Chatting & File transfer

HW2 최종 보고서

|  |  |
| --- | --- |
| **데이터 통신** | 00반 |
| **조** | 6조 |
| **과제 제출일** | 2015.04.15 |
| **조장** | 201102435 박민수 |
| **조원** | 100801937 김동철 |
| **조원** | 201002483 이한가람 |
| **조원** | 201102492 이종훈 |
| **조원** | 201102495 이창영 |

**목차**

1. Chatting & File Transfer 개요

2. Layer Model & 구동 환경

3. 주요 알고리즘

4. Layer 분석

5. 실행 화면

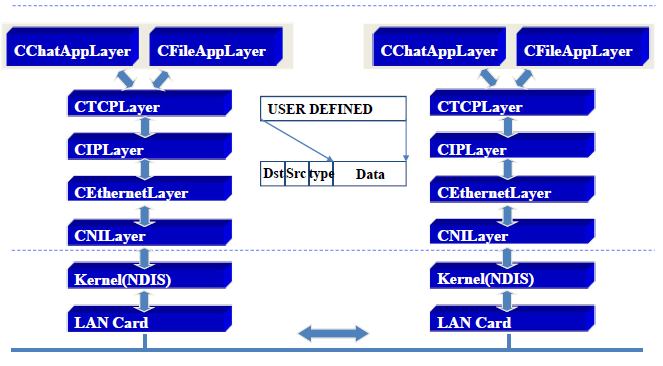
6. 느낀 점

1. **Chatting & File Transfer 개요**

HW2: IPC에서는 단일 시스템 위에 존재하는 프로세스 사이의 통신을 구현하였다. 하지만 이번에 구현한 Chatting & File Transfer는 두 대의 서로 다른 pc를 Network로 연결하고 Packet Driver를 이용하여 서로 채팅을 하거나 파일을 전송하는 통신을 한다. 이때 각 PC에서 구동 되는 프로그램은 상대(목적지)의 Mac Address를 입력 받아 서로에게 채팅 메시지나 파일의 전송을 할 수 있다.

두 과정은 Thread를 이용하여 구현되어 동시에 처리가 가능하다. 즉, 파일을 전송하는 동안에도 채팅 메시지의 전송이 가능하다. 또한, Message나 File의 크기의 제한이 없이 전송할 수 있도록 Data Fragmentation 기능을 구현하였다.

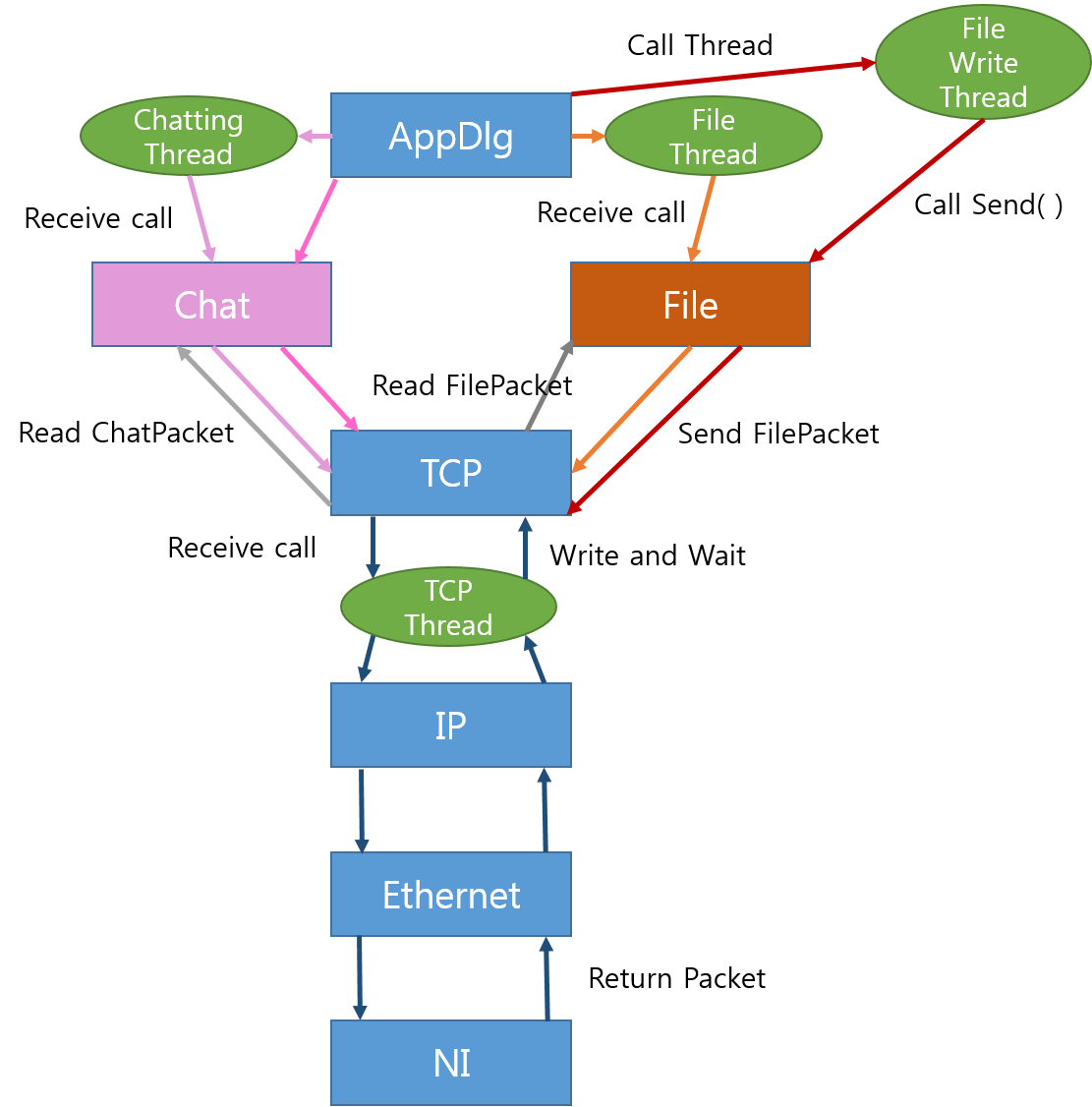
2. **Layer Model & 구동 환경**

. 2.1 Layer Model 

2.2 구동 환경

|  |  |  |
| --- | --- | --- |
|  | PC1 | PC2 |
| OS | Windows 8.1 64bits | Windows 8.1 64bits |
| IDE | Visual Studio Community 2013 | Visual Studio Community 2013 |

3. **주요 알고리즘**



< 실제 구현된 계층 관계 및 Thread >

기본적으로 Chatting과 File Transfer를 동시에 동작할 수 있도록 구현하기 위해 Thread를 이용하였다. 또한 메시지나 파일을 크기에 제한이 없이 보내기 위해 Data Fragmentation 기능을 구현하였다.  또한 Packet 전송 중, Packet 이 손실 되었을 경우 수신 측은 그 즉시 손실 된 Packet의 재 전송을 요구하고, 그 재 전송 요구를 받은 송신 측은 요청 받은 Packet부터 재 전송하는 Selective Repeat ARQ 방식을 적용하려고 하였으나, 받았다는 메시지를 보내려고 하는 것을 구현하기엔 복잡해서 패킷을 전송하다가 잃어버리는 패킷이 발생하면, 받는 쪽에서 받지 못한 패킷의 시퀸스 넘버를 보내 재전송을 요청하고, 그것을 보내는 쪽에서 받으면 받지 못 했다고 알려준 시퀸스 넘버의 패킷부터 다시 전송하게 구현하였다.

4. **Layer 분석**

a.NI Layer (Network Interface Layer)

- pcap 함수를 이용하여 packet을 보내거나 받는 역할을 한다.

1. Ethernet Layer

- Ethernet Layer의 header에는 목적지 주소와 보낸 주소의 MAC Address가 저장되고, Packet의 목적지 주소와 보낸 주소를 검사하여 Packet을 구별한다.(Broadcast도 검사)

1. IP Layer

-본 과제에서 IP address가 필요 없으므로, 계층 구조를 갖추기 위해 Layer를 구성하였다. IP Layer로 들어온 payload에 대해 Send와 Receive 함수가 구현된다.

- IP Header는 있으나 IP Header에 의미 있는 address정보는 추가되지 않는다.

1. TCP Layer

- TCP Layer는 File Layer와 Chat Layer 2개의 Layer의 연결 고리로 수신 측에서 상대방이 보낸 정보의 타입을 분류하여 Chat Layer(0x2080)와 File Layer(0x2090)중 어떤 레이어로 payload를 넘길지 판단하는 계층이다.

- Send 함수는 상위 2개의 Layer에서 넘어온 정보를 하위 Layer로 넘긴다.

- Recieve 함수는 Thread가 구성되어있는 상태에서 Rock변수를 이용하여, 파일 전송과 채팅 정보 전송이 한번에 실행되는 것처럼 착각을 보일 수 있다. 또한 처리되지 않은 패킷 데이터가 덮어 씌워지지 않도록 제어한다.

1. Chat Layer

- Chat Layer로 들어온 payload에 대해 Send와 Recevie함수가 구현되어있다. 이때 Receive의 경우 APP\_DATA\_SIZE보다 2바이트 크게 설정해준다.

- payload size + 1은 문자열을 위한 Null byte를 위한 공간이다.

- payload size + 2는 Chat Data가 fragmentation 되었을 때 , 받아온 Packet이 처음일 때는 1 로, 아닐 때는 0으로 Masking 하여 구분할 수 있도록 해준다.

1. File Layer

- File Lyer의 Send 함수는 크게 1500Byte를 넘는 payload의 길이이냐, 아니냐로 나눠서 만약 넘긴다면 단편화 처리를 하여 하위 Layer로 Send 한다.

- AppDlg Layer의 FileSended 변수는 UI에서 파일 전송을 클릭하면, true 취소하거나 아예 보내지 않는다면 false로 초기화 되어있고, 이 변수를 이용해 File Layer에서 파일 전송 취소 예외를 처리 할 수 있다. 또 packet 전송 실패 시 해당 packet의 시퀀스 넘버부터 다시 packet을 전송하는 과정도 처리하였다.

- Recieve 함수는 Thread로 넘어온 데이터를 처리하는데, 데이터는 처음과 중간 끝이라는 type을 가지고 있다. 각 type별로 처리하여 데이터를 받아 상위 레이어로 전송한다.

1. Chat App Dlg

- 최상위 Layer로 UI를 구현하는 MFC를 커버 한다.

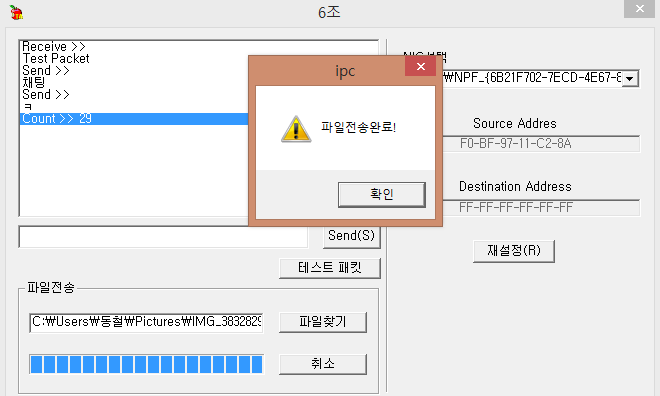
- 구현한 모든 Thread를 start 시킨다.

- Thread는 총 4가지가 구현되어있는데, 채팅 수신을 담당하는 ReceiveThread,  파일 전송을 담당하는 FileThread, NILayer로 부터 패킷을 받아와 TCP까지 이동시켜주는 ReceiveTCPThread,  FileLayer까지 파일을 받아오는 ReceiveFileThread가 있다.

5. **실행 화면**

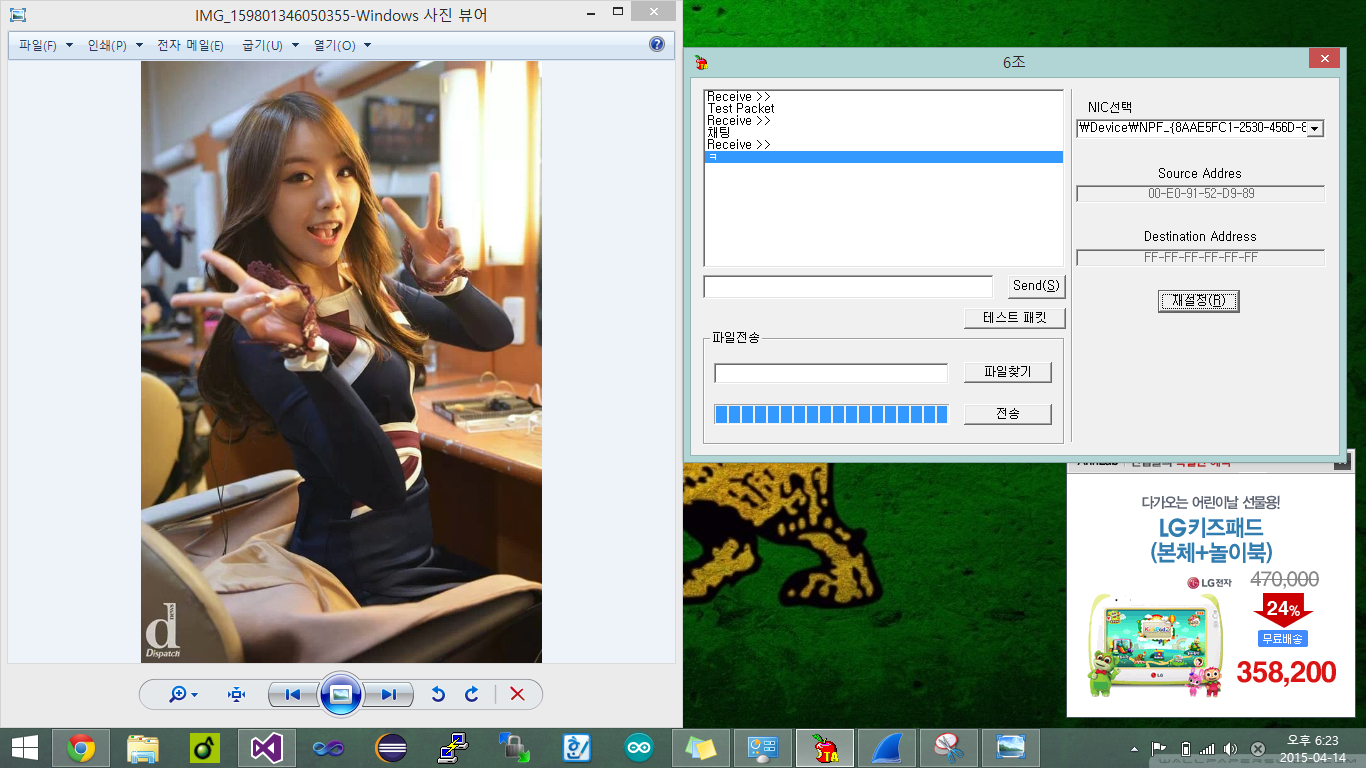


<테스트 모습>

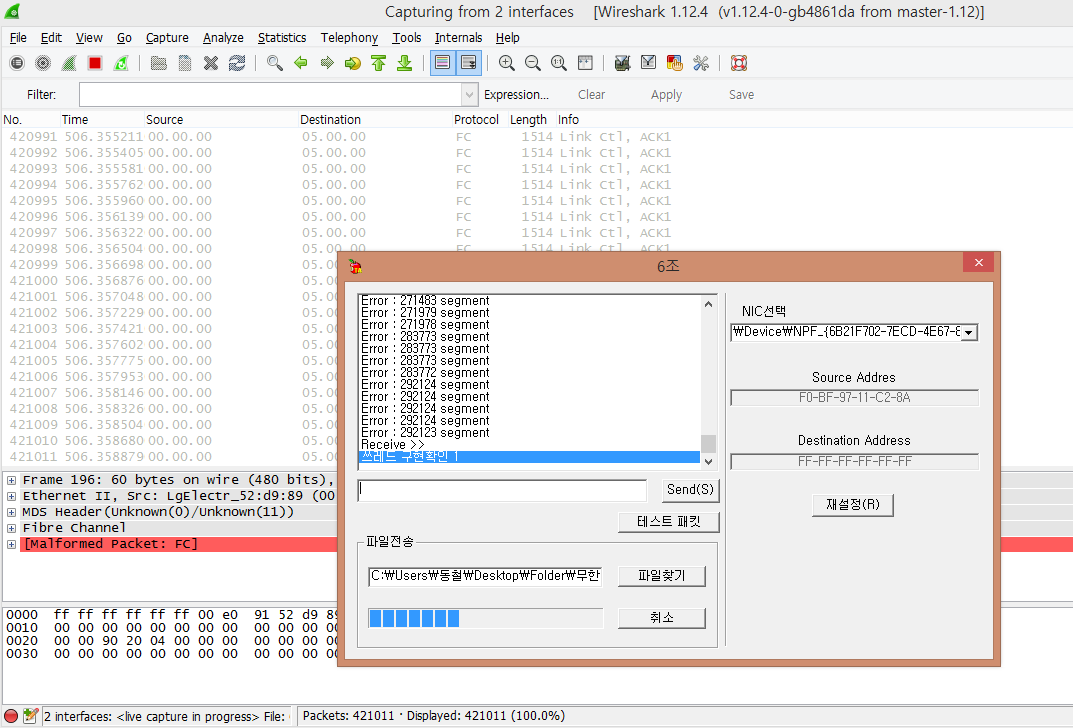


<File 전송 : 작은 크기 파일>

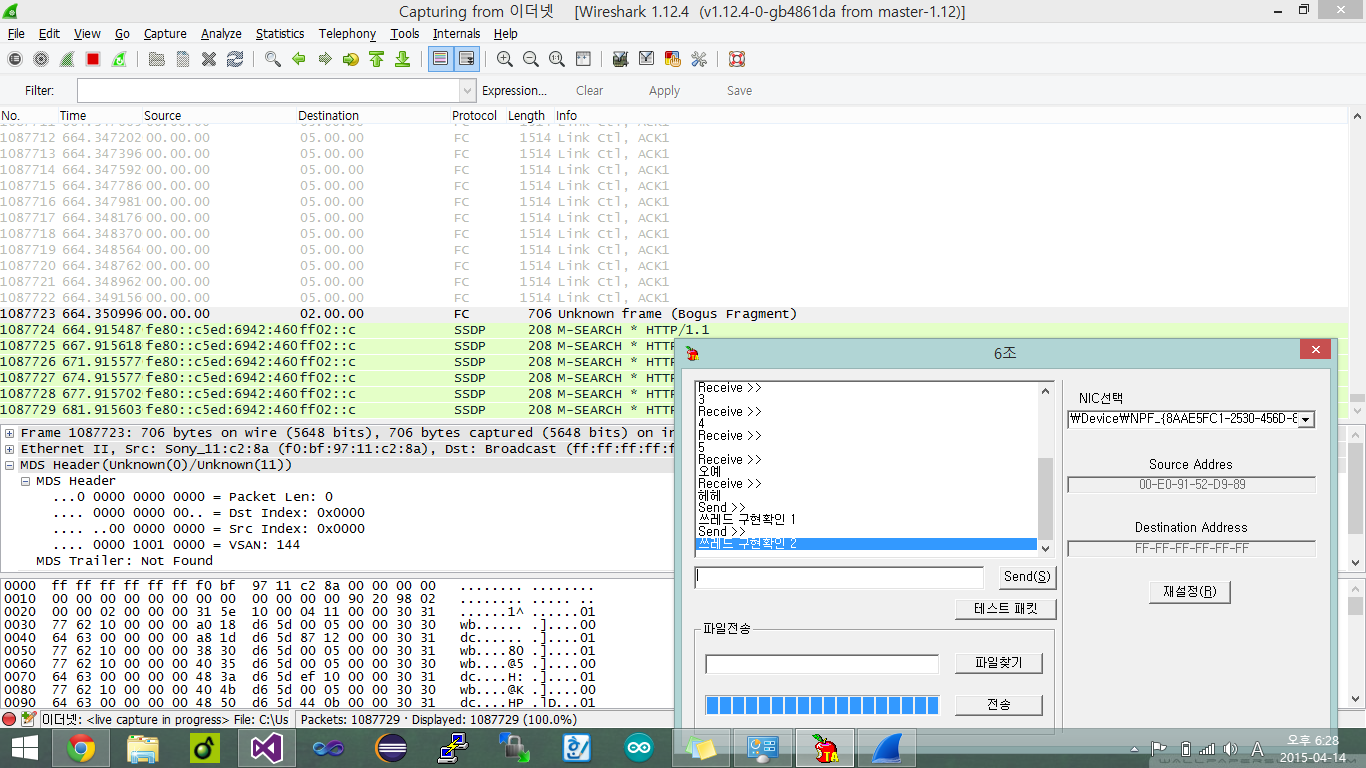
파일 전송 시 파일의 전송이 완료되면 송신자의 채팅창에 총 segment 수가 출력되며 파일 전송을 알리는 MessageBox를 출력한다.

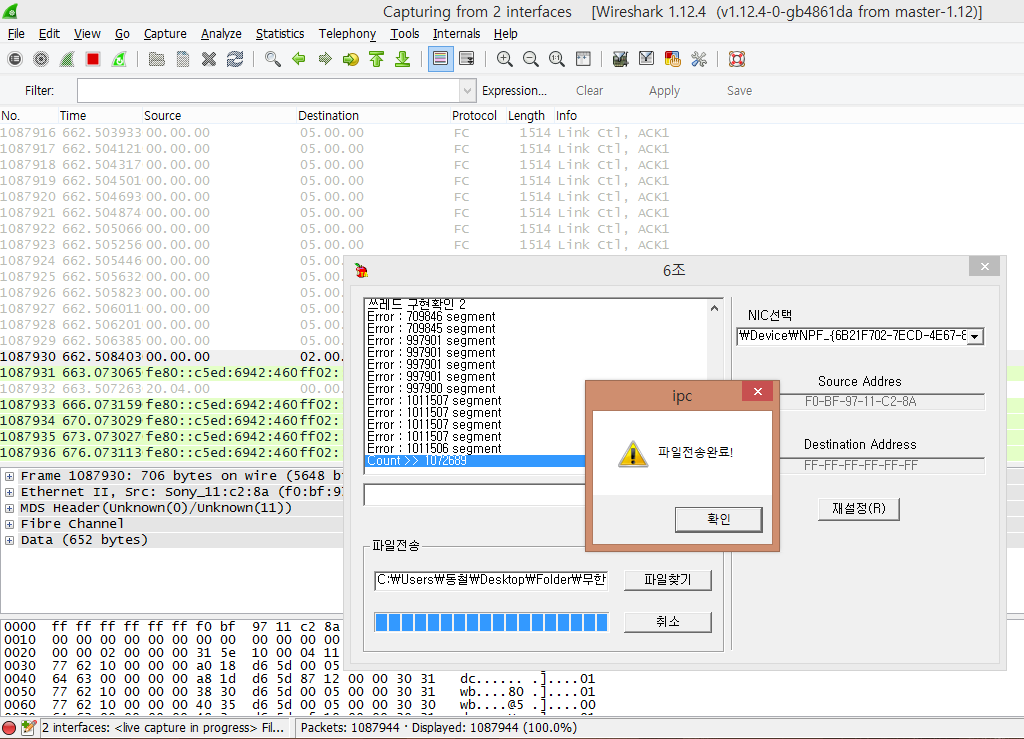


<File 전송 완료  : 작은 크기 파일>

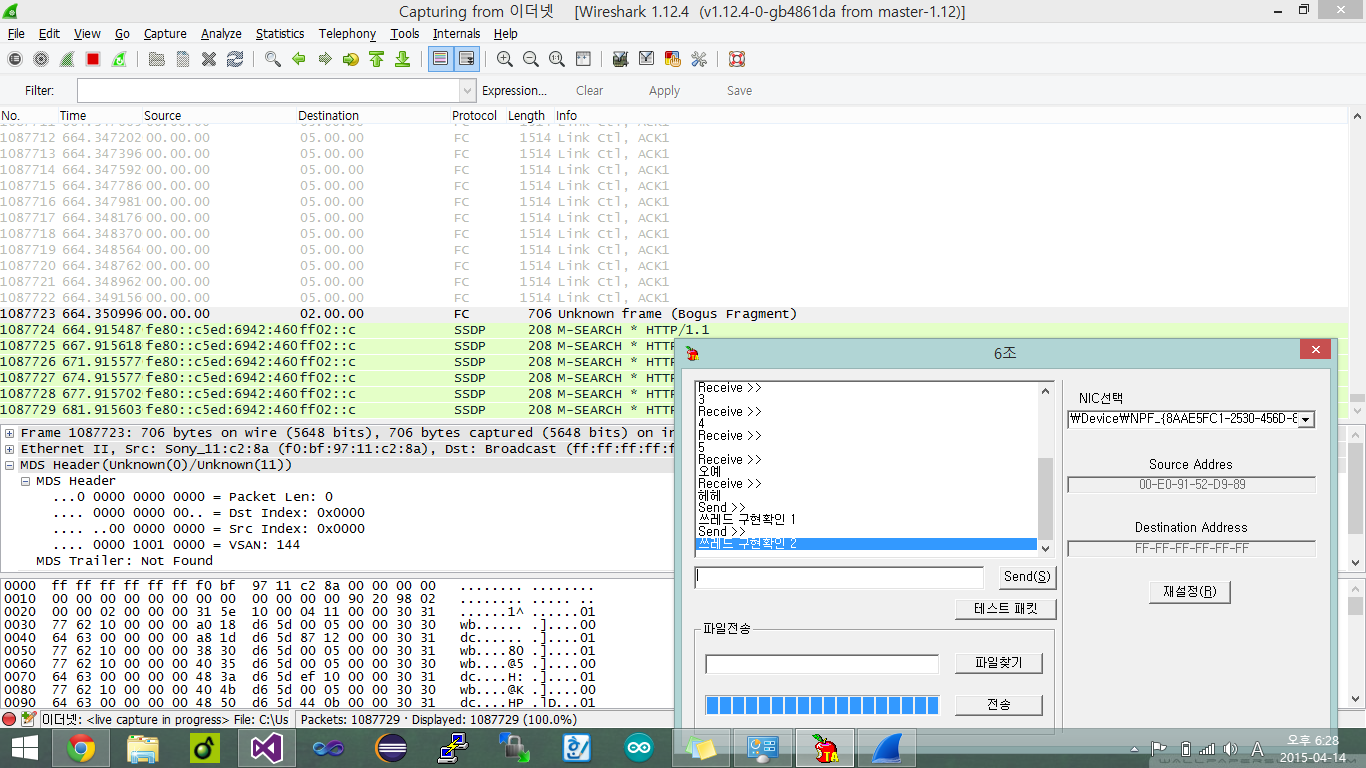


< File 전송 중 Chatting - 수신측>

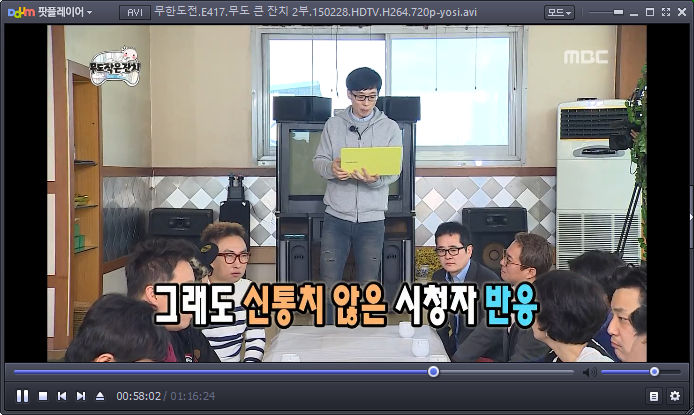
< <File 전송 중 Chatting - 송신측>



<송신자 File 전송 완료 >



<수신자 File 전송 완료 : wireshark에 last packet 도착 >



<보낸 파일 확인 : 1.5G 동영상 파일>

5. **느낀점**

|  |  |
| --- | --- |
| **김동철** | 작년에 조금이나마 들어놨던게 도움이 된 듯하다. 각 계층의 역할 , Thread의 역할 등 흐릿하게 알던 것을 확실하게 알 수 있었다. 특히 팀원들의 하려고 하는 의지가 강해서 조금 더 수월하게 진행할 수 있었다. |
| **이한가람** | 구현 보다 테스트에 어려움을 많이 겪는 과목이 될 것 같다. 코드는 디버깅이 되니 차근차근 해결할 수 있었다. 그러나 네트워크 자체에 문제가 있을 때 원인을 찾고 해결해야 하는 과정이 가장 어려웠다.  앞으로도 같은 어려움이 반복되겠지만 팀원들이 모두 열정적으로 참여하고 있어서 포기하지 않고 완주할 수 있을 것 같다. |
| **박민수** | 이전 IPC의 경우 하긴 했지만 정확하게 이해하지 못하고 했었다는 느낌이 강했는데, 이번 Chatting & File Transfer를 구현하면서 각 계층 구조에 대한 이해를 정확하게 할 수 있었고, Thread를 왜 사용하여야 하는지, 패킷은 어떤 식으로 구성이 되는지, 전송해야 되는 파일의 크기가 큰 경우에는 Data Fragmentation을 어떤 식으로 구현하는지 등에 대해 자세하게 알 수 있었다. |
| **이종훈** | IPC를 통해 계층 간의 상호작용을 이해했지만, 이번 과제가 Thread구현에 중점이 있었고, 과제를 통해 Tread의 개념과 사용 목적을 알 수 있었다. 팀원들의 도움으로 많은 것을 배워 즐겁게 과제를 했다. |
| **이창영** | 이론수업에서 이해하지 못했던 부분들도 직접 조원들과 질의응답을 하며 모르는 부분을 서로 알아갈 수 있어서 문제를  이해하는데 많은 도움이 된 것 같다.  혼자서 과제를 수행했다면 포기했을 것 같기도 한데 조원들과 다함께 서로 격려하며 할 수 있어서 좋은 경험이었던 것 같다. |