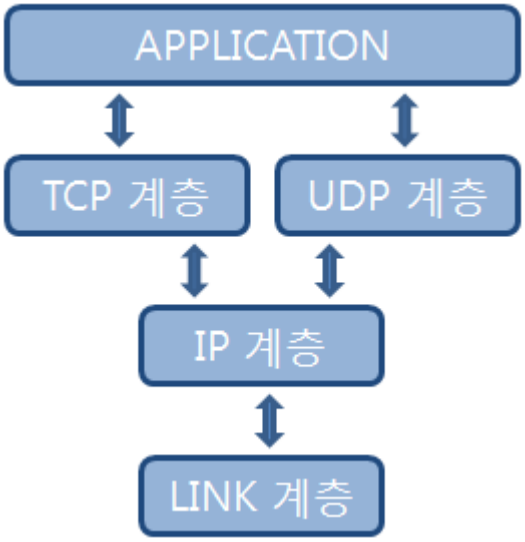
데이터통신 00분반

201402447 한원희

실습 목적

IPC는 Inter-Process Communication의 약자로 "프로세스 사이의 통신"을 의미합니다. 통신이란 기본적으로 데이터를 주고 받는 행위이므로 이 실습은 IPC를 구현하며 어떤 식으로 통신이 이루어지는지 알아보는 과정입니다.

프로토콜 스택



TCP/IP 는 4계층으로 나뉩니다.

LINK 계층

LINK 계층은 물리적인 영역의 표준화에 대한 결과로 가장 기본이 되는 영역입니다.

LAN 같은 네트워크 표준과 관련된 프로토콜을 정의하는 역역이비다.

두 호스트가 인터넷을 통해 데이터를 주고 받을 때 물리적인 연결이 있어야 하므로 이 부분에 대한 표준이 LINK 계층에서 담당합니다.

IP 계층

IP 계층은 목적지로 데이터를 전송하기 위해 어떤 경로를 거쳐갈 것인지를 해결해 줍니다. 하지만 데이터를 전송할 때 마다 거쳐야 할 경로를 선택해 주지만 경로가 일정하지 않습니다.

TCP, UDP 계층

IP에서 알려준 경로를 가지고 데이터의 실제 송수신을 담당하는 계층입니다.

APPLICATION 계층

서버와 클라이언트 사이에 데이터 송수신 규칙이 생기기 마련인데 이것을 설정해 주는 것이 APPLICATION 계층이다.

소켓은 APPLICATION 계층에 속해 있으며 소켓을 기반으로 완성하는 프로토콜 계층입니다. 소켓을 생성하면 앞의 LINK, IP, TCP/UDP 계층에 대한 내용은 감춰집니다.

소켓은 응용 프로그램에서 TCP/IP를 이용하는 창구 역할을 하며 응용 프로그램과 소켓 사이의 인터페이스 역할을 하고 있습니다.

두 소켓이 연결되면 서로 다른 프로세스끼리 데이터를 전달할 수 있습니다. 소켓이 구현됨으로써 네트워크 및 전송 계층의 캡슐화가 가능해집니다.

구현 설명

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    m_LayerMgr.AddLayer(new SocketLayer("Socket"));
    /*
    과제 빈칸 채우기
     ChatApp LayerManager에 추가

     */
    m_LayerMgr.AddLayer(new ChatAppLayer("Chat"));  // 추가
    m_LayerMgr.AddLayer(new IPCDlg("GUI"));

    /*
     과제 ChatApp 연결하기 아래부분 수정

     */
    m_LayerMgr.ConnectLayers(" Socket ( *Chat ( *GUI ) ) ");    // 수정
}
```

Layer를 연결하는 부분입니다. 먼저 Chat을 LayerManager에 추가해주는 부분과 서로의 Layer를 연결해주는 부분입니다.

```
class setAddressListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == Setting_Button) {
            if(Setting_Button.getText() == "Reset") {

                srcAddress.setText("");
                dstAddress.setText("");
                dstAddress.setEnabled(true);
                srcAddress.setEnabled(true);
                Setting_Button.setText("Setting");
            }
            else {
                String Ssrc = srcAddress.getText();
                String Sdst = dstAddress.getText();
                int src = Integer.parseInt(Ssrc);
                int dst = Integer.parseInt(Sdst);
                ((SocketLayer) m_LayerMgr.GetLayer("Socket")).setClientPort(dst);
                ((SocketLayer) m_LayerMgr.GetLayer("Socket")).setServerPort(src);
                ((ChatAppLayer) m_LayerMgr.GetLayer("Chat")).SetEnetSrcAddress(src);
                ((ChatAppLayer) m_LayerMgr.GetLayer("Chat")).SetEnetDstAddress(dst);
                ((SocketLayer) m_LayerMgr.GetLayer("Socket")).Receive();
                Setting_Button.setText("Reset");
                dstAddress.setEnabled(false);
                srcAddress.setEnabled(false);
```

Setting 버튼이 눌렸을 때의 부분입니다.

if부분은 reset 버튼이 활성화 되어 있을 때이고 else 부분은 처음 setting 때 입니다. Src와 Dst를 각각 ChatApplayer의 header 부분에 저장하는 행동과 Src를 서버 포트에 Dst를 소켓의 클라이언트어 저장하는 행동을 취합니다.

```
private class _CAPP_HEADER {
    int capp_src;
    int capp_dst;
    byte[] capp_totlen;
    byte[] capp_data;

    public _CAPP_HEADER() {
        this.capp_src = 0x00000000;
        this.capp_dst = 0x00000000;
        this.capp_totlen = new byte[2];
        this.capp_data = null;
    }
}
```

ChatApplayer의 header 부분입니다. Src와 dst, 총 길이와 데이터가 들어가 있습니다.

```
public byte[] ObjToByte(_CAPP_HEADER Header, byte[] input, int length) {
    byte[] buf = new byte[length +10];
    byte[] srctemp = intToByte4(Header.capp_src);
    byte[] dsttemp = intToByte4(Header.capp_dst);

    buf[0] = dsttemp[0];
    buf[1] = dsttemp[1];
    buf[2] = dsttemp[2];
    buf[3] = dsttemp[3];

    buf[4] = srctemp[0];
    buf[5] = srctemp[1];
    buf[6] = srctemp[2];
    buf[7] = srctemp[3];


    buf[8] = (byte) (length % 256);
    buf[9] = (byte) (length / 256);

    for (int i = 0; i < length; i++)
        buf[10 + i] = input[i];

    return buf;
}
```

헤더의 정보와 받은 채팅(input)을 byte로 변환하는 메소드입니다.

```
public boolean Send(byte[] input, int length) {

    byte[] bytes = ObjToByte(m_sHeader, input, length);
    this.GetUnderLayer().Send(bytes, length + 10);

    return true;
}

public byte[] RemoveCappHeader(byte[] input, int length) {

    for(int i = 0; i < (input.length - 10); i++) {
        input[i] = input[i+10];
    }

    return input;
}
```

ChatAppLayer의 채팅을 보낼때와(send) 받았을때의 행동입니다 (RemoveCappHeader).

보낼때는 byte로 바꾸어 주고 받았을 때에는 Header를 제거한 다음 채팅의 내용만 빼서 input에 저장해줍니다.

```
    public boolean Receive(byte[] input) {
/*
 *  과제 채팅 화면에 채팅 보여주기
 *
 */
        Text = new String(input);
        ChattingArea.append("[RECV] :" + Text + "\n");
        return true;
    }
```

채팅의 내용을 화면에 보여주는 메소드입니다. Header는 이미 제거된 후 input에 채팅의 내용을 저장해 주므로 그것을 단순히 화면에 append해주면 됩니다.

실행 결과