[시뮬레이터]

작성자 김현수 (gustn9515@naver.com)

# 1. 서론

네트워크는 Net+Work의 합성어로 컴퓨터들이 통신 기술을 이용하여 그물망처럼 연결된 통신 이용 형태를 말한다[1]. 최근 네트워크(Network)의 발달로 일반인들도 쉽게 접근이 가능한 형태로 발전했다. 초창기 네트워크는 언제 어디에서나 통신을 가능하게 만들기 위해 인터넷(Internet) 메일 즉 이메일(e-mail)을 사용하게 되었다.

이메일은 1971년 미국의 컴퓨터 엔지니어, 레이 톰린슨이 개발한 프로그램이다[2]. 이메일은 프로그램을 통하여 고유 아이디, 중간에 @기호와 도메인 서버 이름만 작성한다면 상대방과 통신이 가능했다. 하지만 시대가 변화하고 PC(Personal Computer) 보급이 급격하게 늘어나 인터넷 사용자가 점점 많아졌다. 이메일은 인터넷 사용자가 급격하게 증가함에 따라 사용자가 늘어나기 시작했다. 사용자가 급격하게 증가함에 따라 네이버나 구글 등 다양한 웹 페이지들이 등장하며 PC에 있는 프로그램을 사용하지 않아도 웹 페이지에서 편리하게 누구나 사용할 수 있도록 상용화 되었다.

이메일은 처음, 메시지를 전송할 목적으로 만들어졌다[3]. 시간이 지나 모바일 장치가 나와 사용자의 요구 및 불만들이 점점 생겨나기 시작했다. 그러다 2010년 3월 18일에 모바일(Mobile) 메신저 프로그램 카카오톡이 만들어졌다[4]. 카카오톡은 모바일 장치에서는 이메일 사용이 불편하다는 단점을 이용하여, 모바일 장치를 사용하는 유저들에게 많은 혜택을 제공하게 되었다.

이메일에 사용되는 대표적인 데이터 전송 방식은 TCP(Transmission Control Protocol), UDP(User Datagram Protocol)가있다. TCP 방식은 1 대 1 통신으로 3-way handshake 방식으로 신뢰성이 높다고 알려져 있다. UDP 방식은 1대 다 통신이 가능한 방식으로 보낸 후 기도하는(send&pray) 방식으로 신뢰성이 낮다라고 알려져 있다.

# 2. 관련 연구

2.1 네트워크

네트워크는 다양한 범위로 나타낸다. 네트워크의 범위로 분류한 구분방식이 다음 그림 1과 같다.

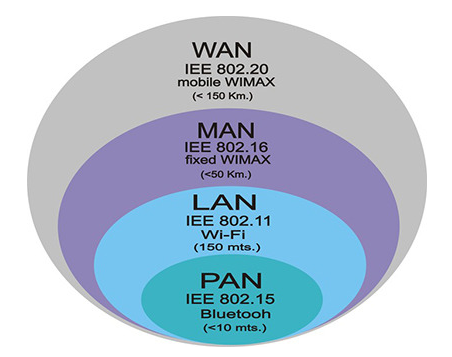


그림 1 네트워크의 종류

작은 순서부터, PAN(Personal Area Network)은 가장 작은 규모의 네트워크이고, LAN (Local Area Network)은 근거리 영역 네트워크이다. MAN(Metropolitan Area Network)는 대도시 영역 네트워크 이며, WAN(Wide Area Network)은 광대역 네트워크이다. 추가적으로 VAN(Value Added Network)는 통신속도와 형식의 변화 또는 통신 경로의 선택 등 여러 종류의 정보서비스가 부가된 통신망이다[1].

무선 네트워크는 특정 주파수를 이용하여 쌍방향 통신을 하거나 단방향 통신이 가능하다. 대표적인 예로 전화기, 무전기, 라디오 등이 있다. 또한 Wi-Fi를 사용한다거나 WAN을 사용하여 외국 서버에 접속을 하는 등 다양한 일을 할 수 있다.

2.2 OSI(Open Systems Interconnection Reference) 7계층 모형

OSI 7계층에는 순차적으로 물리 계층, 데이터 링크 계층, 네트워크 계층, 전송 계층, 세션 계층, 표현 계층, 응용 계층이 있다. 대표적으로 OSI 7 계층은 5~7 단계의 구분이 모호하여 최근에는 4단계로 구분하여 네트워크 계층을 설명하는 TCP/IP 모델이 주로 사용되고 있다. 그림2는 OSI 7계층과 TCP/IP 4계층을 도식화한 모습이다.

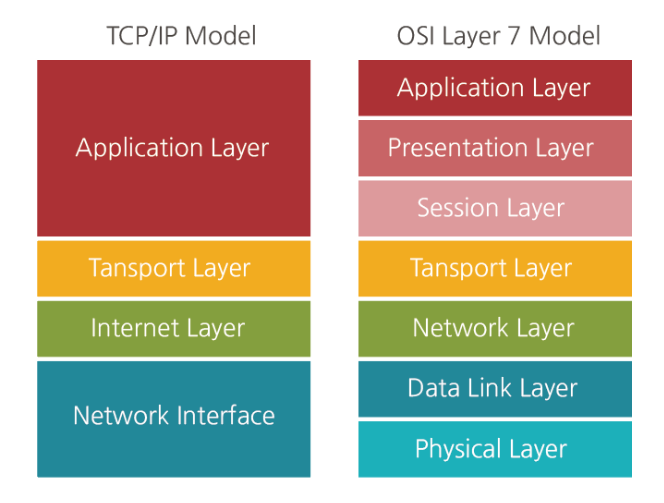


그림 2. OSI 7계층과 TCP/IP 4계층 비교

# 3. 환경 정보

# 4. 네트워크를 통한 수신 및 송신 기능

4.1 전체 형상

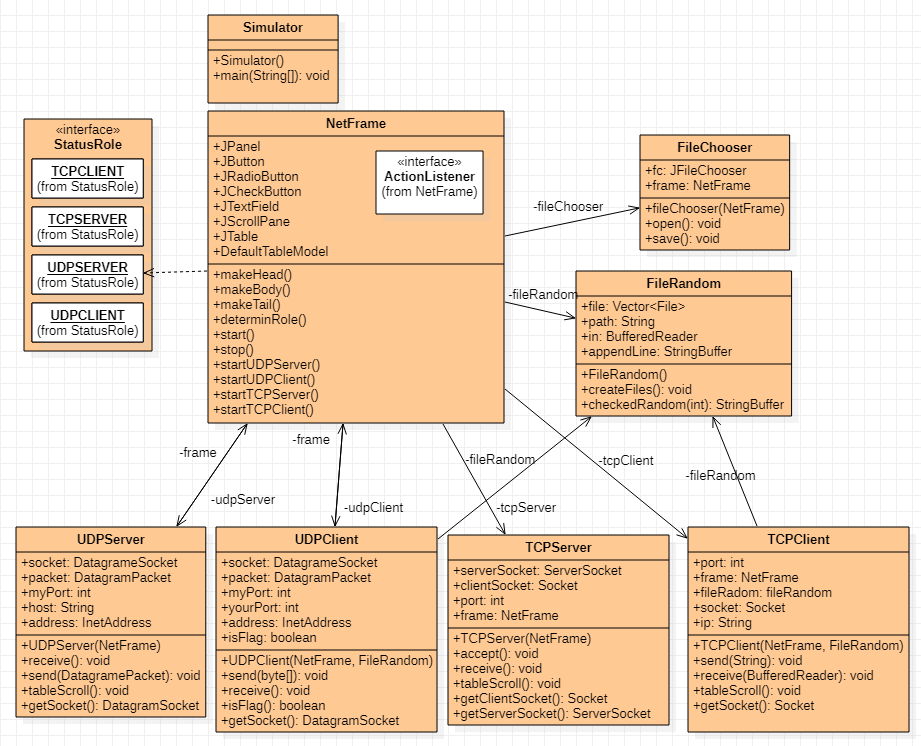


그림3. 시뮬레이터 다이어그램(Diagram)

그림3에서 보는 바와 같이 NetFrame 클래스(Class)는 총 관리자와 동시에 모든 GUI를 담당하고 있다. GUI는 상, 중, 하로 나누어 진다. 상은 makeHead 함수(Method)를 호출하며, JCheckBox, JButton, JTextField, JRadioButton을 이용하여 프로토콜과 기능 및 포트번호, IP주소를 결정하는 역할을 한다. 중앙은 makeBody 함수를 호출하여, JTable을 이용하여 수신한 데이터를 직관적으로 알 수 있도록 만들고, 버튼들을 이용해 파일을 저장 및 불러오기 등의 기능을 수행하는 역할을 한다. 하단부는 makeTail 함수를 호출하여, JTextArea에서 문자를 입력할 수 있으며, 송신 버튼, JSpinner를 이용하여 데이터를 얼마 동안 SLEEP하고, 몇 개의 데이터를 보낼 것 인가를 결정하는 역할을 한다.

NetFrame에서 사용자가 지정한 프로토콜과 기능들을 설정한다. 설정된 프로토콜과 기능을Start버튼을 누를 때, determinRole함수를 호출한다. determinRole 함수에서는 StatusRole 인터페이스의 UDPSERVER, UDPCLEINT, TCPSERVER, TCPCLENT 중 어떤 것인지 판단한다. 판단된 StatusRole을 통해 startUDP, TCPServer, Client 함수를 실행한다.

클래스들은 각각 생성자(Constructor)를 보유하고 있다. 각 생성자에서는 소켓을 초기화 하거나, 포트 번호, IP주소를 초기화 하는 등 데이터를 생성하고, 초기화하는 작업을 한다.

4.2 NetFrame 클래스

|  |
| --- |
| public NetFrame() {  //Frame의 설정  initFrame();  //Frame의 상단부분  makeHeadPanel();  //Frame의 중단부분  makeBodyPanel();  //Frame의 하단부분  makeTailPanel();  setVisible(true);  } |

그림 4. NetFram 클래스의 생성자

NetFrame 클래스는 호출 시 반드시 그림 4의 구조를 호출한다. 첫 번째, initFrame 함수를 호출해 Frame의 사이즈, 타이틀(title), 레이아웃(Layout)의 설정, 이 프로그램이 종료 되었을 때 메모리 누수를 방지하는 설정 등 다양한 설정을 한다. 두 번째, makeHeadPanel 함수는 Panel 클래스를 이용해 상단부분의 옵션 기능, Start/Stop 버튼 IP주소, 포트 번호, 디버그 텍스트 필드까지 만들어 최종 결과를 리턴 한다. 세 번째, makeBodyPanel 함수는 Panel 클래스를 이용해 테이블 및 버튼, 메시지를 작성할 수 있는 JTextArea를 만들어 반환한다. 네 번째, makeTailPanel 함수를 호출하여 하단 부분의 Send 버튼, 사용자가 상대방에게 보내고 싶은 데이터의 양, SLEEP, 특정 알고리즘을 이용한 랜덤 체크 박스를 만들어 리턴 한다. 최종적으로 리턴 된 상단, 중단, 하단을 NetFrame 클래스에 추가 하고 버튼들의 상태 정보들을 결정한다.

4.3 UDP 서버

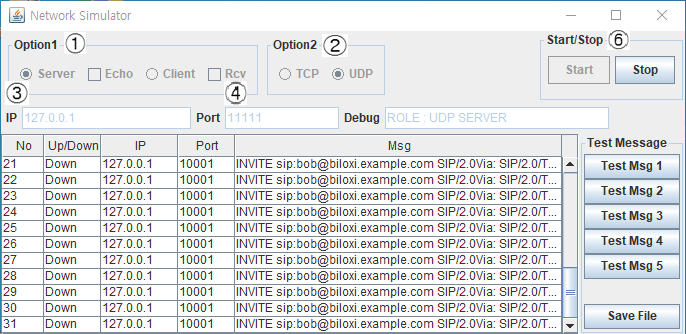


그림 5. UDP서버 데이터를 수신하는 기능

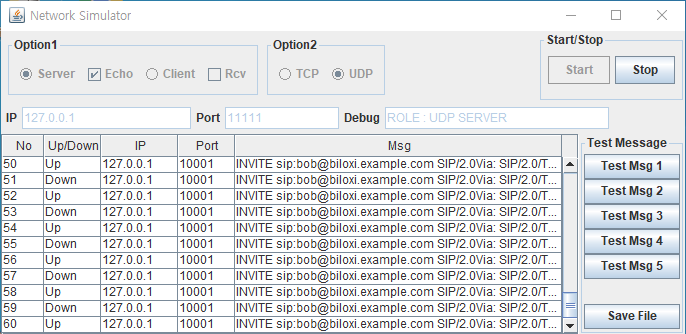


그림 6. UDP서버 수신 및 송신하는 기능

그림 5에서 원형문자로 표기한 순서에 맞추어 순차적으로 진행해야 한다. 첫 번째, 사용자가 서버 또는 클라이언트를 선택하면 각각의 기능(서버 또는 클라이언트)에 따라 서버는 에코서버로 동작을 선택할 수 있고, 클라이언트는 송신기능만 아니라 수신기능도 포함하게 되어 이를 선택할 수 있다. 두 번째, 프로토콜 선택으로 TCP 혹은 UDP를 선택해야만 한다. 세 번째, 자신의 IP번호를 지정한다. 네 번째, 자신이 사용하고 싶은 포트 번호를 지정한다. 이 때 이미 사용되고 있는 포트를 선택하는 경우 운영체제가 거부하고, 또 네트워크 시뮬레이터가 포트번호를 점유하면 다른 프로그램에서 사용할 수 없기 때문에 알려진 포트(well-known port)는 사용을 자제하는 것이 일반적이다[6].

그림5와 그림 6에서 보는 바와 같이 테이블 형태로 데이터를 표시한다. 테이블에서 No는 받은 데이터의 양이며, Up/Down은 클라이언트 측에서 보낸 데이터를 Down, 서버 측에서 보낸 데이터를 Up이라고 표시한다. 테이블에서 IP는 상대방의 IP주소를 뜻하며, Port는 상대방의 포트 번호 이고 Msg는 상대방이 보낸 메시지이다.

그림 6는 그림 5의 설명에서 에코버튼을 누르고 실행하면, 수신한 데이터를 다시 상대방에게 송신을 한다.

|  |
| --- |
| public void receive(){  while(true){  try{  byte[] rcvBuffer = new byte[1024];  //수신하기 위해 패킷을 초기화 한다.  packet = new DatagramPacket(rcvBuffer, rcvBuffer.length);  socket.receive (packet);  //수신한 데이터를 테이블에 추가한다.  //만약 에코버튼이 체크되었다면, 송신한다.  send(packet);  //송신한 데이터를 테이블에 추가한다.  }catch(Exception e){//수신 중 에러(Error) 발생 시 시스템에게 알려줌  System.out.println(e.getMessage()); break;  }  }  }  //수신한 데이터를 다시 송신하는 함수  public void send(DatagramPacket packet){  try{  socket.send(packet);  }catch(Exception e){//송신 중 에러가 발생시 시스템에게 알려준다.  System.out.println(e.getMessage();  }  } |

그림7. UDP 서버 수신 및 송신 하는 구조

그림7는 그림5처럼 원형문자로 표기한 순서에 맞추어 순차적으로 진행한다면, UDPServer 클래스 객체를 생성하게 된다. UDPServer 클래스의 생성자에서는 사용자가 입력한 포트번호와 IP주소를 사용하여 소켓을 초기화한다. 소켓 생성 후 스레드(Thread) 클래스를 사용하여 receive 함수를 호출한다. 이는 스레드를 사용하여 GUI부분이 BLOCKING이 되어 다른 일 처리를 하지 못하게 되는 상황을 방지한다. receive 함수에서는 데이터를 수신하기 위해 패킷을 만든 후 기다린다. 데이터 수신이 되면, 패킷에 저장된다. 그림 5의 에코 버튼을 체크했다면, send함수를 호출한다. send함수는 패킷에 저장되어 있는 데이터를 다시 송신한다. 사용자가 Stop 버튼을 누르면, 개발자가 오류를 쉽게 찾을 수 있도록 콘솔(Console)에 표현하고 소켓에 close 함수를 호출해 장기간 테스트 가능성, 리소스 누수(leak) 방지에 신경 썼다.

4.4 UDP 클라이언트

|  |
| --- |
| public void send(byte[] lineBuffer) {  try {  //포트 번호를 지정한다.  //패킷을 초기화 시킨다.  int count = //사용자가 지정한 Count 값  int sleep = //사용자가 지정한 Sleep 값  // Rcv버튼이 체크되어있다면 수신한다.  receive();  //사용자가 설정해준 count값으로 for문을 실행한다.  for(int i=0; i<count; i++) {  //소켓을 통해 데이터를 보낸다.  socket.send(packet);  //사용자가 지정한 sleep값으로 sleep시킨다.  Thread.*sleep*(sleep);  }  }catch(Exception e) {  System.out.println(“UDPClient Send Fail ”+e.getMessage());  }  }    public void receive() {  while(true) {  try {  byte receiveBuffer[] = new byte[1024];  //데이터를 송신한다.  socket.receive(packet);  }catch(Exception e) {  System.out.println(“UDPClient Receive Fail”+e.getMessage());  break;  }  }  } |

그림 8. UDP 클라이언트 수신 및 송신 구조

그림 8는 사용자가 설정한 포트 번호를 통해 소켓을 초기화 한다. 사용자가 Send 버튼을 누르기 전 사용자가 상대방의 포트 번호를 유동적으로 변경하여 보낼 수 있다. Send 함수는TextArea에 메시지를 작성하여 Send 버튼을 누르게 되면, 스레드 클래스를 이용하여 GUI가 BLOCKING 되지 않도록 도와주어 여러 번 반복적으로 보내는 것도 가능하다. 또 다른 시뮬레이터나, 채팅프로그램 등 다른 프로그램들이 다른 포트 번호를 이용하여 접속한다면, 클라이언트에는 전혀 영향을 미치지 않고, 데이터를 보내는 속도 또한 별다른 차이가 없기 때문에, 멀티 기능이 가능하다.

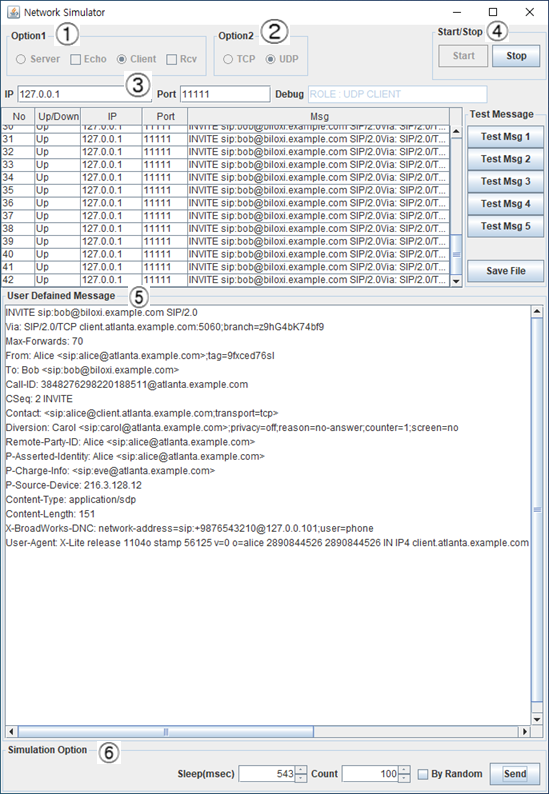


그림 9. UDP 클라이언트 수신하는 기능

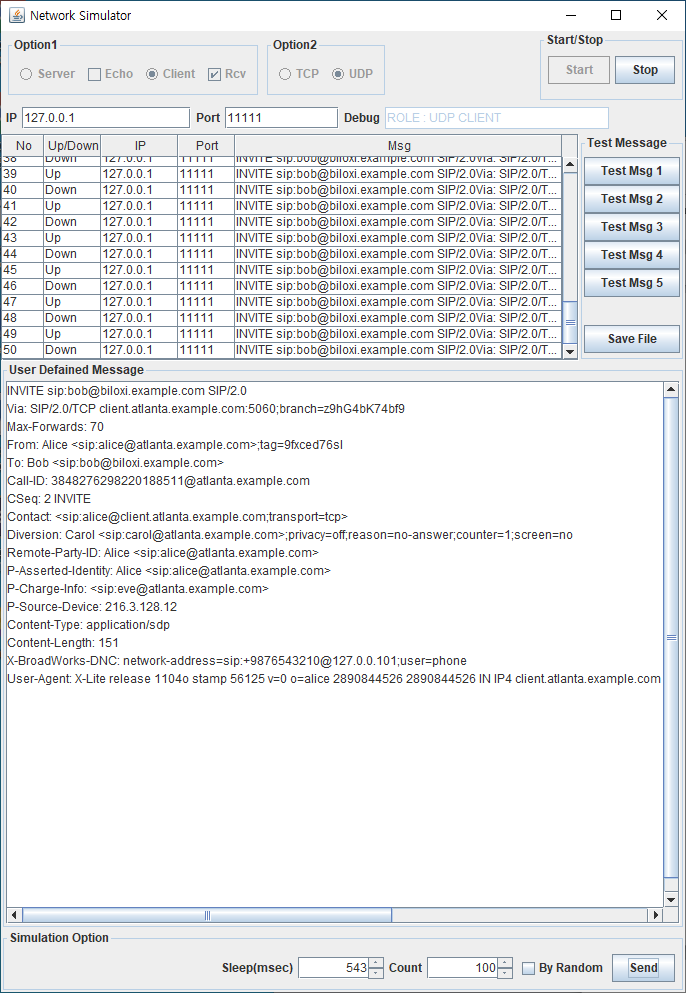


그림 10. UDP 클라이언트 수신 및 송신 기능

그림9에서는 원형문자로 표기한 순서대로 진행한다. 첫 번째부터 네 번째 까지는 그림 5의 설명과 유사하다. 단, 세 번째의 경우 사용자가 설정한 포트번호를 이용하여 소켓을 만든 후, 상대 포트번호를 다시 설정해야 한다. 다섯 번째, 사용자가 전송할 메시지를 작성한다. 현재는 테스트를 위해 SIP INVITE MESSAGE를 전송 중이다[7]. 여섯 번째, 사용자가 보내고자 하는 메시지를 몇 번을 보낼 것인지 얼마만큼 SLEEP 시킬 것인지를 결정할 수 있다. 이는 수천 번, 수십만 번 이상의 테스트를 간단 한 설정을 통해 진행할 수 있고, 전송 간격도 사용자에 의해 미세 결정이 가능하여 다양한 형태의 프로그램의 성능이나 기능 평가에 도움이 된다. Send 버튼을 누른다면, GUI의 BLOCKING을 막기 위해 스레드 클래스를 이용하여 send 함수를 호출한다. 사용자가 Stop 버튼을 누른다면, 소켓에 close 함수를 호출한다. 호출 된 close함수는 재가용성, 장기간 테스트 가능성, 리소스 누수(leak) 방지에 도움이 된다. By Random 체크버튼은 특수한 알고리즘을 통해 체크 시 개발자가 만들어 둔 텍스트 파일들의 내용을 불러와 시시 각각 변화를 주며 상대방에게 송신한다.

4.5 TCP 서버 수신 및 에코 기능

|  |
| --- |
| public void accept() {  while(true) {  try {  clientSocket = serverSocket.accept();  Thread thread = new Thread() {  public void run() {  //수신  }  };  thread.start();  }catch(Exception e) {  break;  }  }  } |

그림 11. TCP ACCEPT 구조

TCP는 1대 1 통신 방식으로 상대방이 접속을 하면, SYN(연결 요청) 메시지를 보낸다. 연결 요청을 받은 수신자는 다시 ACK(응답) 메시지를 보낸다. 이런 방식을 3-wayhandshake 라고 부른다[8]. 그림11은 TCP 방식을 사용하고 있다. 소켓은 상대방이 접속할 때까지 ACCEPT 상태를 유지하고 있다.

상대방이 접속에 성공하기 전 까지 ACCEPT 상태를 유지하고 있기 때문에 GUI가 BLOCKING 되는 현상과 한 사용자뿐만 아니라 다수 사용자에게 멀티기능을 제공하고자 한다면, 스레드 클래스를 사용해야만 한다. 단, 스레드 클래스 사용시 주의점은 스레드 클래스는 모든 자원을 공유함으로써 멀티 접속이 아닌 단일 접속이 될 가능성이 있다.

|  |
| --- |
| public void receive(){  try{  //데이터를 송신할 BufferedReader 생성  //데이터를 수신하는 PrintWriter 생성  while(데이터 수신){  if(소켓이 종료되었는지){  //BufferedReader close();  //PrintWriter close();  break;  }  //데이터 수신 시 테이블에 추가  if(echo 버튼이 선택되었는지)  //데이터를 재송신  }  catch(Exception e){}  } |

그림 12. TCP 서버 수신 및 송신

그림 12은 상대방이 CONNECT하면, 운영체제가 판단하여 사용하지 않는 포트번호를 결정해준다. 데이터는 유니코드(2BYTE)로 송수신하기 때문에, C언어와는 다르게 자바에서는 Reader, Writer 클래스를 사용하는 것이 일반적이다. BufferedReader 클래스가 초기화 되면WHILE 문을 통해 데이터가 있을 때까지 수신하게 된다. 데이터 수신 시 상대방이 DISCONNECT 했을 경우, BufferedReader와 PrintWriter의 재 가용성과 메모리 누수를 방지하기 위해 close 함수를 호출한다. 기능 중 에코 버튼이 체크 되어 있다면, Writer 클래스를 이용해 데이터를 재전송한다.

4.6 TCP 클라이언트 송신 및 수신 기능

|  |
| --- |
| public void send(String line){  try{  //소켓을 초기화 한다.  //line이 null이거나, 비어있는지 판별한다.  //개행 표시가 있을 경우, 데이터를 쪼개어 개행 표시 부분 제거 후 합친다.  String buffer = “”;  for(String text: line.split(“\n”)){  buffer += text;  }  //데이터를 송신할 BufferedReader 생성  //데이터를 수신하는 PrintWriter 생성  //사용자가 지정한 Count 값, Sleep 값을 가져온다.  if(Rcv 버튼이 체크되어있는지){  //데이터 수신  }  //데이터를 테이블에 추가 후 송신한다.  }catch(Exception e){}  } |

그림 13 TCP 클라이언트 수신 및 송신

# 5. 결론 및 향후 연구

[1] “[Web]네트워크란 무엇인가? 네트워크의 정의와 종류 총정리”, [https://coding-factory.tistory.com/340 2019-11](https://coding-factory.tistory.com/340%202019-11)

[2]”[국제]인류 최초의 이메일에 적힌 내용은? QWERTYUIOP | YTN”, [https://www.ytn.co.kr/\_ln/0104\_201604061654007339 2019-11](https://www.ytn.co.kr/_ln/0104_201604061654007339%202019-11)

[3]”전자우편 – 위키백과, 우리 모두의 백과사전” <https://namu.wiki/w/%EC%9D%B4%EB%A9%94%EC%9D%BC> 2019-11

[4]”카카오톡 – 위키백과, 우리 모두의 백과사전” <https://ko.wikipedia.org/wiki/%EC%B9%B4%EC%B9%B4%EC%98%A4%ED%86%A1> 2019-11

[5]”OSI 모형 – 위키백과, 우리 모두의 백과사전”, <https://ko.wikipedia.org/wiki/OSI_%EB%AA%A8%ED%98%95> 2019-11

[6]”TCP/UDP의 잘 알려진 서비스 포트 번호 목록 – well known port”, [https://shaeod.tistory.com/389 2019-11](https://shaeod.tistory.com/389%202019-11)

[7]”SIP method on RFC 3261 – 탱이의 잡동사니”, [http://wiki.pchero21.com/wiki/SIP\_method\_on\_RFC\_3261 2019-12](http://wiki.pchero21.com/wiki/SIP_method_on_RFC_3261%202019-12)

[8]”[Network] TCP 프로토콜 이란? – 공대냥이 - 티스토리”, <https://itragdoll.tistory.com/57>