데이터 통신 00분반

201402447 한원희

**실습 개요**

실습 목적

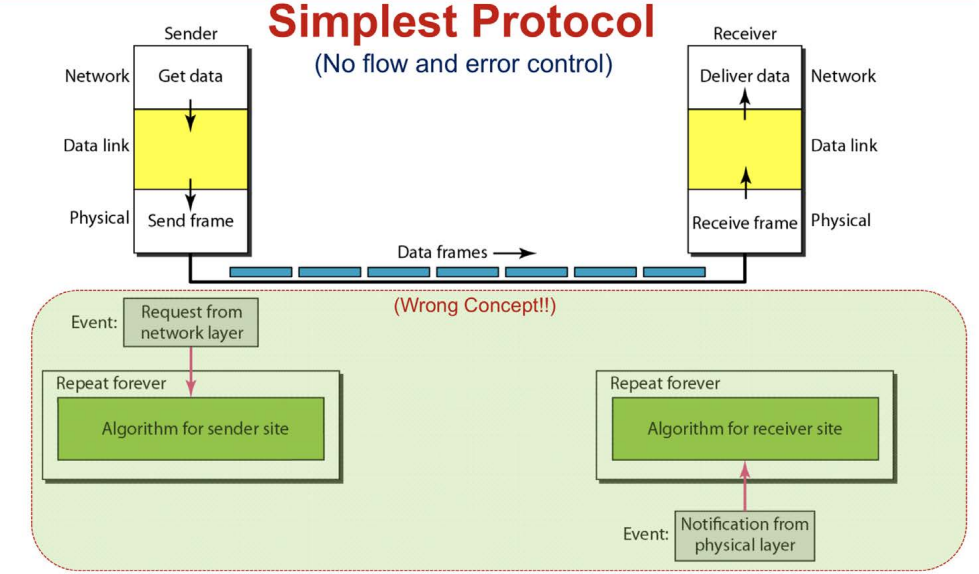
* 이더넷 프로토콜을 이용한 호스트 간의 데이터 정보 교환이 어떤식으로 일어나는지 알아보는 실습입니다. 그 중 흐름이나 에러가 없는 control인 simplest 프로토콜을 구현합니다.

실습 시나리오

* VM을 이용해 두개의 윈도우로 프로그램을 실행시켜 줍니다. 각 윈도우에 이클립스, 자바 등 환경 설정을 해줍니다. 두개의 윈도우는 LAN segment로 연결되어 있습니다. 코드를 수정해 주고 두개의 PC에서 실행시킨 뒤 샤크 프로그램을 통해 잘 전달 되었는지 확인해줍니다.

**프로토콜 스택**

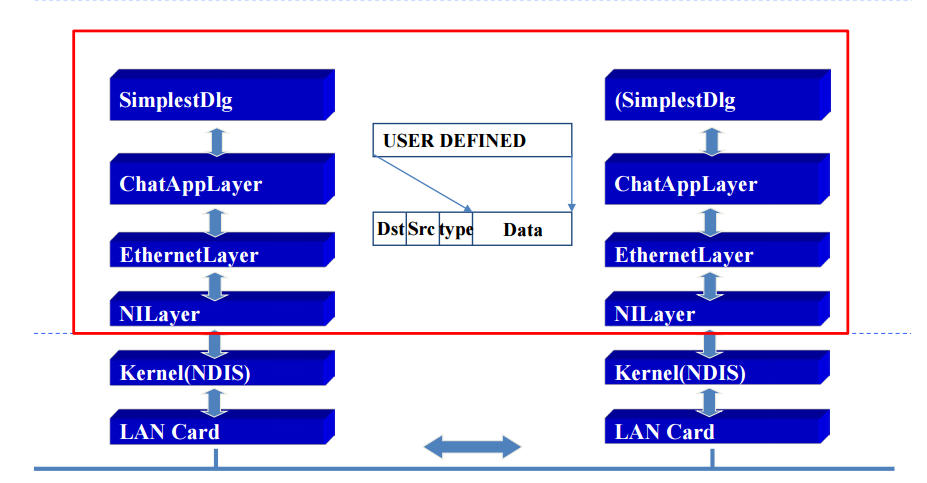
구조 설명



Simplest Protocol은 흐름이나 오류 제어가 없는 단순한 메시지 전송 프로토콜입니다.

하나의 호스트에서 상대방으로 데이터 링크 계층을 사용해 데이터가 전송됩니다.

각 프로토콜의 역할



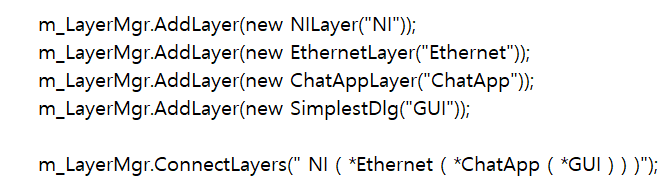
SimplestDlg 에서는 각 레이어를 순서대로 연결하고 사용자 GUI를 사용해 상대방의 입출력 메시지를 확인해 줍니다.

ChatAppLayer는 단순 dialog에서 입력 데이터에 헤더를 붙여서 이더넷 레이어에 전송해 줍니다. 수신시에는 헤더를 제거해 줍니다.

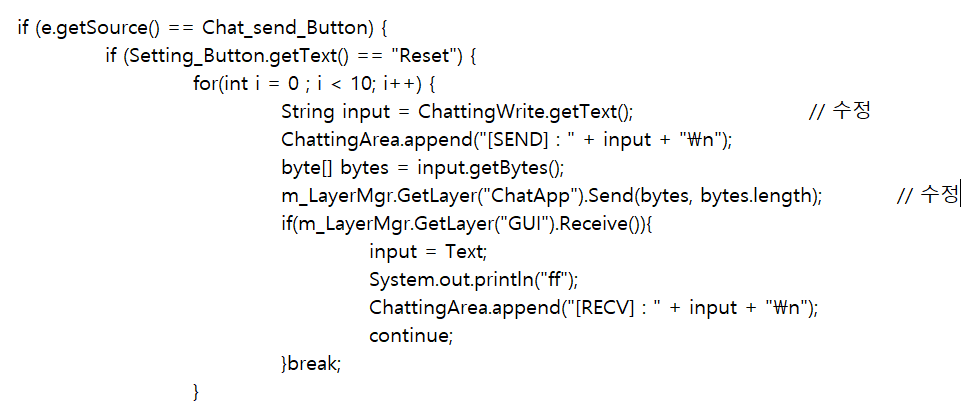
EthernetLayer는 쓰레드를 통해 수신자가 있으면 지속적으로 패킷을 받을 수 있도록 설정해 줍니다.

NlLayer는 mac주소를 가지고 프레임을 주고받는 역할을 합니다.

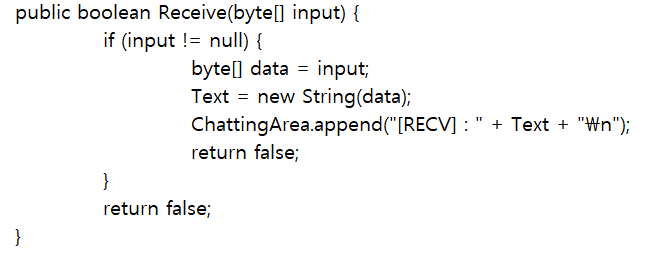
**구현설명**



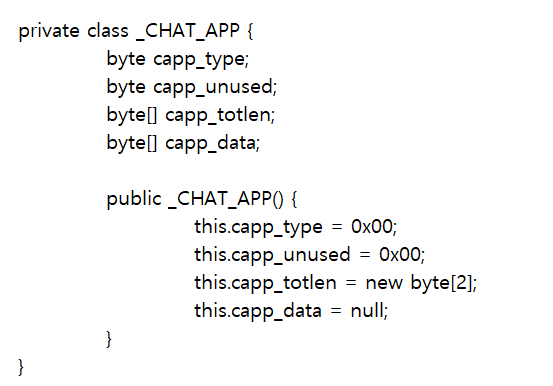
SimplestDlg에 main입니다. 각각의 Layer를 연결해주는 코드입니다.



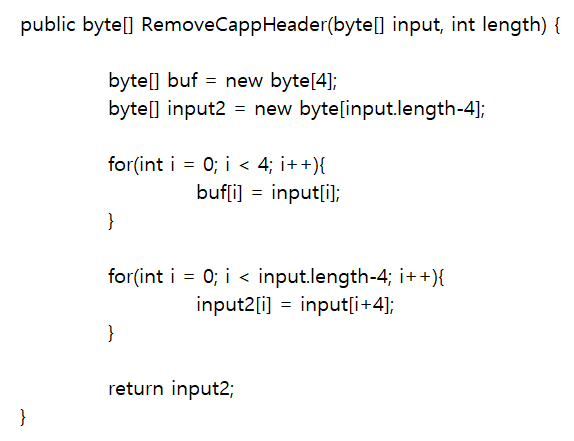
Send 버튼이 눌렸을 경우엔 채팅창에 표시해주고 byte로 바꾸어 주어 아래 Layer인 ChatApp으로 넘겨줍니다. 채팅창에 써준 text를 표시해 주기 위해 기존 코드에서 ChattinWrite.getText() 메소드를 사용했고 또한 정보를 전달하는 Layer를 Ethernet에서 ChatApp으로 수정하였습니다.



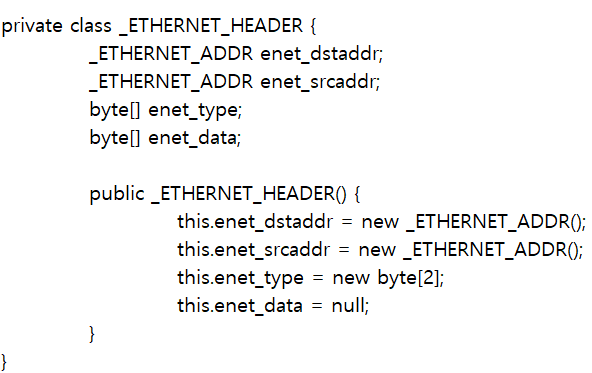
Receive 메소드입니다. 받은 data를 string으로 바꾸어 채팅창에 출력해줍니다.



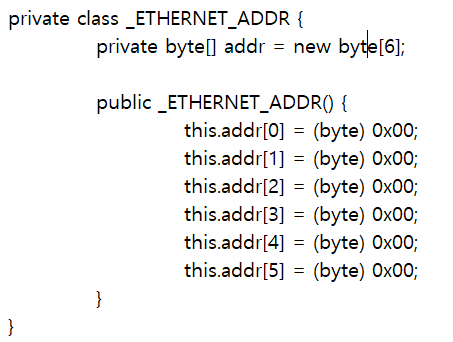
ChatAppLayer의 헤더 부분입니다. Totlen, type, unused 총 4바이트를 사용합니다.



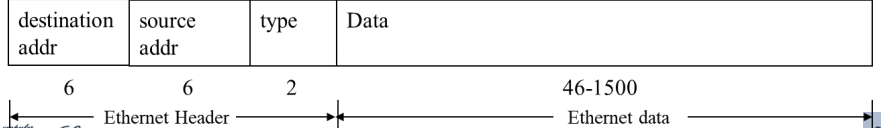
받은 CappHeader를 벗겨주는 메소드입니다. CappHeader를 벗긴 뒤에 Data부분을 input2에 넣어 return 해 줍니다.



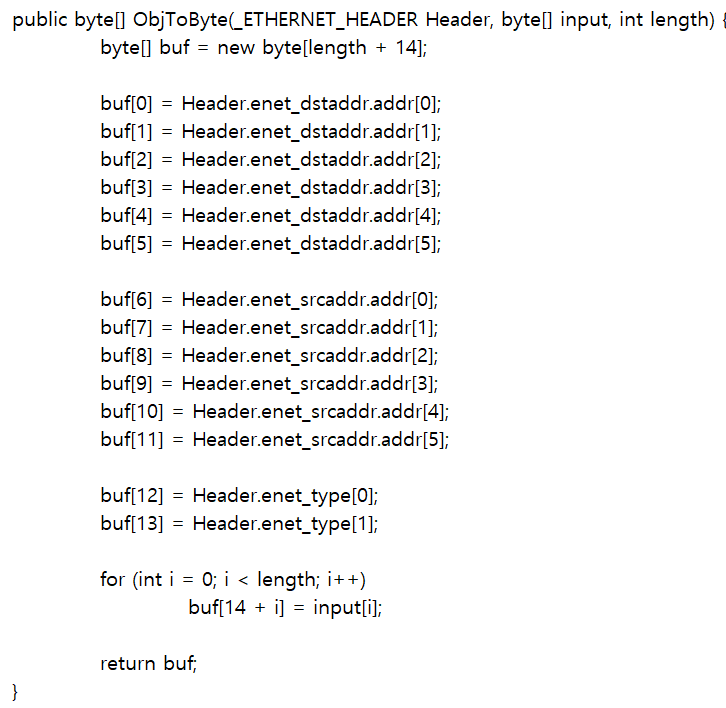
Ethernet 입니다. 목적 주소와 소스 주소, type과 data를 담아둘 헤더를 만듭니다.



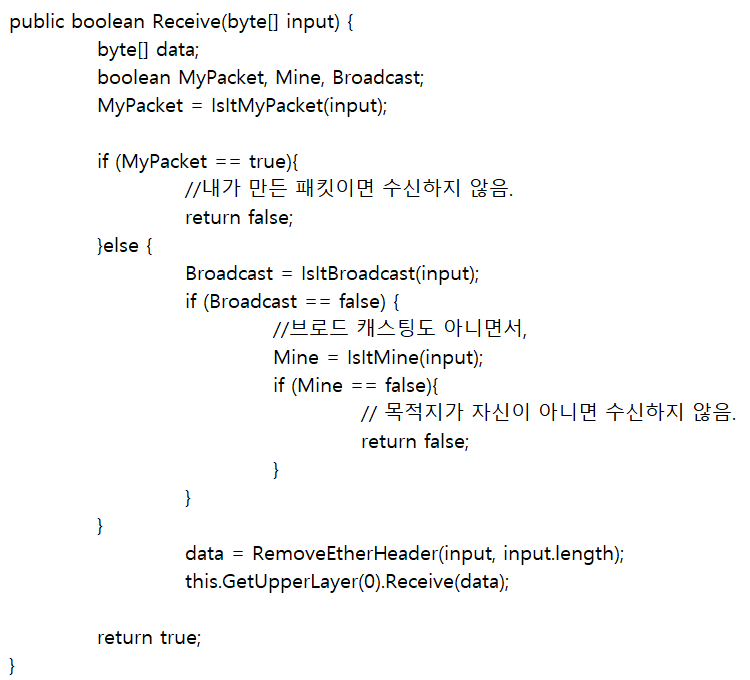
주소의 크기는 6바이트의 배열입니다.



전체 헤더의 크기입니다.

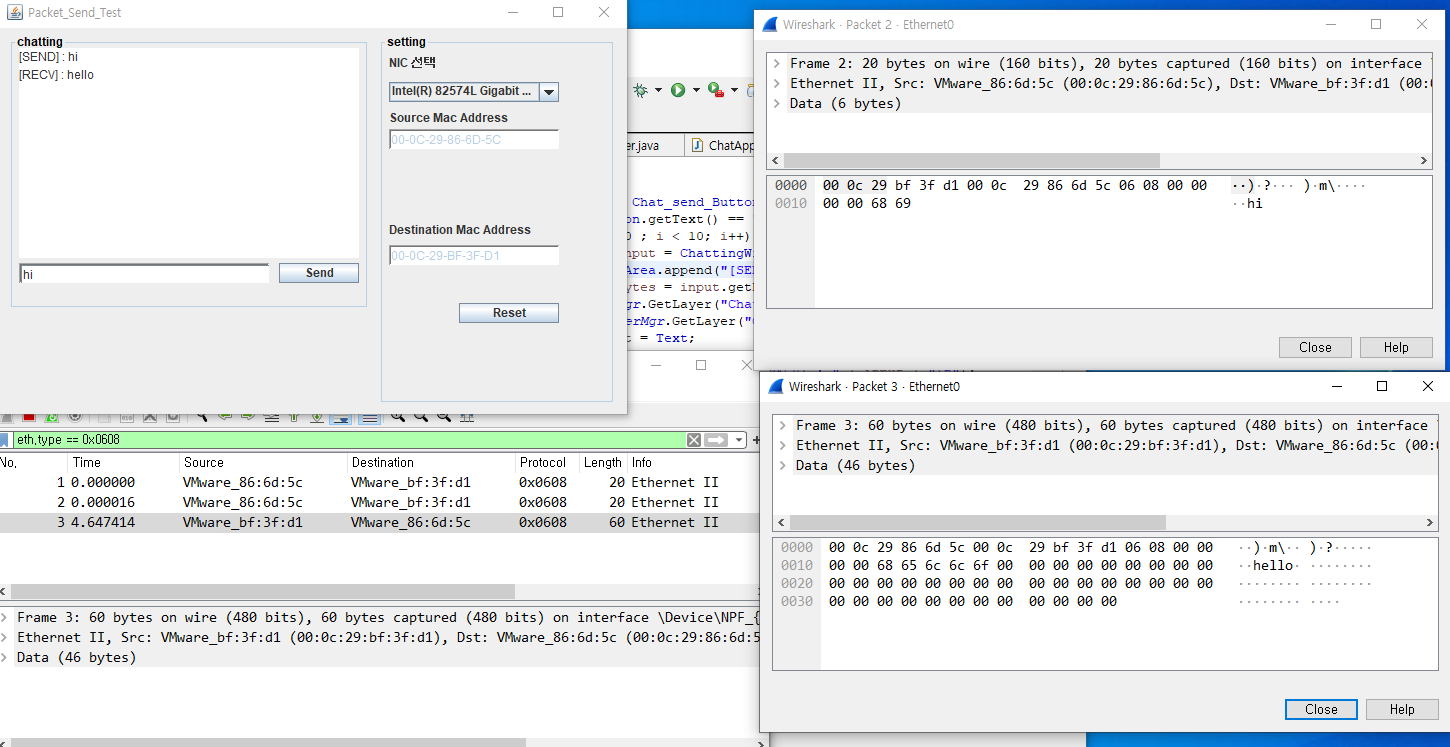


헤더 전체에서 0~5 까지는 목적지 주소, 6~11 까지는 소스 주소입니다. 12, 13은 enet\_type입니다 0x06, 0x08로 지정해 줍니다. 그 이후의 값은 data입니다.

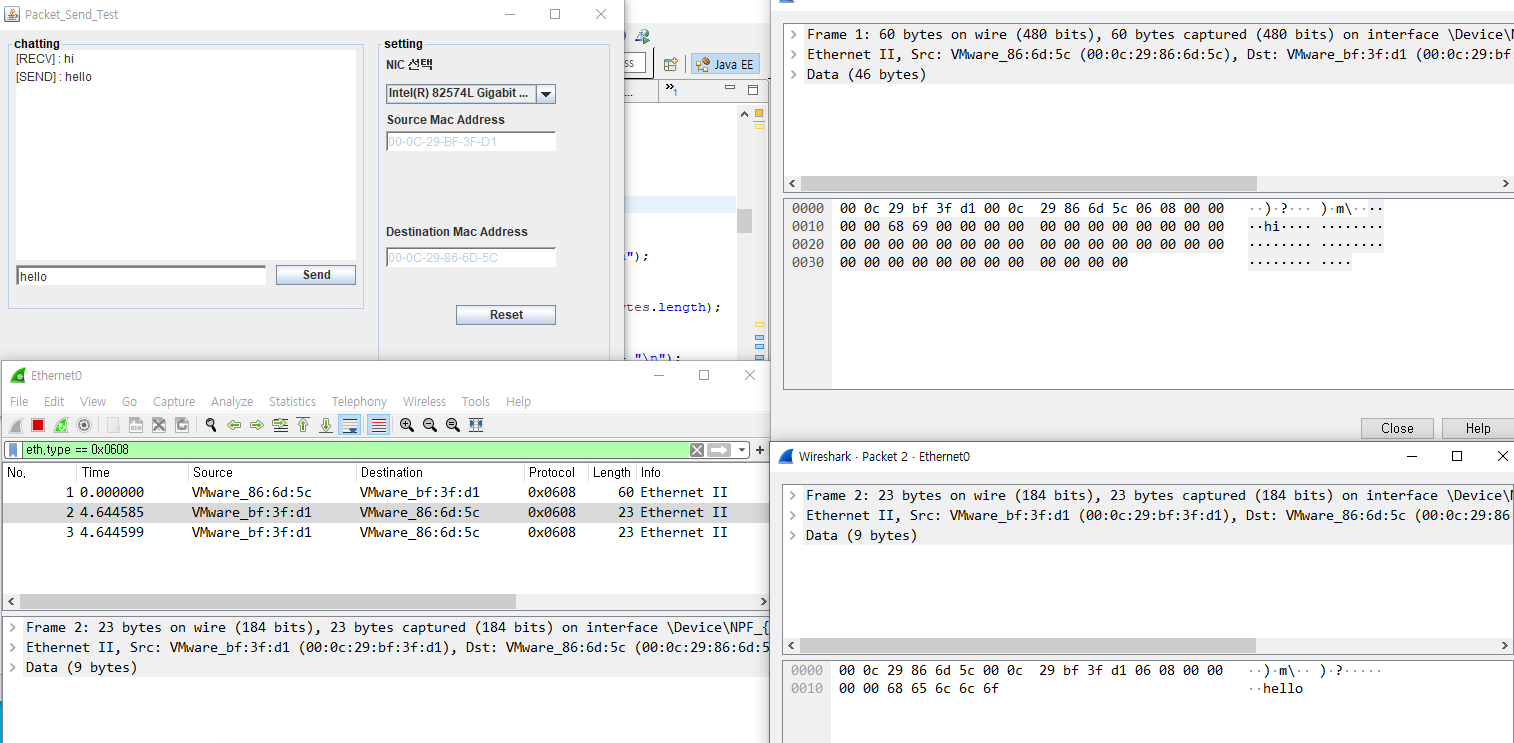


Receive 메소드입니다. 자신의 주소와 목적지 주소와 받은 데이터의 목적지 주소를 비교해줍니다.

**실행결과**



Vm1



Vm2