Chatting & File Transfer

2022 데이터통신 00반 3조

201802162 지명하

201802052 김기훈

201802057 김동훈

201802142 이현민

202002503 배준서

1. 실습 환경

두 대의 가상 환경 Host PC를 이용합니다. 각 PC에는 Windows OS에 Java가 설치되어 있으며 서로 가상의 스위치를 통해 연결되어 있습니다.

2. 실습 목적

이전 실습에서 Simplest, Stop&Wait 프로토콜 방식으로 두 호스트 간 통신을 구현하며 이해한 것에 더해 문자열 데이터 뿐 아니라 파일 전송 기능을 추가함으로써 계층 구조로 구성된 프로토콜을 이용한 통신 방법에 대한 이해도를 높이는 것을 목적으로 합니다.

3. 조별 구성원의 역할

지명하 - FileAppLayer, UI Layer(Dlg) 작성

배준서 - FileAppLayer 작성

김기훈 - 보고서 작성

이현민 - EthernetLayer, FileAppLayer 작성

김동훈 - ChatAppLayer 작성

4. 실습 시나리오

두 대의 가상 Host PC에서 각각 프로그램을 실행합니다.

PC 1(송신 측)에서는 전송할 파일을 선택하고 텍스트를 입력합니다. 파일은 파일 전송, 텍스트는 텍스트 박스 옆 전송 버튼을 통해 전송을 시작합니다.

텍스트 전송이라면 ChatAppLayer - EthernetLayer를 통해 NILayer로 전달되는데, ChatAppLayer 에서는 텍스트 길이와 단편화 정보, EhternetLayer 에서는 송/수신지의 주소와 Layer/ACK 패킷 구분 정보를 담은 헤더가 추가되어 Encapsulation Packet이 만들어집니다.

파일 전송이라면 FileAppLayer - EthernetLayer를 통해 NILayer로 전달됩니다. FileAppLayer 에서는 파일 크기와 단편화 정보, 파일 송수신을 위한 메시지 정보, EthernetLayer 에서는 송/수신지의 주소와 Layer/ACK 패킷 구분 정보를 담은 헤더가 추가되어 Encapsulation Packet이 만들어집니다.

만들어진 패킷은 NILayer를 통해 네트워크로 전송됩니다.

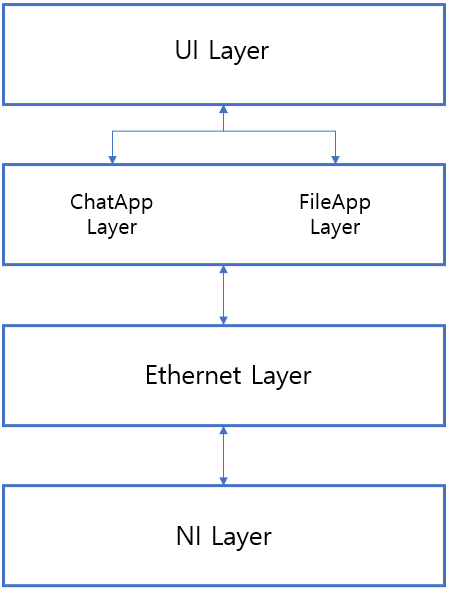
PC 2(수신 측)에서는 PC 1으로부터 패킷을 수신합니다.

패킷이 수신되면 EthernetLayer에서는 PC 2로 전송된 패킷인지, 자신이 자신에게 보낸 패킷이 아닌지 확인한 뒤, ChatAppLayer 또는 FileAppLayer 중 어디로 전달해야 할 패킷 또는 ACK인지 확인하여 ChatAppLayer와 FileAppLayer로 헤더를 제거한 패킷을 전달합니다.

텍스트 전송 패킷이 수신된 경우, 헤더 정보를 통해 단편화가 이루어진 패킷인지 판단해 아니라면 수신해 UILayer로 전달하고 단편화된 패킷이라면 수신한 패킷들을 모아 UILayer에 전달합니다.

파일 전송 패킷이 수신된 경우도 마찬가지로, 헤더 정보를 통해 단편화가 이루어졌는지 판단해 패킷을 모아 UILayer에 전달합니다.

UILayer 에서는 전달받은 데이터를 텍스트라면 화면에 표시하고 파일은 지정한 위치에 저장하도록 합니다.

5. 프로토콜 스택

프로토콜 스택은 UI, ChatApp/FileApp, Ethernet, NI의 4계층으로 구성되어 있습니다.

최상위 계층은 UI Layer입니다.

UI Layer는 송신 시 프로그램 사용자로부터 전송할 텍스트와 파일을 입력 받습니다. 텍스트를 입력 받아 전송하는 경우 하위 계층으로 전달할 때 ChatApp Layer로 전달합니다. 파일을 전송하는 경우에는 FileApp Layer로 전달합니다.

수신 시에는 하위 계층으로부터 전달받은 데이터를 각각 ChatApp Layer로부터 온 것인지, FileApp Layer로부터 온 것인지 구분하여 ChatApp Layer로부터 온 텍스트 데이터라면 프로그램을 통해 사용자에게 데이터를 출력합니다. FileApp Layer로부터 받은 파일이라면 사용자가 지정한 위치에 수신한 파일을 저장합니다.

다음 계층은 ChatApp/FileApp Layer입니다.

ChatApp Layer는 텍스트 데이터 송수신을 위한 계층입니다. 송신 시 사용자가 입력한 텍스트 데이터가 UI Layer로부터 전달되면 데이터의 크기가 단편화 없이 한 번에 송신할 수 있는 크기인 MTU 이상인지 판단합니다. MTU 이하라면 데이터의 크기와 단편화를 하지 않아도 된다는 타입 정보를 담은 헤더를 추가하여 Ethernet Layer로 넘겨주게 됩니다. MTU 이상의 크기를 가진 데이터라면 MTU 이하의 크기로 단편화 하고 단편화 된 데이터에 해당 데이터의 크기와 단편화 된 데이터라는 정보를 담은 헤더를 추가하여 Ethernet Layer에 전달합니다.

수신 시에는 수신된 데이터의 헤더 정보를 통해 단편화 된 데이터인지 판단합니다. 단편화 되지 않은 데이터의 경우 UI Layer로 전달하고, 단편화 된 데이터들은 단편화 된 첫 데이터부터 마지막 데이터까지 수신한 후 합쳐서 UI Layer에 전달합니다.

FileApp Layer는 파일 송수신을 위한 계층입니다. 송신 시 사용자가 전송할 파일을 UI Layer로부터 전달받습니다. 송신을 시작하면 수신 측에 파일을 바로 전송하지 않고 파일 이름을 가진 데이터를 전송하며 파일을 수신할 것인지 묻는 패킷을 보내게 됩니다. 해당 패킷의 정보를 통해 수신자가 파일을 수신하고 싶지 않은 경우 파일을 송신하지 않고, 파일을 수신하겠다는 패킷이 도착하면 파일 송신을 시작합니다. ChatApp Layer에서와 마찬가지로 MTU 보다 큰 파일의 경우 단편화를 통해 데이터를 나누고 해당 데이터에 크기와 단편화 된 데이터라는 정보를 담은 헤더를 추가하여 Ethernet Layer에 전달합니다. MTU 이하의 파일의 경우 파일 크기와 단편화 되지 않은 데이터라는 정보를 담은 헤더를 추가해 전달합니다.

수신 시에는 상기한 바와 같이 파일의 수신 여부를 결정합니다. 수신을 결정하면 송신자에게서 온 데이터를 전달받습니다. 마찬가지로 헤더에 담긴 정보를 통해 단편화 된 파일인지 판단하여 단편화 된 파일은 첫 데이터부터 마지막 데이터까지 수신 후 다시 파일 상태로 UI Layer에 전달합니다.

다음 계층은 Ethernet Layer입니다.

Ethernet Layer에서는 송신 시 ChatApp/FileApp Layer로부터의 패킷에 송수신지 주소와 둘 중 어떤 계층의 패킷인지 혹은 ACK인지의 정보를 담은 헤더를 추가해 NI Layer에 전달합니다.

수신 시에는 헤더 정보를 통해 ChatApp과 FileApp 중 어떤 Layer의 패킷 혹은 ACK 패킷인지 판단하여 각 Layer로 전달합니다.

다음 계층은 NI Layer입니다.

NI Layer에서는 Pcap 라이브러리로 작성된 코드에 의해 패킷이 네트워크로 송수신됩니다.

6. 구현 설명

LayerInterface

텍스트이(가) 표시된 사진

자동 생성된 설명

각 Layer들이 가지고 있어야 할 메소드들을 인터페이스로 정의하여 각 Layer에서 구현할 수 있도록 했습니다. 이 인터페이스는 아래의 BaseLayer에서 구현됩니다.

BaseLayer

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

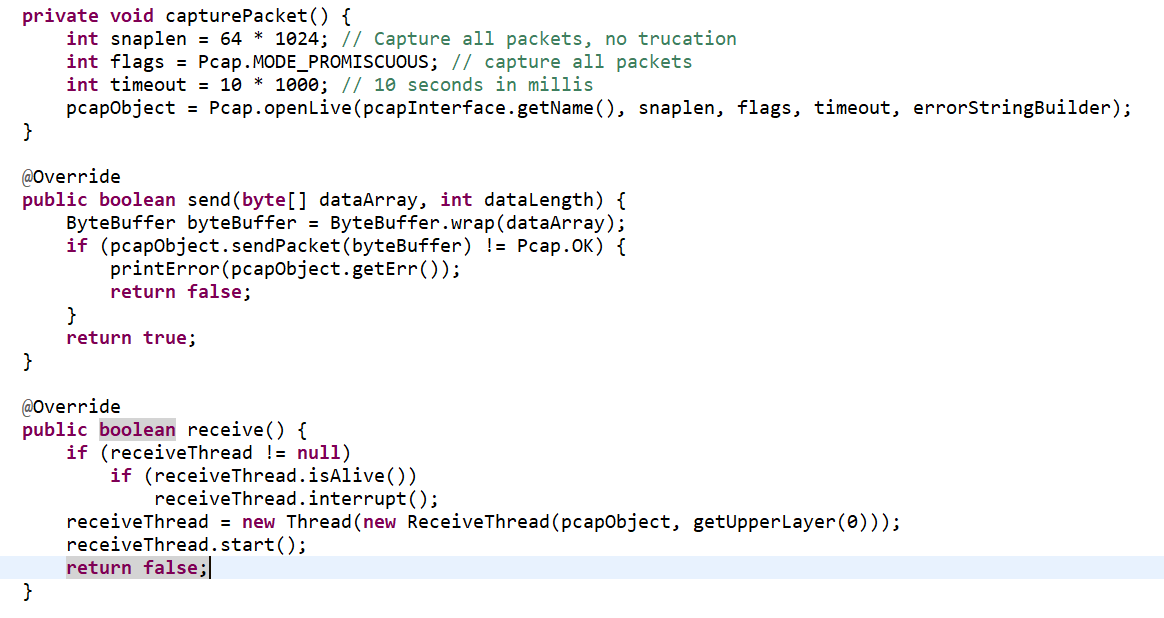
자동 생성된 설명

BaseLayer에서는 위의 인터페이스에 선언된 메소드들을 구현합니다. layer에 대한 getter 및 setter, 송수신에 필요한 send와 receive를 각각 구현합니다. UI Layer부터 NI Layer까지 모든 Layer들은 Base Layer를 상속받아 각 메소드를 Override 하는 방식으로 작성되었습니다.

NI Layer

텍스트이(가) 표시된 사진

자동 생성된 설명



기존 Simplest와 Stop&Wait에서와 마찬가지로 Pcap 라이브러리를 통해 패킷을 전달합니다. 추가로 windows 환경 뿐 아니라 linux 환경을 지원하도록 구현했습니다. send에서는 pcap으로 패킷을 보내고 receive에서는 패킷을 받아 상위 Layer로 전달합니다.

EthernetLayer

텍스트이(가) 표시된 사진

자동 생성된 설명

Ethernet 프레임을 만드는 메소드입니다. 이 때 TCP/IP 헤더를 사용하지 않기 때문에 MTU 크기와맞추기 위해 TCP/IP 헤더 크기인 40 만큼을 조정합니다. 또한 통신에 필요한 데이터의 최소 크기가 60이기 때문에 60보다 작은 경우 데이터 크기가 60 이상이 되도록 크기를 맞춰줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ehternet Layer의 헤더를 제거합니다. TCP/IP 프로토콜의 헤더 크기 만큼인 40을 위에서 헤더에 더했기 때문에 40 + 주소, 타입 헤더 크기 14로 총 54만큼이 헤더 크기이기 때문에 수신한 프레임에서 54를 빼 헤더를 제거합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

패킷을 하위 Layer로 전달하는 Send 메소드입니다. 상위 계층의 Layer가 두 가지이기 때문에 LayerName 이라는 변수를 이용해 어느 Layer에서 온 데이터인지 구분합니다. ChatApp Layer의 데이터라면 ACK인지 혹은 전송해야 할 데이터인지 확인해 타입 헤더 값을 조정하여 NI Layer에 전달합니다. FileApp Layer의 데이터도 마찬가지로 동작합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명NI Layer로부터 수신된 패킷을 전달받는 receive 메소드입니다. 먼저 패킷의 브로드캐스트 여부, 목적지가 자신인지 여부, 자신이 전송한 패킷이 자신에게 수신되었는지 여부를 확인합니다. 이후 헤더를 제거하며 헤더 정보를 통해 각각 ChatApp의 ACK/데이터, FileApp의 ACK/데이터 여부를 파악해 상위 레이어로 전달합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ethernet Layer의 헤더입니다. 헤더의 필드는 목적지 주소, 송신지 주소, ChatApp과 FileApp Layer 중 어느 곳의 ACK/데이터인지를 담은 타입, 데이터 정보로 구성됩니다.

ChatAppLayer

텍스트이(가) 표시된 사진

자동 생성된 설명

상기했듯이 TCP/IP 헤더에 사용되는 크기인 40이 빠지고, ChatApp 헤더 크기인 4가 추가로 빠져 단편화 없이 전송 가능한 최대 크기 MTU를 1456으로 설정합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

헤더를 추가해 프레임을 만드는 메소드입니다. 헤더 크기 4만큼 데이터 크기를 조정하고 헤더 정보를 담습니다. 텍스트이(가) 표시된 사진

자동 생성된 설명

Ethernet Layer로부터 프레임을 수신한 경우 ChatApp의 헤더 크기인 4만큼을 프레임에서 빼 헤더와 데이터를 분리합니다. 텍스트이(가) 표시된 사진

자동 생성된 설명

MTU 크기 이상의 데이터를 전송 시 사용하는 메소드입니다. 첫 번째로 보내는 데이터는 단편화 된 데이터를 전송한다는 타입 정보와 전체 데이터 크기를 담은 헤더를 붙여 프레임을 만들고 MTU 크기 만큼 데이터를 나누어 전송합니다. 중간 과정은 MTU 크기만큼 나누어진 데이터를 순차적으로 전송합니다. 전송 시에, 나누어진 하나의 데이터를 보내고 나면 바로 전송하지 않고 ACK 패킷이 도착했는지를 기다린 후에 다음 데이터를 전송합니다. 전송이 끝나고 마지막 데이터가 전송될 때에는 MTU 크기가 아닌 남은 데이터의 크기 만큼 프레임을 만들어 전송합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

데이터를 전송하는 메소드입니다. 단편화하지 않은 데이터의 경우 헤더의 타입 정보를 단편화 하지 않은 데이터 타입으로 넣고 전송합니다. 단편화 된 데이터의 경우 위의 fragmentSend 메소드를 호출하여 단편화 전송합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

패킷을 수신하는 메소드입니다. frame이 null로 수신되었다면 ACK 패킷입니다. null이 아니라면 헤더에서의 타입 부분을 보고 결정합니다. 0x00인 경우 단편화하지 않은 데이터이므로 상위 Layer로 전달합니다. 0x01의 경우 단편화 된 데이터의 시작 부분이므로 텍스트를 저장할 배열에 데이터를 넣고 ACK를 송신한 뒤 다음 데이터를 기다립니다. 0x02의 경우도 마찬가지로 단편화 된 데이터를 배열에 차례대로 추가하며 ACK 송신을 반복합니다. 타입이 0x03이라면 데이터를 모아 둔 배열에 마지막 데이터를 추가하고 해당 배열을 상위 Layer로 전달합니다. 텍스트이(가) 표시된 사진

자동 생성된 설명

ChatApp의 헤더 필드는 전체 데이터 길이, 단편화, 데이터 정보로 구성됩니다.

FIleAppLayer

텍스트이(가) 표시된 사진

자동 생성된 설명

ChatApp Layer에서와 마찬가지로 헤더 크기를 제외한 크기로 MTU 값을 설정합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

헤더를 추가해 프레임을 만드는 메소드입니다. 파일 데이터 크기에 헤더 크기를 더한 크기로 프레임을 만들고 데이터와 헤더를 프레임에 추가합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Ethernet Layer로부터 프레임을 전달받으면 헤더 크기인 12를 빼 헤더와 데이터를 분리합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

ACK와 비슷하게 동작하지만 ACK와는 다르게 파일을 전송할 때 수신자에게 파일을 수신받을 것인지에 대한 응답을 기다리는 메소드입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

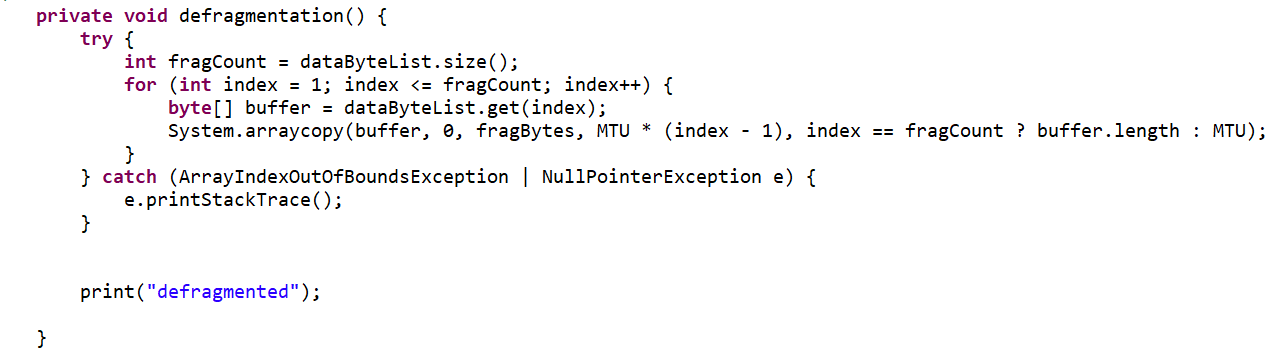
자동 생성된 설명

ChatApp에서와 같이 동작하는 단편화 전송 메소드입니다. 파일을 MTU 크기로 나누어서 전송합니다. 첫 번째로 전송하는 데이터는 단편화 된 파일 데이터의 시작 부분이라는 정보와 파일 크기를 담은 헤더가 추가되어 Ethernet Layer로 전송됩니다. 마찬가지로, ACK를 기다린 후 다음 전송을 이어갑니다. 마지막 데이터를 보내기 전 까지는 단편화 된 파일 데이터라는 정보를 담은 헤더가 추가되어 Ethernet Layer로 전송되고 ACK를 기다리기를 반복합니다. 마지막 데이터를 보내는 경우 단편화 된 파일 데이터의 끝 부분이라는 정보를 담은 헤더가 추가되고, 남은 데이터의 크기만큼 크기를 조절해 Ethernet Layer로 전달합니다. 각 과정에서 전송이 이루어지면 progress bar를 업데이트 해 전송이 얼마나 이루어졌는지를 사용자에게 보여줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

파일명과 함께 파일을 수신할 것인지 묻는 응답 요청에 응답을 전송하는 메소드입니다. 파일을 수신하고 싶다면 수신할 것이라는 응답을 보냅니다. 응답에 따라 파일 송신 여부가 결정됩니다.



MTU 크기로 단편화 된 파일 데이터를 합쳐주는 메소드입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자가 지정한 경로에 수신한 파일을 저장하도록 하는 메소드입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

파일 데이터를 전송하는 메소드입니다. 파일 데이터의 크기가 MTU보다 크다면 fragmentSend를 호출하여 단편화 하여 Ethernet Layer로 전달합니다. 파일 데이터의 크기가 MTU 이하라면 헤더를 추가하여 Ethernet Layer로 전달합니다.

텍스트이(가) 표시된 사진

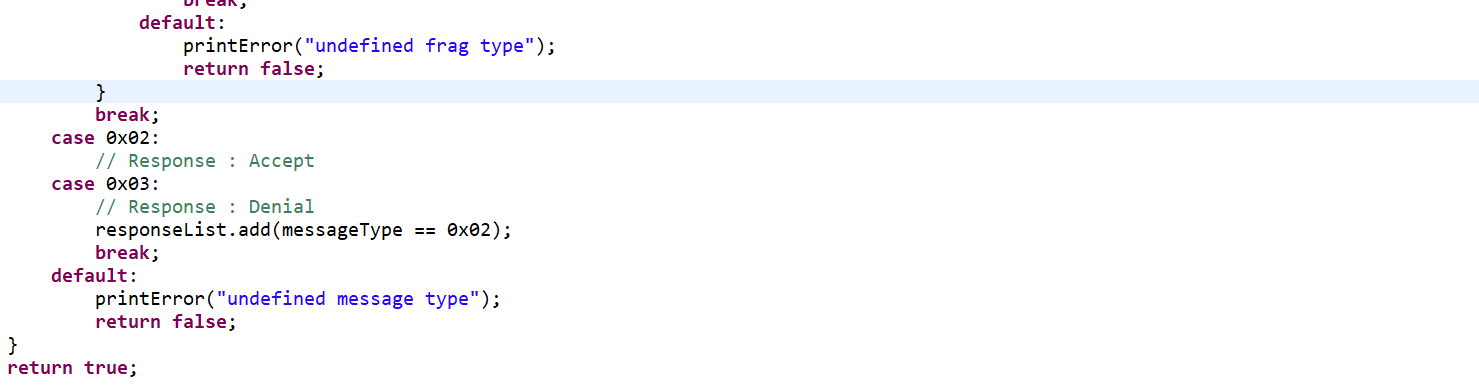
자동 생성된 설명

전송할 파일을 읽어와 전송하는 메소드입니다. 송신자의 filepath로부터 전송할 파일을 읽어옵니다. 파일을 전송하기 전, 파일 이름과 함께 파일을 수신할 것인지 수신자에게 요청을 보냅니다. waitResponse를 통해 파일 전송이 허가되면 위의 Send 메소드를 통해 파일이 전송됩니다. 만약 수신자가 파일 전송을 불허하면 파일 전송이 취소됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명



Ethernet Layer로부터 프레임을 전달받으면 ACK 패킷인지 확인합니다. 파일 데이터가 수신되었다면 헤더에서 타입 부분을 분리해 어떤 종류의 데이터인지 판단합니다. 파일 이름이나 응답이 아닌 0x01로 파일 데이터인 경우 단편화 타입 정보를 확인합니다. 단편화 되지 않은 파일 데이터의 경우 파일을 저장합니다. 단편화 된 데이터 중 첫 번째 데이터라면 배열에 데이터를 임시 저장합니다. 단편화 된 데이터의 중간 부분이라면 MTU 크기씩 데이터 배열에 순차적으로 저장합니다. 이 때 몇 번째 데이터인지 표시하고 progressbar를 업데이트 합니다. 마지막 데이터라면 배열에 데이터를 추가한 뒤 defragmentation을 호출하여 데이터들을 하나의 파일로 합칩니다.



FileApp Layer의 헤더는 파일의 총 크기, 단편화 정보, ACK/데이터/Response 종류 정보, 단편화 전송 시 몇 번째 데이터인지에 대한 정보로 구성되어 있습니다.

UI Layer

텍스트이(가) 표시된 사진

자동 생성된 설명

UI Layer에는 main 메소드가 있고, 이 main 메소드에서는 Layer manager에 각 Layer를 추가하고 연결합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트를 송신하는 경우입니다. 주소 설정이 완료된 상태에서 send 버튼이 눌렸을 때 사용자가 입력한 텍스트를 ChatApp Layer로 전달합니다. 이 때 byte 형태로 길이와 함께 전달합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

파일을 송신하는 경우입니다. 주소 설정이 완료된 상태에서 filesend 버튼이 눌렸을 때 송신자의 filepath로부터 파일을 전송합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

하위 레이어에서 데이터를 수신하면 layerName을 통해 어느 Layer에서 전달된 것인지 구분합니다. ChatApp Layer에서 전달된 텍스트 데이터라면 GUI를 통해 프로그램 사용자에게 수신한 텍스트 내용을 출력합니다.

FileApp Layer에서 전달된 파일 데이터의 경우는 FileApp Layer에서 이미 지정한 경로로 저장한 상태이므로 파일이 수신되었다는 메시지를 프로그램 사용자에게 출력합니다.

7. 실험 환경

VMware 가상환경에서 두 개의 Host가 가상 스위치를 통해 연결된 상태에서 진행

두 Host는 서로 다른 물리적 주소를 가짐

8. 실행 결과

9. 프로젝트 장점 및 느낀 점

프로젝트 장점 : 파일 저장 위치를 수신자가 선택 가능하게 구현하였습니다.

progress bar 구현으로 전송 상황을 확인 가능합니다.

Windows OS가 아닌 리눅스 OS에서도 실행 가능합니다.

수신자가 파일 수신 여부를 선택 가능합니다.

느낀 점

지명하 : 이번 팀 프로젝트로 자바 GUI 프로그래밍과 레이어 간의 데이터 통신 규격에 대하여 이해할 수 있었고, 무엇보다 팀에서 조장의 필요성과 중요성을 알 수 있었던 것 같습니다.

김기훈 : 여러 Layer를 거치면서 데이터가 전송되기 위한 준비가 갖추어지고 실제로 전송되는 과정이 복잡하지만 정교하다는 점을 알게 되었고 여러 가지 상황에서 데이터를 효과적이고 효율적으로 전송하기 위한 고민이 많이 필요할 것이라 생각했습니다.

김동훈 : 단편화를 하게 되면, 전송시간이 빠르고 통신 중에 오류를 효과적으로 제어할 수 있다는 장점과 동시에, 너무 작은 블록으로 단편화 하게 되면, 재결합 시 처리 시간이 길어지고, 실제 데이터 외에 헤더 등의 부수적인 데이터가 많아져 비효율적인 문제가 생긴다는 것을 알 게 되었고, 가장 적합한 단편화 크기를 찾는 것이 중요함을 알게 되었습니다.

이현민 : 간단한 file이나 text를 송수신 할 때에도 안정적으로 송수신 하기위해 여러 Layer를 거쳐 추가적인 정보를 포함하여야 하고 또한 크기가 큰 data는 단편화 하여 송수신 하는게 좋다는 것을 알게 되었습니다

배준서 : 단순히 File data를 전송하는 데에도 여러 헤더 정보와 패킷 송수신이 이뤄짐을 알 수 있었습니다. 그리고 Layer계층을 구현할 때 헤더 정보를 읽을 때에도, 가장 자주 사용되는 헤더 정보가 우선 비교되도록 하는 편이 일반적인 통신 속도도 더 빨라지지 않을까 생각했습니다.

특히 FileAppLayer의 경우 송수신에 MTU 단편화가 많아서, 이러한 제어문의 설계가 실제 성능에도 영향을 미침을 알게 되었습니다.