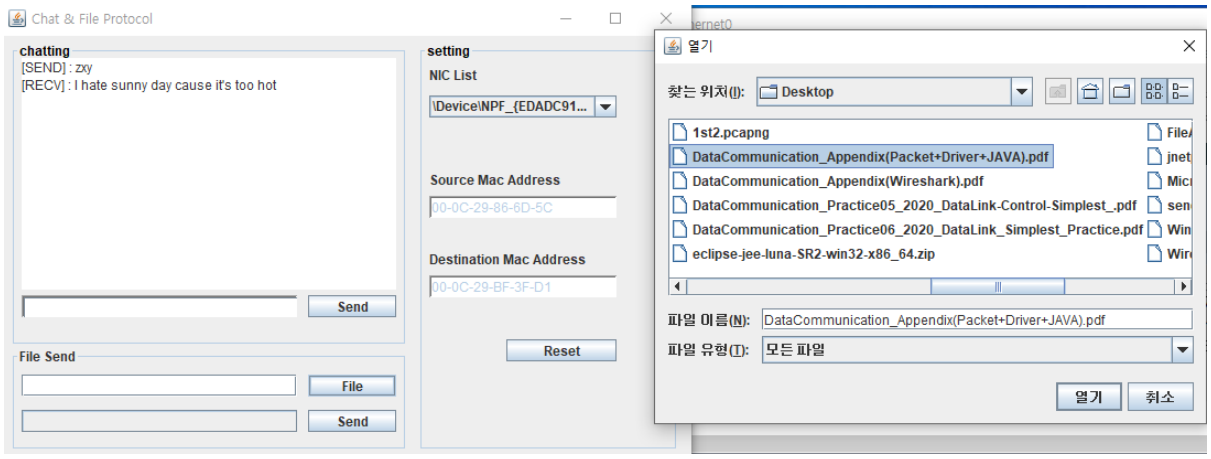
결과화면



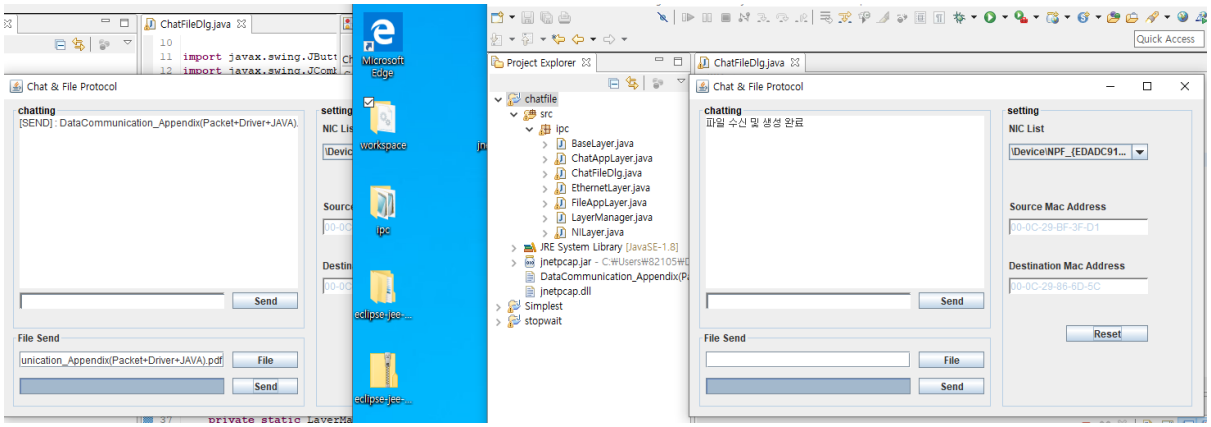
Zxy를 보내는 장면



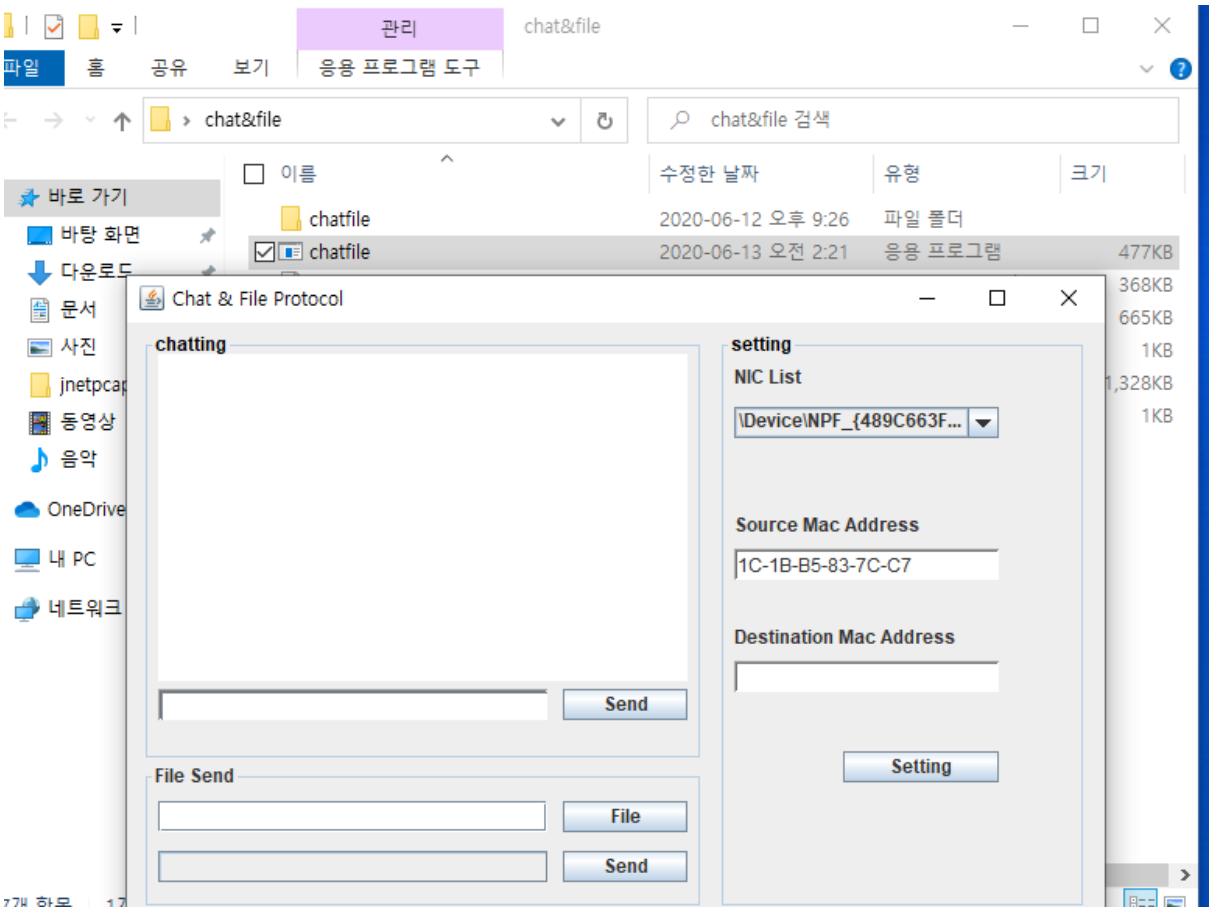
Zxy 수신



파일 선택 화면



파일 보내기 및 파일 수신 확인 오른쪽 화면에서 받은 파일이 보입니다.



exe파일 동작 확인