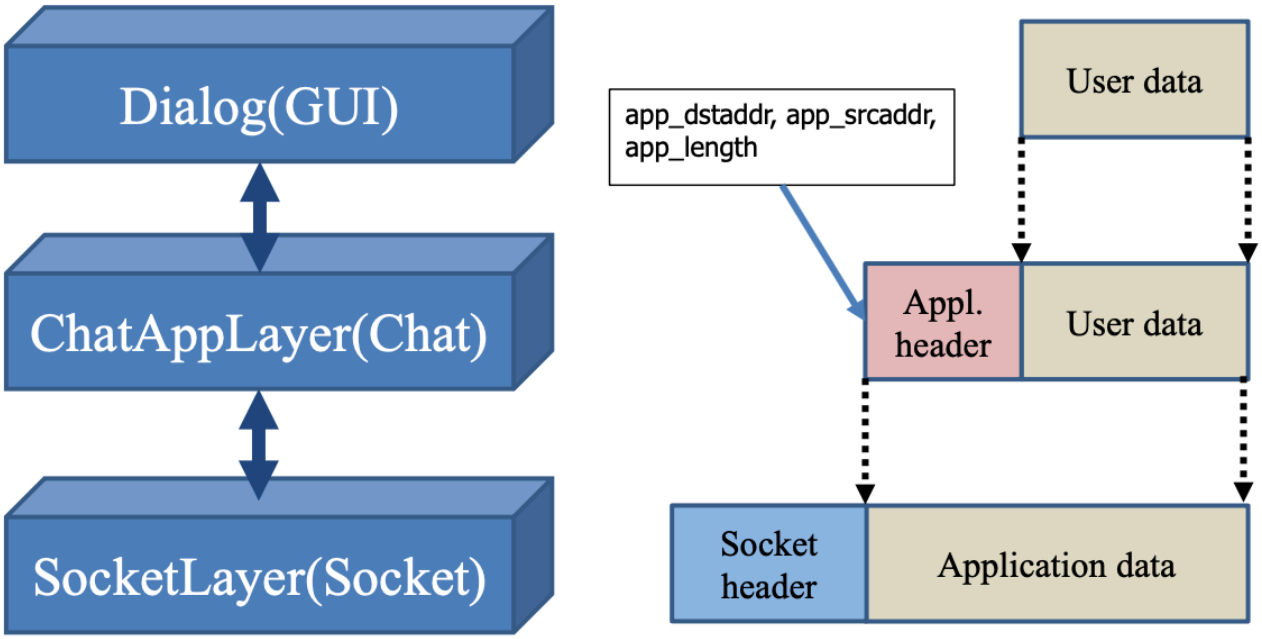
데이터 통신 과제
제출일 : 2020-04-10
201602022 양승훈

Inter-Process Communication

실습 개요
Socket 네트워크를 통해 시스템 내의 프로세스간의 통신을 구현한다.

프로토콜 스택
구조



프로토콜 스택은 각 레이어를 단계적으로 분리하고 계층간의 상하관계를 정의해 절차를 밟아 가며 통신을 하도록 한 것이다. 각 계층은 아래 계층으로부터 서비스를 제공받고 위 계층에 서비스를 제공한다. 계층을 내려가며 필요한 데이터를 추가하는 것이 Encapsulation, 계층을 올라가며 필요없는 데이터를 제거하는 과정이 Demultiplexing이다. 이번 Inter-Process Communication 과제에서는 위 그림같이 계층이 구성되어, 계층을 내려가며 메시지의 출발지, 도착지정보가 Encapsulation되고, 계층을 올라오며 Demultiplexing되고 GUI상에 표시되도록 한다.

TCP/IP 계층에서 소켓의 위치
소켓이 작동하는 방식으로 볼 때 세션을 만들고 서비스 포인트를 지정하는 역할을 모두 하므로 OSI 7계층에서 세션/전송계층, TCP/IP에서의 응용/전송계층의 일을 하는것이라고 처음에 생각을했고, 이 생각에 기반해서 구글링을 통해 알아본 결과, 소켓은 어느 OSI 7계층에서 어디에도 속하지 않는 완전히 다른 개념이라는 글을 찾을 수 있었고, 여러 계층을 거치지 않고, 소켓을 통해 단 3개의 계층으로 줄여 응용계층과 소켓계층만으로 더욱 간단하게 네트워크 통신을 할 수 있도록 해주는 것이라는 글도 찾을 수 있었다.

구현 설명

IPCDlg.java

```
52  public static void main(String[] args) {
53      // TODO Auto-generated method stub
54      m_LayerMgr.AddLayer(new SocketLayer("Socket"));
55      m_LayerMgr.AddLayer(new ChatAppLayer("ChatApp"));
56      m_LayerMgr.AddLayer(new IPCDlg("GUI"));
57      m_LayerMgr.ConnectLayers(" Socket ( *ChatApp ( *GUI ) ) ");
58  }
```

LayerManager인 m_LayerMgr에 SocketLayer, ChatAppLayer, IPCDlg를 추가하고 연결해준다.

```
150     private static final boolean DEBUG = true;
151
152     public void debugMSG(String message) {
153         if(DEBUG) {
154             System.out.println("[IPCDlg]" + message);
155         }
156     }
```

Debug Message를 출력해가며 코드를 짜기위해 디버그 상수와 함수를 구현해줬다. 동일한 방식으로 ChatAppLayer, SocketLayer에도 구현해줬다.

```
159     public void actionPerformed(ActionEvent e) {
160         String action = e.getActionCommand();
161         if(action.equals("Setting")) {
162             Setting_Button.setText("Reset");
163             srcAddress.setEnabled(false);
164             dstAddress.setEnabled(false);
165
166             int source = Integer.parseInt(srcAddress.getText());
167             int destination = Integer.parseInt(dstAddress.getText());
168
169             debugMSG("소스 포트 입력 : " + source);
170             debugMSG("도착지 포트 입력 : " + destination);
171
172             ((SocketLayer) (m_LayerMgr.GetLayer("Socket"))).setClientPort(source);
173             ((SocketLayer) (m_LayerMgr.GetLayer("Socket"))).setServerPort(destination);
174             ((ChatAppLayer) (m_LayerMgr.GetLayer("ChatApp"))).SetEnetSrcAddress(source);
175             ((ChatAppLayer) (m_LayerMgr.GetLayer("ChatApp"))).SetEnetDstAddress(destination);
176
177             ((SocketLayer) (m_LayerMgr.GetLayer("Socket"))).Receive();
178         }
179         else if(action.equals("Reset")) {
180             Setting_Button.setText("Setting");
181             srcAddress.setText("");
182             dstAddress.setText("");
183             srcAddress.setEnabled(true);
184             dstAddress.setEnabled(true);
185
186             debugMSG("포트 세팅값 초기화");
187
188             ((SocketLayer) (m_LayerMgr.GetLayer("Socket"))).setClientPort(0);
189             ((SocketLayer) (m_LayerMgr.GetLayer("Socket"))).setServerPort(0);
190             ((ChatAppLayer) (m_LayerMgr.GetLayer("ChatApp"))).SetEnetSrcAddress(0);
191             ((ChatAppLayer) (m_LayerMgr.GetLayer("ChatApp"))).SetEnetDstAddress(0);
192         }
193         else if(action.equals("Send")) {
194             if(Setting_Button.getText().equals("Reset") && (srcAddress.getText() != null || dstAddress.getText() != null)) {
195                 String message = ChattingWrite.getText();
196                 ChattingWrite.setText("");
197                 ChattingArea.append("[SEND]:" + message + "\n");
198
199                 debugMSG("메시지 전송 : " + message);
200
201                 byte[] messageBytes = message.getBytes();
202                 ((ChatAppLayer) (m_LayerMgr.GetLayer("ChatApp"))).Send(messageBytes, messageBytes.length);
203             }
204             else {
205                 ChattingArea.append("주소 설정 오류\n");
206             }
207         }
208     }
209 }
```

actionPerformed함수는 GUI상에서 버튼이 눌리거나 하는 ActionEvent가 발생하면 그에 따라 기능을 수행하는 역할을 한다. IPCDlg에서는 Setting, Reset, Send 3가지 기능을 구현해야한다. 따라서 인자로 받아진 ActionEvent의 텍스트를 얻어와서 비교하는 방식을 사용했다. 텍스트가 Setting인 경우 버튼의 텍스트를 Reset으로 변경하고, srcAddress와 dstAddress를 수정하지 못하도록 setEnable(false)를 해주었다. 그리고 그 텍스트를 정수로 변환하여 SocketLayer와 ChatAppLayer의 클라이언트와 서버 포트로 설정해주고 SocketLayer의 Receive함수를 통해 Thread를 동작시켜준다.

만약 텍스트가 Reset인 경우 버턴의 텍스트를 Setting으로 변경하고 srcAddress와 dstAddress를 공백으로, 다시 설정할 수 있도록 setEnable(true)를 해준다. 설정했던 SocketLayer와 ChatAppLayer의 클라이언트와 서버 포트도 0으로 초기화해준다. 텍스트가 Send인 경우에는 일단 메시지를 전송할 수 있는 상황인지 확인한다. Setting_Button의 텍스트가 Reset이고 srcAddress와 dstAddress중 빈 것이 없다면 정상적으로 포트가 설정된 것이므로 Send버튼의 동작을 정의한다. ChattingWrite에 쓰여진 텍스트를 저장하고 그를 비워준다. 또 채팅 영역에 [Send]:와 함께 표시되도록 해준다. 그리고 그 메시지를 getBytes함수를 통해 바이트배열로 만들어주고 이를 ChatAppLayer의 send함수를 통해 SocketLayer로 전송해준다. 만약 정상적으로 포트가 설정되지 않았다면, 채팅 영역에 “주소 설정 오류” 라는 텍스트를 출력하도록 한다. Receive함수는 IPCDlg의 아래 레이어인 ChatAppLayer에서 호출되어 바이트배열을 전달받는데 이는 demultiplexing되어있어 문자열로 바꿔주면 바로 표시될 수 있다. 그래서 이를 new String(input)을 통해 문자열로 만들어주고 이를 채팅 영역에 표시해준다.

ChatAppLayer.java

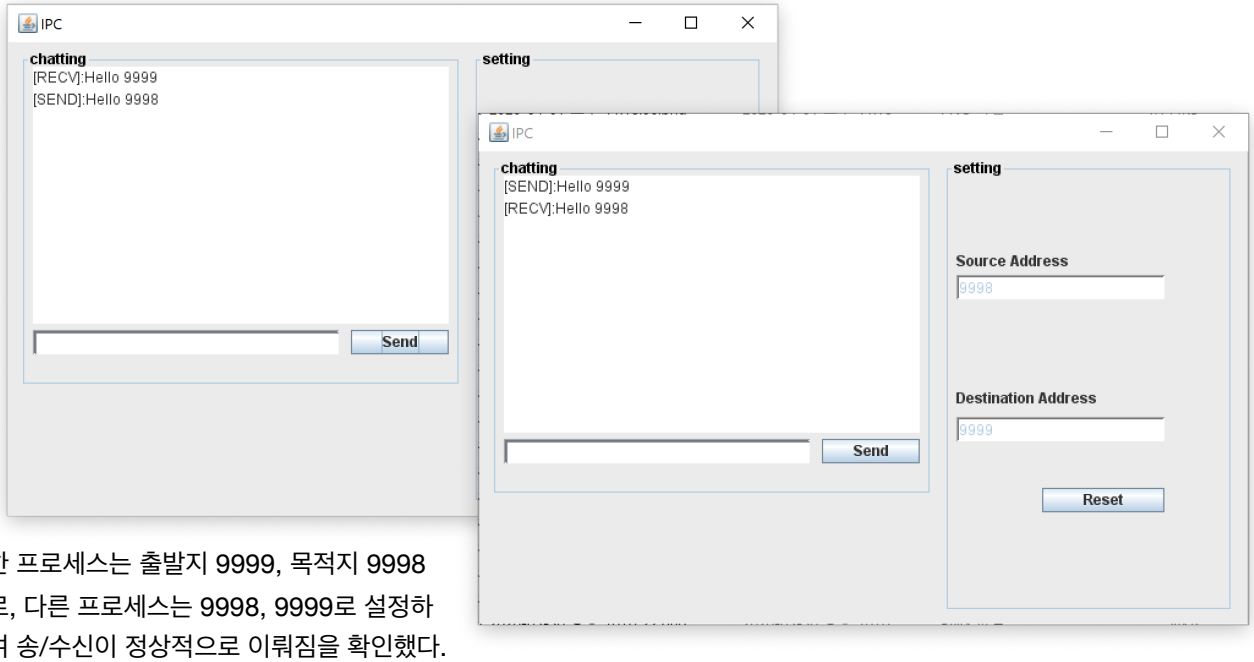
```
211    public boolean Receive(byte[] input) {
212        ChattingArea.append("[RECV]:" + new String(input) + "\n");
213        return true;
214    }
```

ChatAppLayer에서 Send함수는 IPCDlg에서 받은 바이트 배열을 objToByte함수를 통해 헤더와 바이트 배열을 합해(encapsulation) SocketLayer로 전달하는 역할을 한다. SetEnetSrcAddress와 DstAddress함수가 소스와 도착지 포트를 받아서 ChatAppLayer의 헤더인 _CAPP_HEADER, m_sHeader에 포트 정보를 입력하는 역할을 한고, 이는 objToByte함수를 통해 IPCDlg로부터 전달받은 바이트 배열에 합쳐진다. objToByte함수를 통해 encapsulation되면 헤더로 인해 크기가 10 증가한다.

```
80     public byte[] RemoveCappHeader(byte[] input, int length) {
81         byte[] decodedBytes = new byte[length - 10];                    //인코딩된 정보를 디코딩할 때 앞 10개는 추가된 데이터이므로 제거해준다.
82         for(int i = 10; i < length; i ++) {
83             decodedBytes[i - 10] = input[i];
84         }
85         return decodedBytes;
86     }
```

RemoveCapHeader는 SocketLayer로부터 받은 바이트 배열에서 헤더를 제거(demultiplexing)하는 과정이다. encapsulation될 때 크기 10의 헤더가 메시지 바이트의 앞에 붙기 때문에, 이 10 바이트를 제외한 나머지를 반환해준다.

실행 결과



한 프로세스는 출발지 9999, 목적지 9998로, 다른 프로세스는 9998, 9999로 설정하여 송/수신이 정상적으로 이뤄짐을 확인했다.

References

- **Quara : https://www.quora.com/Where-do-network-sockets-go-in-the-OSI-model**
- **티스토리 블로그 : https://blockdmask.tistory.com/169**