데이터 통신 Assignment 03

과제 제출일자 : 2019/04/09
학번 : 201502085, 이규봉
분반 : 00
과제 : Simplest 프로토콜

실습 개요, 목적

Simplest 프로토콜을 이용해, 랜선으로 서로 다른 두 컴퓨터의 두 프로세스를 연결해 통신하는 것이 목적. 송신 측 프로세스에서 String을 입력하면 수신 측 프로세스에서 화면에 표시하면 된다.

실습 시나리오

1) 프로그램을 실습할 두 대의 컴퓨터를 준비하고 랜선으로 연결함.
2) Wireshark로 이더넷 드라이버에서 송수신 되는 패킷을 확인해, 컴퓨터끼리 연결이 잘 되었는지를 확인함.
3) PC 1에서 NIC_ComboBox를 알맞게 설정하여 Source Mac Address를 셋팅하고, Destination Mac Address엔 상대 컴퓨터의 Source Mac Address를 직접 입력함.
4) PC 2에서도 똑같이 Mac Address를 설정한 후, 두 프로세스에서 Setting 버튼을 클릭해 연결
5) 채팅을 전송할 프로세스에서 채팅창에 짧은 길이의 String 데이터를 키보드로 입력하고, Send 버튼을 클릭함
6) SimplestDlg (GUI Layer)에서 아래 레이어의 Send를 호출하고 String 데이터를 byte[] 로 인코딩 해, 전송함. 아래 레이어는 그 아래 레이어로 데이터를 전송하는데 이 때 해당하는 레이어의 정보를 헤더로 만들어 붙임.
7) NILayer에서 jnetPcap을 이용해 랜선으로 데이터를 전송함.
8) 수신하는 쪽 프로세스의 NILayer에선 랜선으로 들어온 데이터는 모두 들어오며, 위 레이어의 Receive를 호출.
9) Ethernet Layer 쪽 Receive 함수에서, Ethernet Frame을 확인해 들어온 데이터의 Dst Address가 본인 프로세스가 갖고 있는 Source Address와 일치하는지, 또는 들어온 데이터가 브로드캐스팅인지 확인해, 일치하는 경우 필요 없어진 Ethernet 헤더를 제거하고, 위 쪽 레이어로 남은 데이터를 올려 보냄.
10) ChatAppLayer에서도 비슷한 과정이 일어나며, 다른 레이어들을 추가로 넣는다면 그 레이어들에서도 같은 과정이 일어날 것.
11) 결과적으로 최상위 레이어 (GUI) 측에선, 메시지를 송신한 프로세스 측이 보낸 데이터를 그대로 전달받게 됨. byte[]을 다시 String으로 인코딩한 후, Chatting Area 쪽에 append 해 수신한 메시지를 표시함.

프로토콜 스택의 구조

NILayer가 이번 과제 소프트웨어 계층에서 제일 밑에 있게 되며, 위에 Ethernet 레이어, ChatApp 레이어, SimplestDlg 레이어 순으로 스택이 위치하게 됩니다.

각 프로토콜 (layer) 의 역할

NILayer는 Receive_Thread를 돌리며 들어오는 패킷이 있는지를 계속 검사합니다. 들어오는 패킷이 있다면 모두 EthernetLayer로 올려 보냅니다. EthernetLayer의 Receive에선 패킷이 갖고 있는 Source Address와 Destination Address가 EthernetLayer가 갖고 있는 Destination Address와 Mac Address와 같은지를 검사하여, 참인 경우 Mac Address에 해당하는 바이트들을 제거한 후, 위로 올려보냅니다. Receive는 그 외에도 패킷의 목적지가 ff-ff-ff-ff-ff-ff 이고, 출발지가 상대 컴퓨터인 경우에도 (브로드캐스팅) 위 레이어의 Receive를 호출합니다. ChatAppLayer에선 RemoveCappHeader를 호출해 헤더를 제거한 후, 위 레이어로 데이터를 올려보냅니다.

구현 설명

※ NILayer에 Thread를 사용하는 이유

프로그램의 Setting 버튼이 눌린 이후 NILayer에선 해당 랜 드라이버에서 계속 패킷을 받아와야 합니다. 패킷에는 해당 채팅 프로그램이 이 Mac 주소로 보낸 패킷도 있을 수 있고, 브로드캐스팅 (ff-ff-ff-ff-ff-ff) 한 패킷이나, 아예 다른 패킷도 있을 수 있습니다. 그런데 Thread를 사용하지 않고 채팅 프로그램의 Main thread에서 패킷을 계속 받고 있으면 무한 루프에 빠져 GUI에서 Setting 버튼 이벤트 처리, Reset 버튼 이벤트 처리, Close 이벤트 처리 등 다른 작업을 처리할 수 없게 됩니다. 따라서, Runnable을 구현한 Receive_Thread 클래스를 정의해 새 쓰레드에서 무한 루프를 돌리며 패킷이 생길 때 마다 계속해서 가져오게 한 것 입니다.

※ NILayer, EthernetLayer가 Mac 주소를 파싱해 얻어오는 과정

SimplestDlg의 두 Mac address 창에 두 컴퓨터의 Mac Address를 적습니다. 이 때, Source Address는 NIC ComboBox를 통해 드롭박스내에서 선택할 수 있습니다. 입력된 Mac Address는 이더넷 레이어가 갖고 있는 _ETHERNET_Frame 객체의 출발지, 목적지에 파싱되어 들어갑니다. 이 때, byte의 범위는 -128 ~ 127이고 Mac 주소 한 토큰의 값의 범위는 0 ~ 255 이므로 인코딩이 필요합니다. 그래서 저는 String을 일단 byte로 캐스팅하고 예외가 던져질 경우 받아서 256을 빼는 방식으로 처리했습니다. NILayer의 경우, 코드를 약간 더 깔끔하게 작성하기 위해 SetAdapterNumber의 인자를 String으로 바꿨습니다. 인자로 들어온 Mac Address (프로세스를 실행한 컴퓨터의 Mac Address가 되어야 합니다.) 와 같은 맥 주소를 갖는 m_pAdapterList 내 객체가 있는지 확인하고, 있는 경우 그 찾은 인덱스 값을 m_iNumAdapter에 대입해 넣습니다. 이렇게 두 레이어는 Mac 주소를 얻어와 두 컴퓨터를 식별하는데 활용합니다.

※ EthernetLayer의 Receive 과정

NILayer는 사실 상 이더넷으로 지나가는 모든 패킷을 수신하기 때문에 EthernetLayer에서 원하지 않는 패킷들을 걸러내야 합니다. 이렇게 하기 위해, Addressing이란 메서드를 만들어, 데이터를 보내는 과정에서 0바이트부터 11바이트까지를 각각 데이터를 수신하는 컴퓨터의 Mac 주소, 송신하는 컴퓨터의 Mac 주소로 하며, 12바이트와 13바이트를 타입 값으로 채웁니다. 이 때 타입 값은 두 컴퓨터의 채팅 프로세스에서 같은 값을 사용해야 하며, 다른 패킷들의 프로토콜에서 사용하는 값이 아니여야 합니다. (저는 임의로 0x03, 0x00를 사용해 실습했습니다.) 이렇게 Receive에선 패킷이 갖고 있는 타입 byte와 목적지 프로그램의 헤더 객체가 갖고 있는 타입 byte가 같은지 검사하고, 다르다면 무조건 false를 리턴함으로써, 더미 패킷들을 걸러냅니다. 그리고 패킷이 갖고 있는 목적지의 비트 패턴이 모두 ff이고 패킷이 갖고 있는 출발지와 헤더 객체가 갖는 목적지의 byte 값이 모두 같으면서, 그 값이 출발지 byte값과는 다를 때, 이 경우를 브로드캐스팅으로 수신합니다. 그리고 패킷이 갖고 있는 목적지 byte들과 헤더 객체의 출발지 byte 값들이 서로 갖고 패킷이 갖는 출발지 byte와 헤더 객체의 목적지 byte들이 같을 때 1:1 채팅으로 데이터를 수신하게 됩니다.

※ SimplestDlg에서 컴퓨터의 모든 Mac 주소를 가져오는 과정

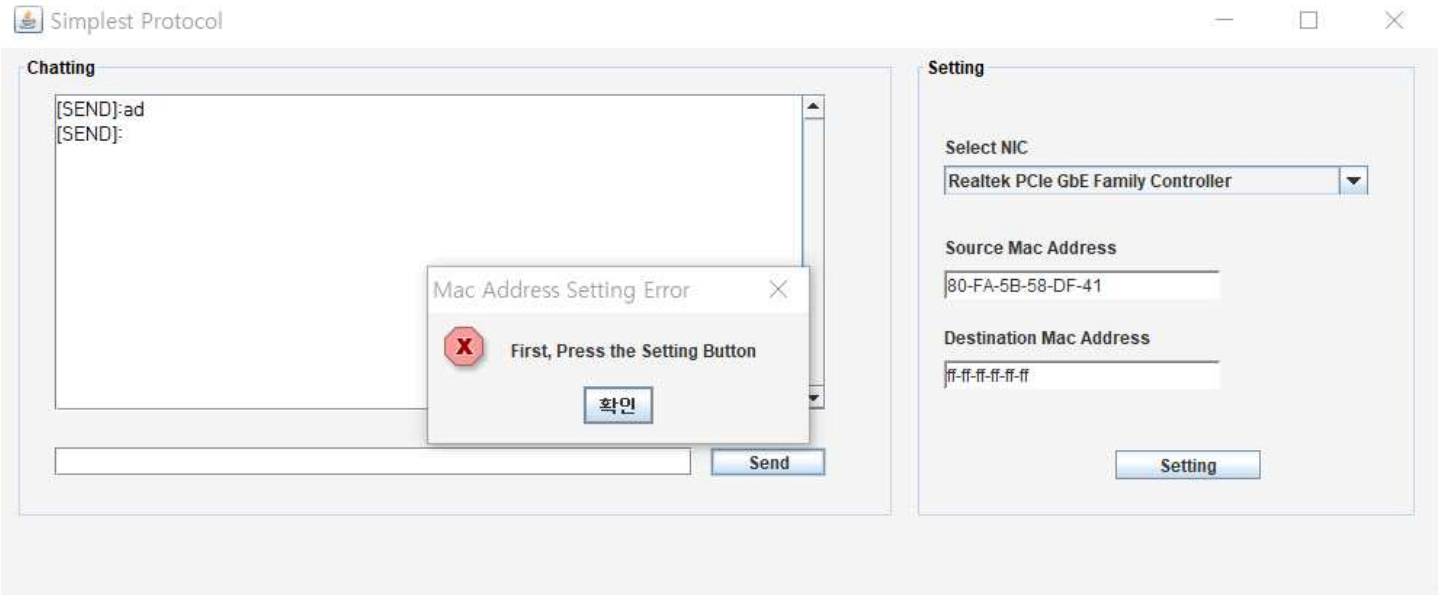
SimplestDlg 클래스는 생성될 때, NICComboBox를 getAllNetworkName의 리턴값을 인자로 받아 생성합니다. getAllNetworkName 메서드는 NetworkInterface.getNetworkInterfaces 메서드를 사용하여 컴퓨터에 연결되어 있는 모든 NetworkInterface들의 열거형을 받아옵니다. 이 열거형 값을 하나하나 돌며, getHardwareAddress 메서드를 사용해 Mac 주소를 가져온 후, 인코딩 해 얻은 Mac 주소들을 모두 ArrayList 객체에 저장해 놓습니다. 또한, 네트워크들의 이름을 getDisplayName 메서드를 활용해 String[] add 해, 열거형에 더 남은 객체가 없을 때 리턴합니다.

※ SimplestDlg에서의 에러처리

SimplestDlg는 다음과 같은 에러처리를 하도록 구현했습니다.

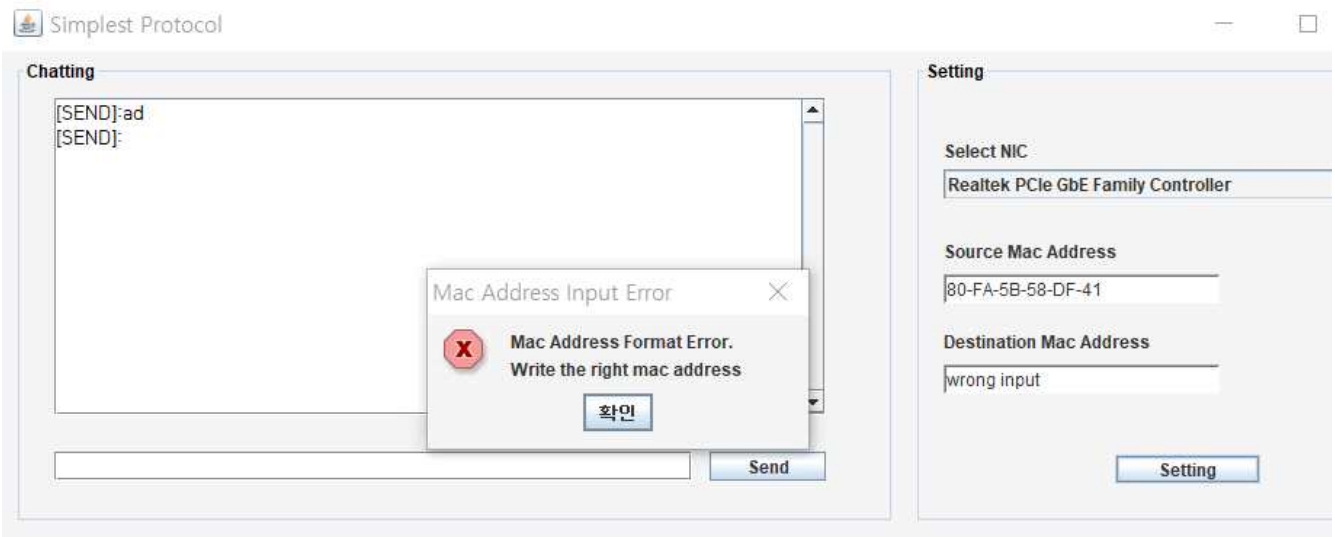
1) jnetpcap.jar, jnetpcap.dll이 실행프로그램과 같은 경로에서 발견되지 않거나, winpcap이 제대로 설치되지 않은 경우 => 사용자가 exe 파일을 jnetpcap.jar, jnetpcap.dll 파일 없이 실행하려 할 때, 프로그램이 제대로 실행되지 않았다는 것을 명확하게 전달하기 위해 구현했습니다.

2) Setting 버튼을 눌러 Mac address를 입력하기 전에 Send 버튼을 눌러 메시지를 전송하려 한 경우



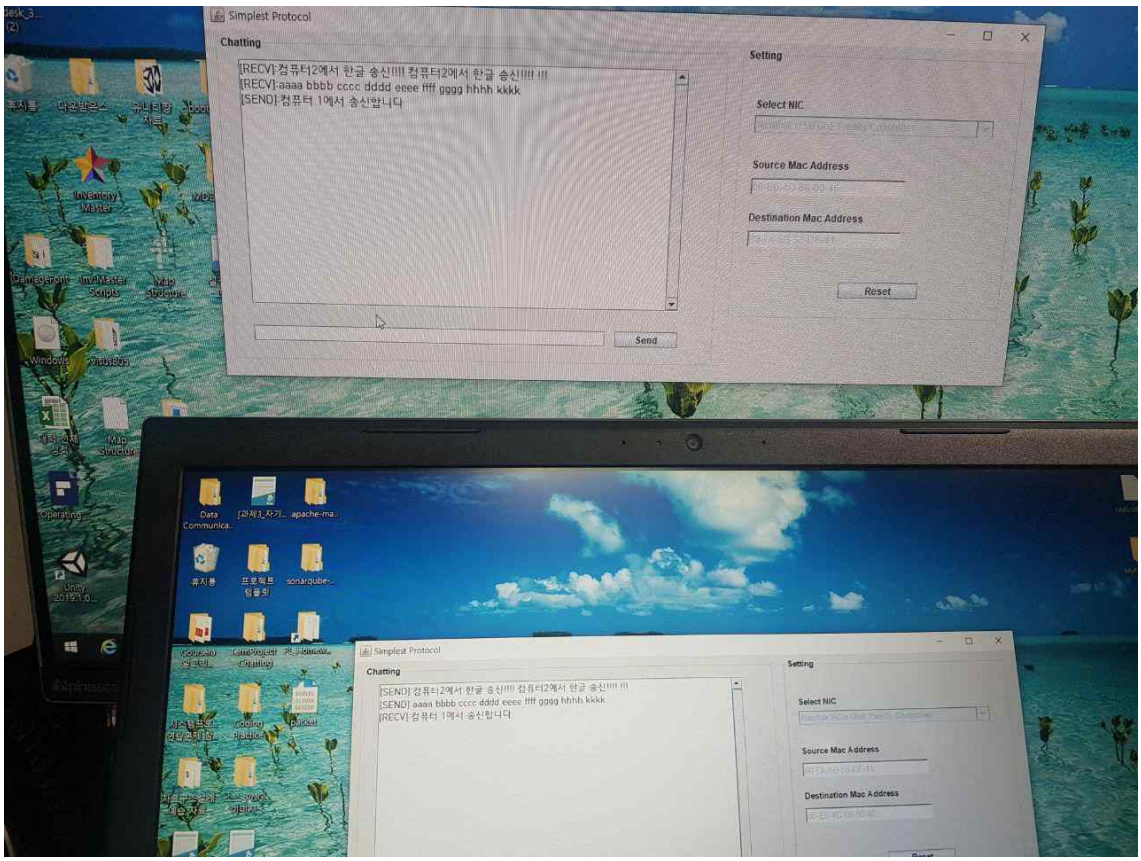
3) 올바르지 않은 Mac 주소 형식 입력

=> 목적지의 Mac 주소는 사용자가 직접 입력해야 하므로, 잘못된 값을 설정할 수 있으므로 구현했습니다. 구현에는 java.util.regex.* 패키지와 1개의 정규식을 사용했습니다.

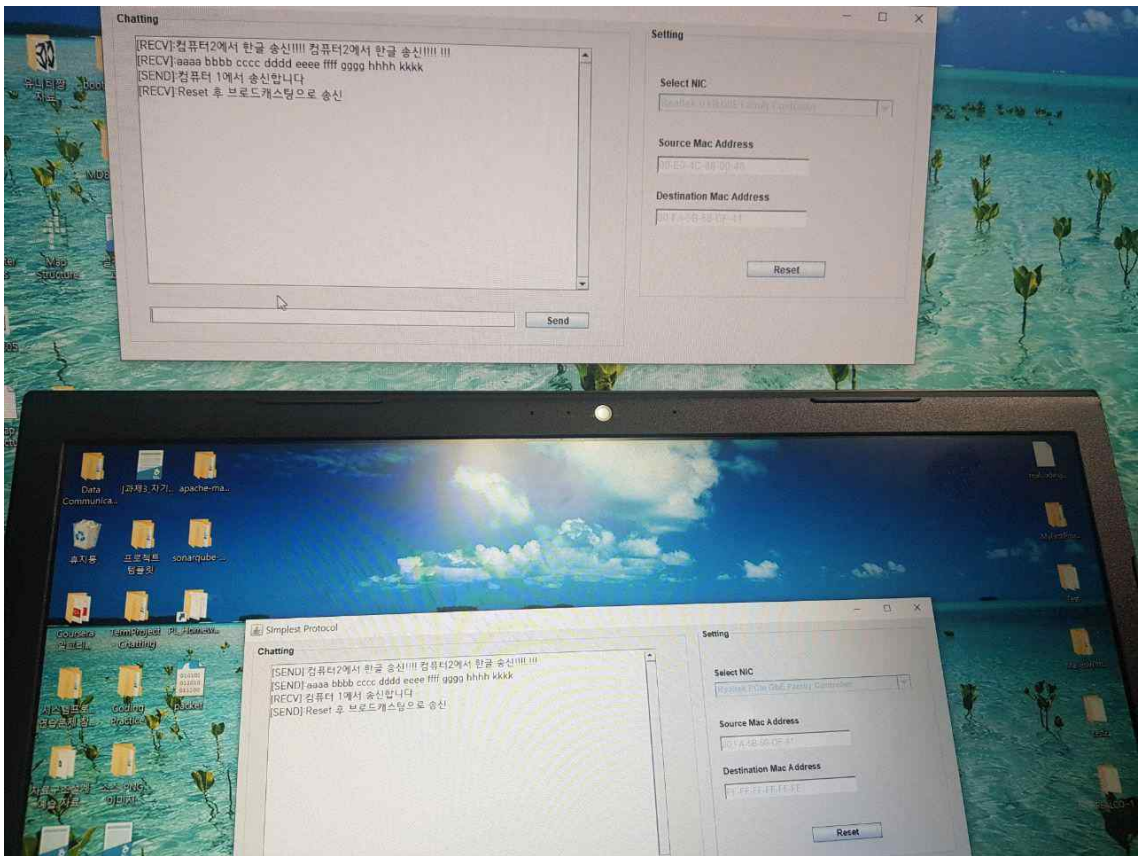


실행 스크린샷 및 사진

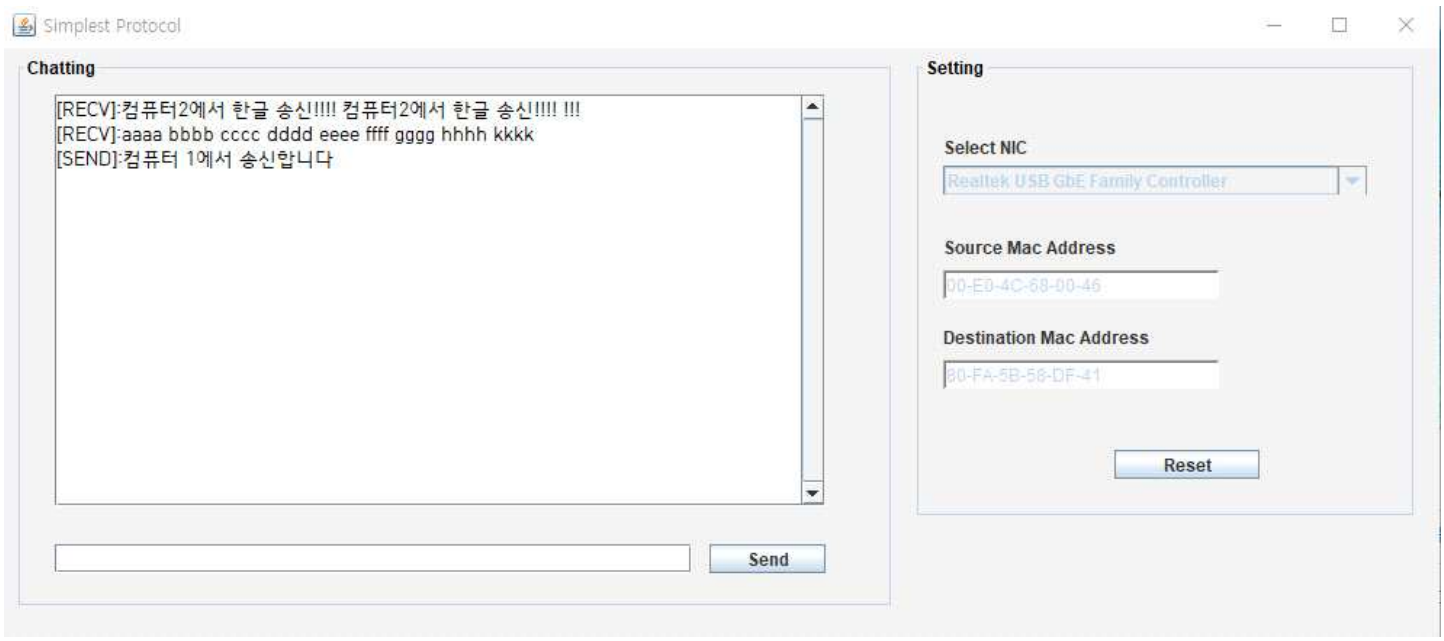
[두 컴퓨터 간에 통신]



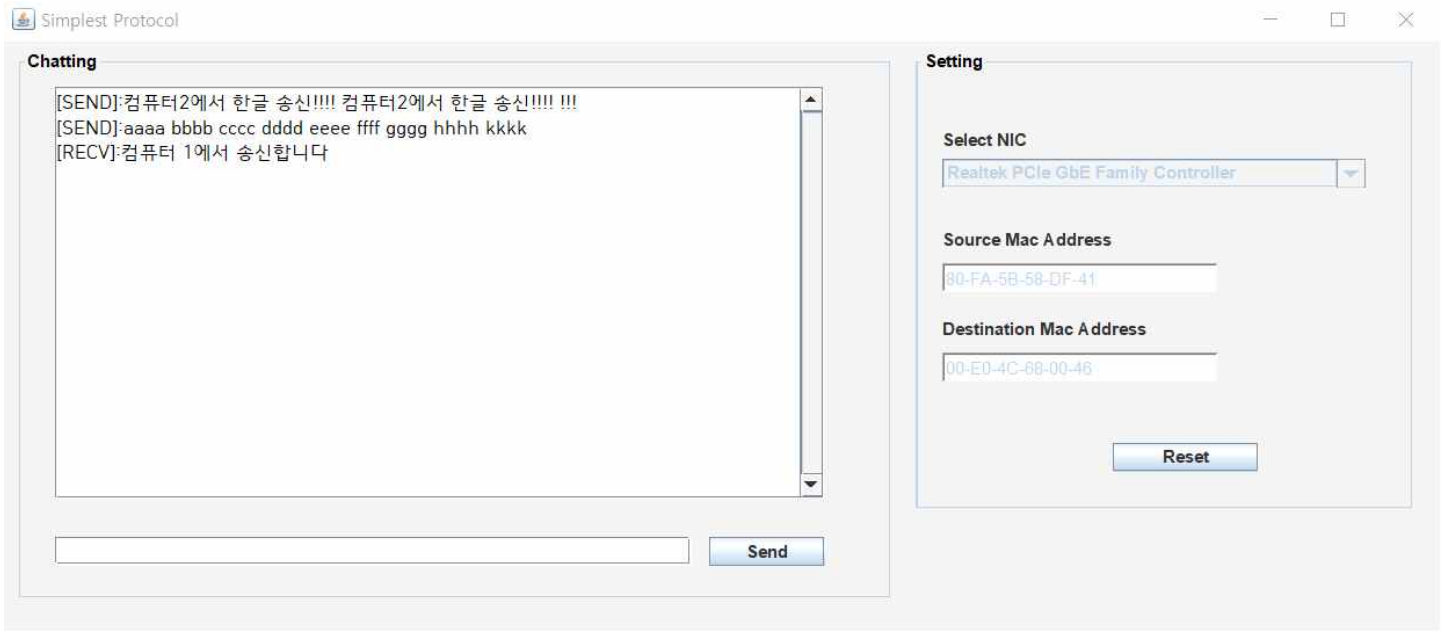
[브로드캐스팅]



[컴퓨터1 (00-E0-4C-68-00-46) 의 화면]



[컴퓨터2 (80-FA-5B-58-DF-41) 의 화면]



참고 웹페이지, 자료 URL

https://jeong-pro.tistory.com/155
- wireshark 설치

https://myeonguni.tistory.com/1593
- 자바에서 패킷 캡쳐를 위한 jNetPcap 개발 환경 구축하기

http://jnetpcap.sourceforge.net/docs/jnetpcap-1.0b3-javadoc/org/jnetpcap/Pcap.html
- Pcap API

https://myeonguni.tistory.com/1593
- dll 파일 정적 로드로 임포트

https://binshuuuu.tistory.com/60
- 이클립스에 Winbuilder 설치.

https://codippa.com/how-to-get-all-mac-addresses-of-a-system-from-java-program/
- **자바에서 모든 mac 주소 얻어오는 법.**

https://hiseon.me/2018/03/20/libpcap-tutorial/
- 주석 작성할 때 참고

https://m.blog.naver.com/PostView.nhn?blogId=jsky10503&logNo=221230275532&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

- 주석 작성할 때 참고

http://blog.naver.com/PostView.nhn?blogId=kjhfreedom21&logNo=110085820033
- JScrollPane을 이용한 스크롤바 추가

http://blog.naver.com/PostView.nhn?blogId=wisthood00&logNo=140163625279
- JTextarea 글자 수 제한

https://stackoverflow.com/questions/30679878/how-to-detect-backspace-in-a-keytypedevent/30679971
- backspace 입력 탐지