進階電腦網路

TCP Socket – 視窗版 線上聊天與圖片傳輸

心得報告

資工二

S0954010 謝宥宣

2022/05/26

目錄

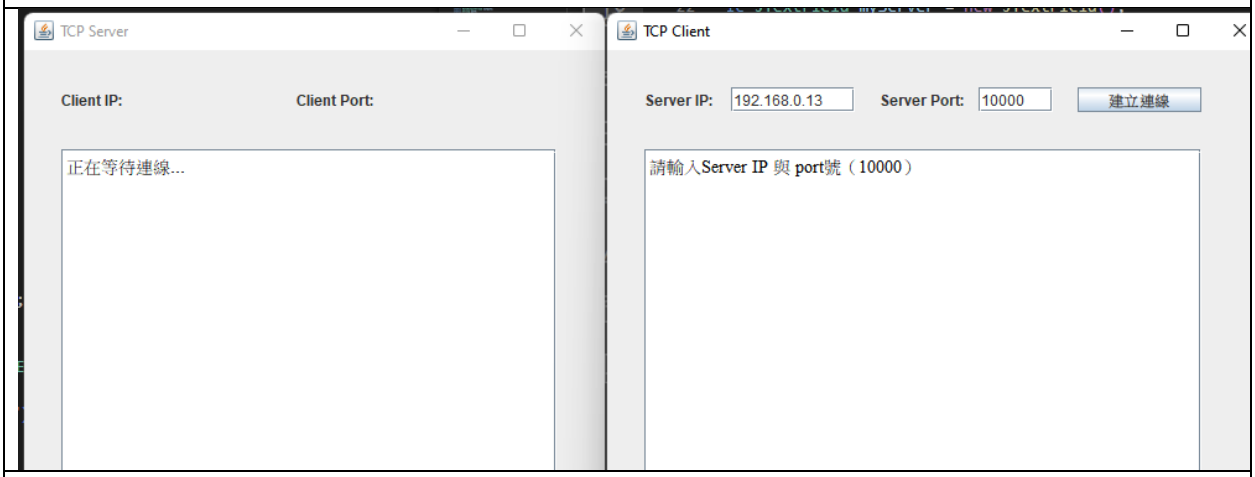
程式功能說明

- 本專題包含兩個 Java 程式，一個 Server 程式與一個 Client 程式。
- 本專題可透過 TCP 傳輸，在兩裝置間雙向傳輸訊息與圖片。
- 先執行 chat-ftps 程式，將會以兩個 server socket 等 Client 端連線（Port 10000 用以傳送文字訊息，Port 10001 用以傳送圖片檔案）
- 執行 chat-ftpc 程式後，輸入 server IP address 與 port 號（10000）後按下連線，client 端將傳送連接請求，成功後將於聊天區域顯示成功連線訊息，且 server 端亦會同步更新目前連線階段訊息。
- 連線成功後，於 server 端與 client 端上方將分別顯示 client IP address、 port 號與 server IP address，port 號。
- 程式除 main 執行緒外，各多分為 3 個不同執行緒，分別用來接收文字、接收圖片與傳送文字/圖片。其中接收文字、圖片的執行緒以 while 迴圈不停等待接收訊息，直到程式停止。
- 連線建立後即可開始聊天。聊天文字可於視窗下方 JTextPane 輸入，輸入完成後點擊傳送即可傳送出去。
- 若是要傳送圖片，點擊圖片按鈕後即可開啟檔案選擇起，選擇要傳送的圖檔。
- 程式有額外設計限制選擇傳輸檔案的類型，因設計只可傳輸圖檔，因此只接受 *.GIF,*.PNG,*.JPG, *.JPEG 之檔案。（額外功能）
- 選擇完要傳送的檔案後，將在按鈕下方顯示圖檔的檔案名稱。（額外功能）
- 圖檔傳輸後，將另存於本地與程式同路徑上，程式結束執行後仍可保留。（額外功能）

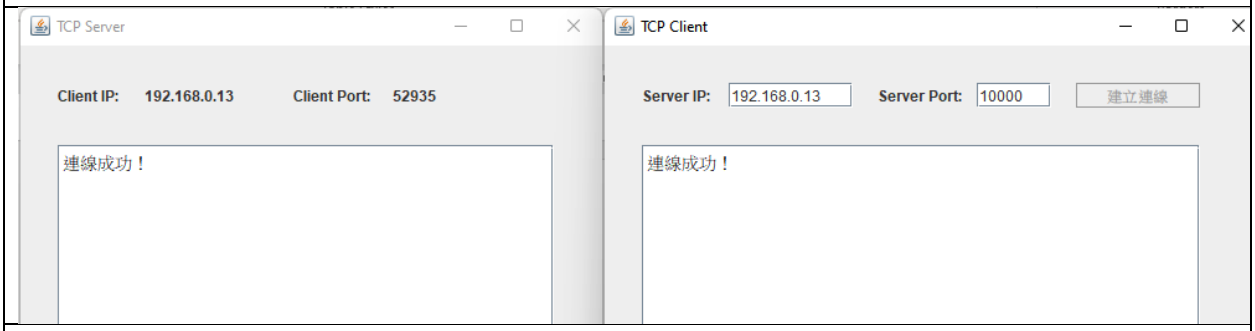
- 聊天時，聊天室視窗將自動滑動至視窗底部，以確保最新訊息能被顯示出來，不需要另外滑動滑鼠。（額外功能）
- Client 端按下結束後，Client 端將傳送”EOF”給 Server 端，爾後關閉與 Server 端的連線並關閉視窗。
- Server 端在接收到 Client 端的連線結束訊息後，將回復程式至初始狀態，待下一次 Client 的連線請求。

程式流程說明

Server 端開啟，等待連線請求， Client 端輸入 Server IP address 與 Port 號後，點擊按鈕建立連線。



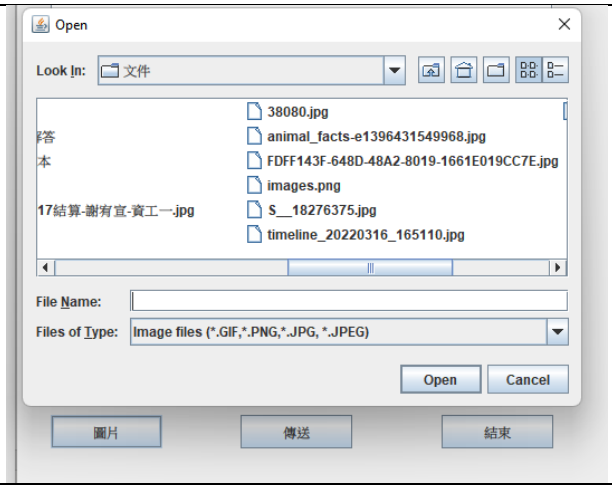
連線成功後，兩端將顯示對方的資訊，下方聊天區將顯示連線成功訊息。



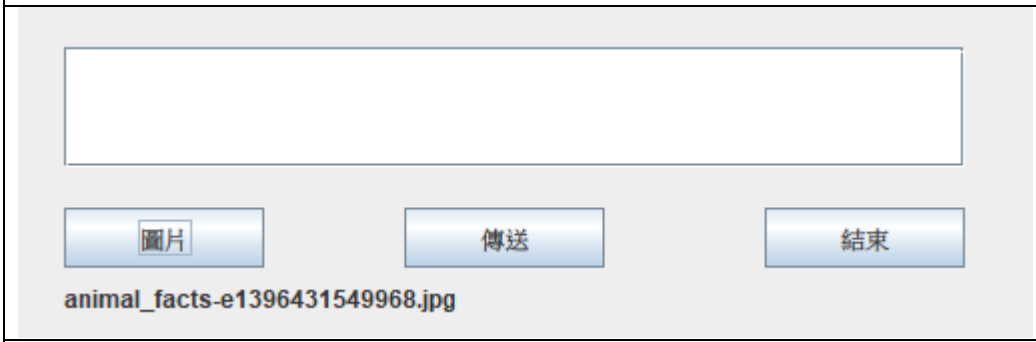
連線建立後，即可於下方文字區輸入要傳送的訊息。完成後按下傳送將傳出訊息。

傳輸與接收到的訊息都將顯示在兩端聊天室中

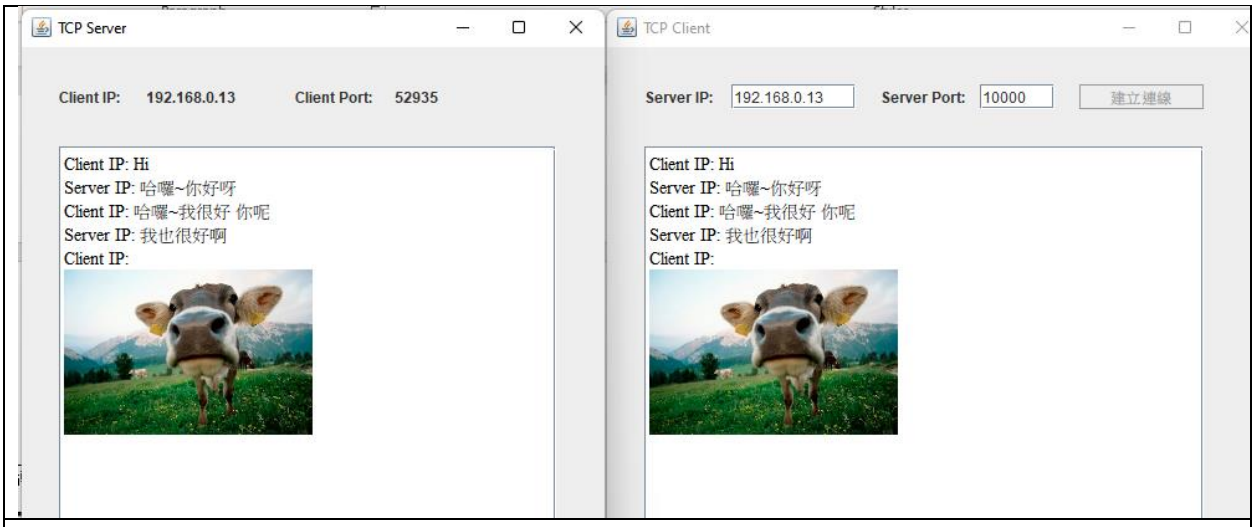
若要傳送圖檔，點擊圖片按鈕即可選擇檔案。



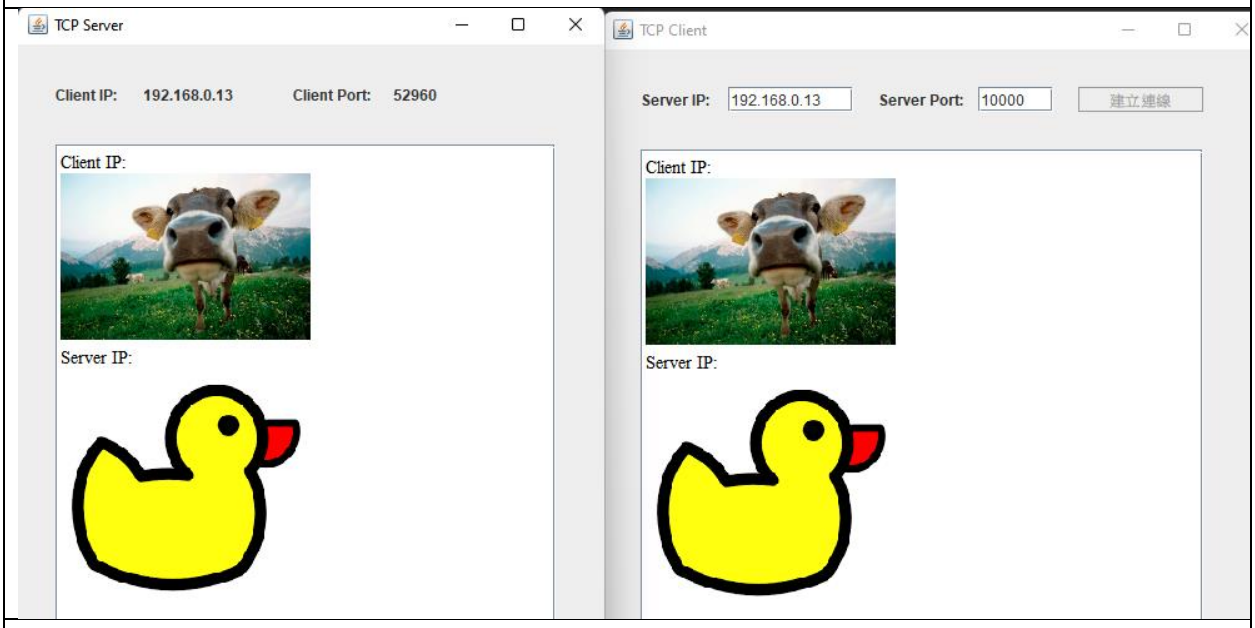
選擇後，選擇之圖片名稱將顯示於圖片按鈕下方。



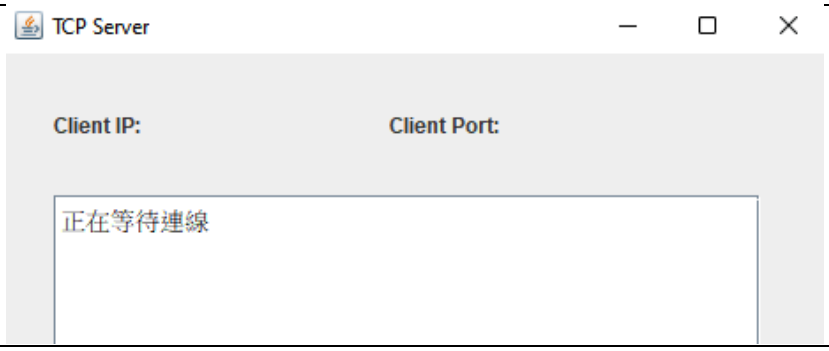
點擊傳送後，將傳送圖片給對方。



如同文字傳輸，圖片傳輸亦為雙向。

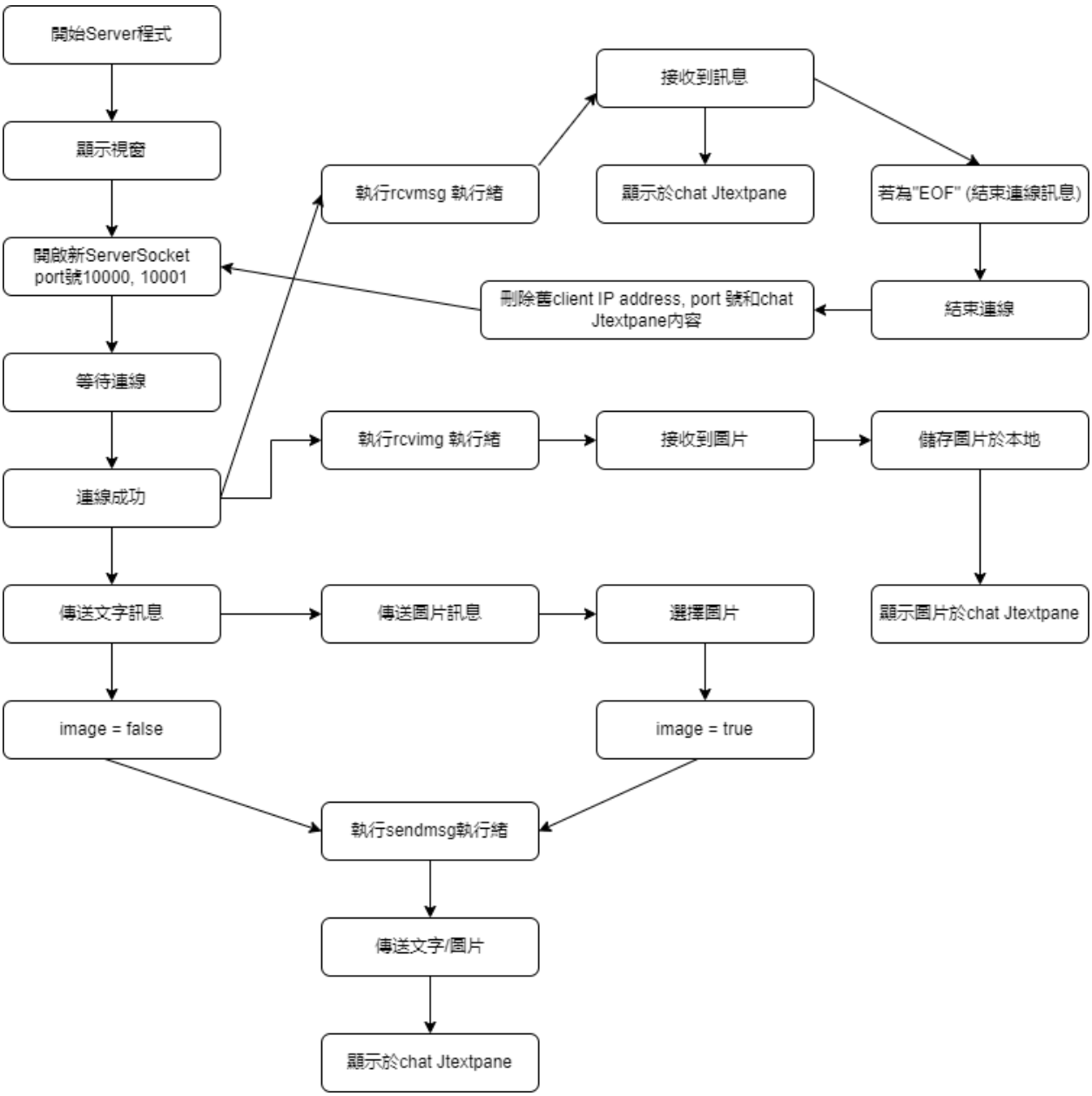


若要結束連線，點擊下方結束按鈕即可關閉 Client 端視窗，Server 端亦將恢復至等待連線階段。

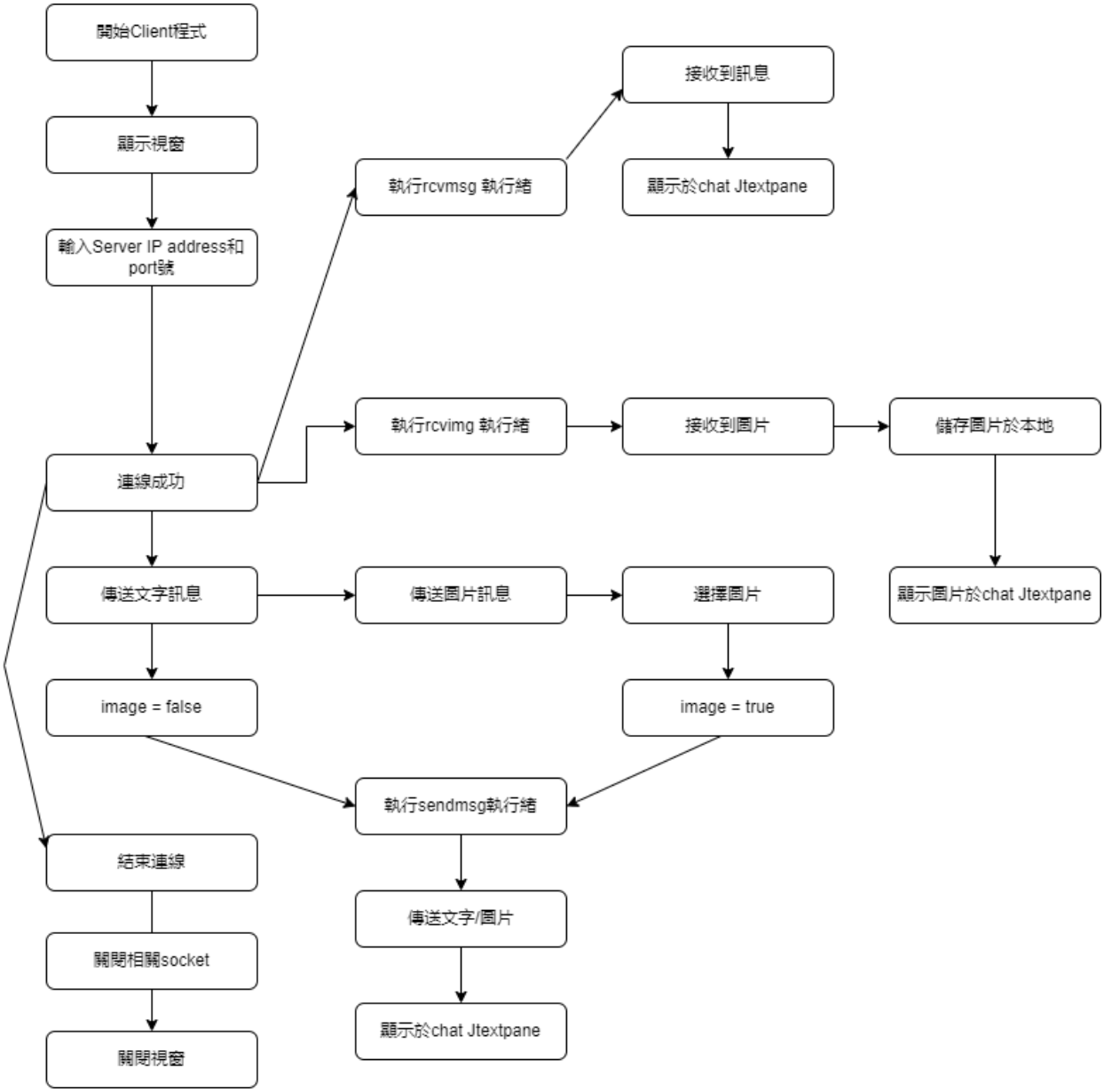


程式流程圖

Server 端



Client 端


開始Client程式
顯示視窗
輸入Server IP address和port號
連線成功
執行rcvmsg 執行緒
接收到訊息
顯示於chat Jtextpane
執行rcvimg 執行緒
接收到圖片
儲存圖片於本地
傳送文字訊息
傳送圖片訊息
選擇圖片
顯示圖片於chat Jtextpane
image = false
image = true
執行sendmsg執行緒
結束連線
傳送文字/圖片
關閉相關socket
顯示於chat Jtextpane
關閉視窗

程式碼註解

Server 端

```
import javax.swing.JFrame;
import java.awt.*;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import javax.swing.*;
import java.awt.event.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.lang.Thread;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import javax.swing.JFileChooser;
import java.io.OutputStream;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.io.FileOutputStream;

public class tcp_server extends JFrame {

   static JTextPane chat = new JTextPane();
```

```
static JTextPane tosend = new JTextPane();
static JButton pic = new JButton("圖片");
static JButton send = new JButton("傳送");
static String toprint = "";
static DataInputStream input;
static DataOutputStream output;
static DataInputStream input1;
static DataOutputStream output1;
static String path;
static File selectedFile;
static Boolean image = false;
static ServerSocket ss;
static ServerSocket ss1;
static Socket sc;
static Socket sc1;
static String img = "";
static JLabel myClient = new JLabel();
static JLabel myPort = new JLabel();
static JLabel selectedfilelabel = new JLabel();
static int imgcnt = 0;
public static void main(String[] args) throws Exception {

   JFrame frm = new JFrame("TCP Server");
   frm.setLayout(null);
   Container ctp = frm.getContentPane();
   ctp.setLayout(null);
```

```
JLabel Client_label = new JLabel("Client IP: ");    //display connected client's IP address & port num.
JLabel Client_port = new JLabel("Client Port: ");

JScrollPane jsp = new JScrollPane(); //add scrollbar for chat and text area
JScrollPane jsp2 = new JScrollPane();

Client_label.setBounds(30, 30, 60, 20);
myClient.setBounds(100, 30, 100, 20);
Client_port.setBounds(220, 30, 80, 20);
myPort.setBounds(300, 30, 60, 20);
// chat.setBounds(30, 80, 400, 600);
jsp.setBounds(30, 80, 400, 600);
jsp.setViewportView(chat);
jsp2.setBounds(30, 700, 400, 60);
jsp2.setViewportView(tosend);
// tosend.setBounds(30, 700, 400, 60);
pic.setBounds(30, 780, 100, 30);
send.setBounds(160, 780, 100, 30);
pic.addActionListener(new Actlis());
send.addActionListener(new Actlis());
chat.setEditable(false);
chat.setContentType("text/html"); //set to display html in chat textpane
chat.setText("正在等待連線...");
selectedfilelabel.setBounds(30, 810, 300, 30);

ctp.add(Client_label);
ctp.add(myClient);
```

```
ctp.add(Client_port);
ctp.add(myPort);
// ctp.add(chat);
// ctp.add(tosend);
ctp.add(pic);
ctp.add(send);
ctp.add(jsp);
ctp.add(jsp2);
ctp.add(selectedfilelabel);
frm.setSize(485, 900);
frm.setLocation(500, 150);
frm.setVisible(true);

frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
ss = new ServerSocket(10000); //create 2 new serversockets, one for text message and one for image.
ss1 = new ServerSocket(10001);
sc = ss.accept();   //create 2 sockets and wait for client's connection.
sc1 = ss1.accept();
// System.out.println("run");

// System.out.println("success");
String clientIP = sc.getInetAddress().getHostAddress(); //get client's information and display
myClient.setText(clientIP);
String clientport = sc.getRemoteSocketAddress().toString().split(":")[1];
myPort.setText(clientport);

input = new DataInputStream(sc.getInputStream());
```

```
    output = new DataOutputStream(sc.getOutputStream());

    input1 = new DataInputStream(sc1.getInputStream());
    output1 = new DataOutputStream(sc1.getOutputStream());
    new rcvmsg(); //run new thread for receiving message
    new rcvimg(); //run new thread for receiving image
    chat.setText("連線成功！");
  }

  static public class rcvmsg extends Thread { //thread for receiving messages
    public rcvmsg() {
      new Thread(this).start();
    }

    public void run() {
      // System.out.println("can receive");
      while (true) {

        try {
          String str = "";
          str = input.readUTF();
          if (!str.equals("")) {
            if (str.equals("EOF")) { //if client sends EOF, meaning to end connection

              System.out.println("end");
              input.close(); //close everything and restart everything
              input1.close();
```

```
    output.close();
    output1.close();
    sc.close();
    sc1.close();
    toprint = "";
    chat.setText(toprint);
    myClient.setText("");
    myPort.setText("");
    chat.setText("正在等待連線");
    // System.out.println("run");
    Thread.sleep(1000);
    ss = new ServerSocket(10000); //recreate ServerSocket
    ss1 = new ServerSocket(10001);
    sc = ss.accept(); //wait for next connection
    sc1 = ss1.accept();
    String clientIP = sc.getInetAddress().getHostAddress();
    myClient.setText(clientIP);
    String clientport = sc.getRemoteSocketAddress().toString().split(":")[1];
    myPort.setText(clientport);
    input = new DataInputStream(sc.getInputStream());
    output = new DataOutputStream(sc.getOutputStream());
    input1 = new DataInputStream(sc1.getInputStream());
    output1 = new DataOutputStream(sc1.getOutputStream());
    new rcvmsg();
    new rcvimg();
  }
  str = str.replaceAll("\n", "<br>"); //normal text message, replace enter with <br>
```

```
                toprint += "Client IP: " + str + "<br>";
                chat.setText(toprint); //print text message on chat area
                chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area
automatically
              }

            } catch (Exception e) {
              // TODO: handle exception
            }
          }
        }
      }

    static public class rcvimg extends Thread { //Thread for receiving images
      public rcvimg() {
        new Thread(this).start();
      }

      public void run() {
        while (true) {

          try {
            // System.out.println("can rsv img");

            BufferedInputStream in;
            InputStream ins = sc1.getInputStream();
            in = new BufferedInputStream(ins);
            byte[] b = new byte[1024];
```

```
            ByteArrayOutputStream buf = new ByteArrayOutputStream();
            int len;
            while ((len = in.read(b)) > 0) {
               buf.write(b, 0, len); //write to buffer
               // System.out.println("receiving");
            }
            System.out.println("img received");
            OutputStream out = new FileOutputStream(new File("imgs" + imgcnt + ".jpg")); //Save image as file,
filename = imgs + imgcnt + .jpg
            buf.writeTo(out);
            sc1.shutdownInput();
            // in.close();
            out.close();
            in = null;
            String img = "<img src= 'file:imgs" + imgcnt + ".jpg' width = '200'> <br>"; //display image on chat
area using image saved previously
            toprint += "Client IP: <br>" + img;
            imgcnt++; //img counter +1
            chat.setText(toprint); //print image
            chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area automatically
          } catch (Exception e) {
            //System.out.println(e.toString() + "server rcv");
          }
        }
      }
    }

   static public class sendmsg extends Thread { //Thread for sending messages (texts and images)
```

```
    private String msg;

    public sendmsg(String str) {
       msg = str;
       new Thread(this).start();
    }

    public void run() {
       if (image == false) { //if sending text
          try {
             msg = msg.replaceAll("\n", "<br>"); //replace enter with <br>
             if (!msg.equals("")) {
                output.writeUTF(msg); //send message to client
                toprint += "Server IP: " + msg + "<br>";
                chat.setText(toprint); //display message on chat area
                chat.setCaretPosition(chat.getDocument().getLength());  //scroll to bottom of chat area
automatically
                }

          } catch (Exception e) {
             System.out.println("訊息傳送失敗");
          }
       } else { //sending image
          System.out.println("start sending");
          image = false; //restore default image value
          String img = "<img src= 'file:\\" + path + "' width = '200'> <br>"; //display image on chat area using
html syntax
```

```
// toprint += "Server IP: ";
toprint += "Server IP: <br>" + img;
chat.setText(toprint);
chat.setCaretPosition(chat.getDocument().getLength());  //scroll to bottom of chat area automatically
try {

   FileInputStream fis = new FileInputStream(new File(path));
   OutputStream os = sc1.getOutputStream();
   BufferedOutputStream bos = new BufferedOutputStream(os);
   //System.out.println("start sending123");
   byte[] b = new byte[1024];
   int len;

   while ((len = fis.read(b)) != -1) {
      bos.write(b, 0, len);
   }
   System.out.println("sending");
   bos.flush();

   sc1.shutdownOutput(); //must shutdown output, otherwise it won't send
   // bos.close();
   fis.close();
   // bos = null;
   System.out.println("sent");
   selectedfilelabel.setText(""); //restore default Jlabel value
} catch (Exception e) {
   System.out.println(e.toString() + "Server send");
```

```
            }
        }
    }
}

static public class Actlis extends WindowAdapter implements ActionListener // .addActionListener(new Actlis());
{
    public void windowClosing(WindowEvent e) { // close everything if X clicked
        try {
            input.close();
            input1.close();
            output.close();
            output1.close();
            sc.close();
            sc1.close();
        } catch (Exception eee) {
            // TODO: handle exception
        }
        System.exit(0);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == send) { //if send button clicked

            String str = tosend.getText();
            new sendmsg(str); //send message
            tosend.setText(""); //restore texting area
```

```
        }
        if (e.getSource() == pic) { //if sendpic button clicked
          JFileChooser cfile = new JFileChooser();
          cfile.setFileFilter(new FileNameExtensionFilter("Image files (*.GIF,*.PNG,*.JPG, *.JPEG)", "GIF", "PNG",
              "JPG", "JPEG")); //limit selected file type *extra function
          int returnfile = cfile.showOpenDialog(pic);
          if (returnfile == JFileChooser.APPROVE_OPTION) { //check if any file is selected
            selectedFile = cfile.getSelectedFile(); //get selected file
            selectedfilelabel.setText(cfile.getSelectedFile().getName()); //display the name of selected file
            path = selectedFile.getAbsolutePath(); //get path of the file
            image = true;
          } else {
            selectedfilelabel.setText("檔案未選擇");
          }
        }
      }
    }
}
```

Client 端

```
import javax.swing.JFrame;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.net.Socket;
import java.lang.Thread;
import java.net.InetAddress;
import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import javax.swing.JFileChooser;
import java.io.OutputStream;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;

public class tcp_client {
   static JTextField myServer = new JTextField();
   static JTextField myPort = new JTextField();
   static JButton connect = new JButton("建立連線");
   static JTextPane chat = new JTextPane();
```

```
static JTextPane tosend = new JTextPane();
static JButton pic = new JButton("圖片");
static JButton send = new JButton("傳送");
static JButton end = new JButton("結束");
static String toprint = "";
static DataInputStream input;
static DataInputStream input1;
static DataOutputStream output;
static DataOutputStream output1;
static InetAddress server_ip;
static String path;
static File selectedFile;
static Boolean image = false;
static Socket sc;
static Socket sc1;
static JLabel selectedfilelabel = new JLabel();
static int imgcnt = 0;
public static void main(String[] args) throws Exception {
    JFrame frm = new JFrame("TCP Client");
    frm.setLayout(null);
    Container ctp = frm.getContentPane();
    ctp.setLayout(null);

    JLabel Server_label = new JLabel("Server IP: ");
    JLabel Server_port = new JLabel("Server Port: ");
    JScrollPane jsp = new JScrollPane(); //add scrollbar for chat and text area
    JScrollPane jsp2 = new JScrollPane();
```

```
Server_label.setBounds(30, 30, 60, 20);
myServer.setBounds(100, 30, 100, 20);
Server_port.setBounds(220, 30, 80, 20);
myPort.setBounds(300, 30, 60, 20);
connect.setBounds(380, 30, 100, 20);
connect.addActionListener(new Actlis());
pic.addActionListener(new Actlis());
// chat.setBounds(30, 80, 450, 600);
jsp.setBounds(30, 80, 450, 600);
jsp.setViewportView(chat);
jsp2.setBounds(30, 700, 450, 60);
jsp2.setViewportView(tosend);
// tosend.setBounds(30, 700, 450, 60);
pic.setBounds(30, 780, 100, 30);
send.setBounds(200, 780, 100, 30);
send.addActionListener(new Actlis());
end.setBounds(380, 780, 100, 30);
chat.setEditable(false);
chat.setContentType("text/html");
chat.setText("請輸入 Server IP 與 port 號（10000）"); //set default chat text
selectedfilelabel.setBounds(30, 810, 300, 30);
end.addActionListener(new Actlis());
ctp.add(Server_label);
ctp.add(Server_port);
ctp.add(myServer);
ctp.add(myPort);
ctp.add(connect);
```

```
    ctp.add(jsp);
    ctp.add(jsp2);
    ctp.add(pic);
    ctp.add(send);
    ctp.add(end);
    ctp.add(selectedfilelabel);
    frm.setSize(550, 900);
    frm.setLocation(500, 150);
    frm.setVisible(true);
    frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

static public class rcvmsg extends Thread { //Thread for receiving messages

    public rcvmsg() {
        new Thread(this).start();
    }

    public void run() {
        // System.out.println("can receive");
        while (true) {
            try {

                String str = "";
                str = input.readUTF();
                if (!str.equals("")) {
                    str = str.replaceAll("\n", "<br>"); //normal text message, replace enter with <br>
```

```
                toprint += "Server IP: " + str + "<br>";
                chat.setText(toprint); //print text message on chat area
                chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area
automatically
              }
            } catch (Exception e) {
              // TODO: handle exception
            }
          }
        }
      }

    static String img = "";

    static public class rcvimg extends Thread { //Thread for receiving images
      public rcvimg() {
        new Thread(this).start();
      }

      public void run() {
        while (true) {
          try {
            // System.out.println("can rsv");
            BufferedInputStream in;

            InputStream ins = sc1.getInputStream();
            in = new BufferedInputStream(ins);
            byte[] b = new byte[1024];
```

```
            ByteArrayOutputStream buf = new ByteArrayOutputStream();
            int len;
            while ((len = in.read(b)) > 0) {//
               buf.write(b, 0, len); //write to buffer
            }
            System.out.println("received");
            OutputStream out = new FileOutputStream(new File("imgc" + imgcnt +".jpg")); //Save image as file, filename = imgs + imgcnt + .jpg
            buf.writeTo(out);
            out.close();
            sc1.shutdownInput();
            // in.close();
            in = null;
            String img = "<img src= 'file:imgc" + imgcnt + ".jpg' width = '200'> <br>"; //display image on chat area using image saved previously
            toprint += "Server IP: <br>" + img;
            imgcnt++;
            chat.setText(toprint);
            chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area automatically
          } catch (Exception e) {
            //System.out.println(e.toString() + "Client rcv");
          }
        }
      }
    }

    static public class sendmsg extends Thread { //Thread for sending messages (texts and images)
```

```
    private String msg;

    public sendmsg(String str) {
       msg = str;
       new Thread(this).start();
    }

    public void run() {
       if (image == false) { //if sending text
          try {
             msg = msg.replaceAll("\n", "<br>");
             if (!msg.equals("")) {
                output.writeUTF(msg); //send message to server
                toprint += "Client IP: " + msg + "<br>";
                chat.setText(toprint); //display message on chat area
                chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area
automatically
             }
          } catch (Exception e) {
             System.out.println("訊息傳送失敗");
          }
       } else { //sending image
          image = false; //restore default image value
          String img = "<img src= 'file:\\" + path + "' width = '200'> <br>"; //display image on chat area using
html syntax
          // toprint += "Server IP: ";
          toprint += "Client IP: <br>" + img;
          chat.setText(toprint);
```

```
            chat.setCaretPosition(chat.getDocument().getLength()); //scroll to bottom of chat area automatically
            try {
               FileInputStream fis = new FileInputStream(new File(path));
               OutputStream os = sc1.getOutputStream();
               BufferedOutputStream bos = new BufferedOutputStream(os);
               byte[] b = new byte[1024];
               int len;
               System.out.println("ready to send.");
               while ((len = fis.read(b)) != -1) {
                  bos.write(b, 0, len);
               }
               bos.flush();
               sc1.shutdownOutput(); //must shutdown output, otherwise it won't send
               // bos.close();
               fis.close();
               bos = null;
               System.out.println("sent!");
               selectedfilelabel.setText(""); //restore default Jlabel value
            } catch (Exception e) {
               System.out.println(e.toString() + "Cleint send");
            }
         }
      }
   }

   static InetAddress serverIPformat;
   static int serverport;
```

```
  static public class Actlis extends WindowAdapter implements ActionListener // .addActionListener(new
Actlis());
  {
    public void windowClosing(WindowEvent e) {
      System.exit(0);
    }

    // 192.168.50.128

    public void actionPerformed(ActionEvent e) {
      if (e.getSource() == connect) { //if connect button clicked
        // System.out.println("run");
        try {
          String serverIP = myServer.getText();
          InetAddress serverIPformat = InetAddress.getByName(serverIP);
          int serverport = Integer.parseInt(myPort.getText());
          sc = new Socket(serverIPformat, serverport); // 10000
          sc1 = new Socket(serverIPformat, 10001); // 10001
          // System.out.println("success");
          input = new DataInputStream(sc.getInputStream());
          input1 = new DataInputStream(sc1.getInputStream());
          output = new DataOutputStream(sc.getOutputStream());
          output1 = new DataOutputStream(sc1.getOutputStream());
          new rcvmsg(); //run new thread for receiving message
          new rcvimg(); //run new thread for receiving image
          chat.setText("連線成功！");
          connect.setEnabled(false); //disable connect button to prevent multiple connection causing errors
```

```
    } catch (Exception eee) {
        // TODO: handle exception
    }

}
if (e.getSource() == send) { //if send button clicked

    String str = tosend.getText();
    new sendmsg(str); //send message
    tosend.setText(""); //restore texting area
} else if (e.getSource() == pic) { //if sendpic button clicked
    JFileChooser cfile = new JFileChooser();
    cfile.setFileFilter(new FileNameExtensionFilter("Image files (*.GIF,*.PNG,*.JPG, *.JPEG)", "GIF", "PNG",
            "JPG", "JPEG")); //limit selected file type *extra function
    int returnfile = cfile.showOpenDialog(pic);
    if (returnfile == JFileChooser.APPROVE_OPTION) { //check if any file is selected
        selectedFile = cfile.getSelectedFile(); //get selected file
        selectedfilelabel.setText(cfile.getSelectedFile().getName()); //display the name of selected file
        path = selectedFile.getAbsolutePath(); //get path of the file
        image = true;
    } else {
        selectedfilelabel.setText("檔案未選擇");
    }
}

else if (e.getSource() == end) { //if end button clicked
    try {
```

```
                new sendmsg("EOF"); //send "EOF" to server
                Thread.sleep(500); //wait for 500 millisec to process the sending process
                input.close(); //close everything
                input1.close();
                output.close();
                output1.close();
                sc.close();
                sc1.close();
                System.exit(0); //close the window
            } catch (Exception ec) {
                // TODO: handle exception
            }
        }
    }
  }
}
```

問題、討論與心得

問題與討論

1. 問題：圖片無法顯示於聊天區域中。

   解決辦法：一開始聊天室是用 textarea 設計的，但上網查詢資料後發現 textarea 只能顯示文字，無法顯示圖片。因此在詢問學長後，改以 textpane 設計聊天區域，並加入 chat.setContentType("text/html"); 指令，使 textpane 能夠以 html 的語法顯示相對應的物件，如此一來，圖片就可以以<img src=”” >的語法顯示，並可以設定圖片顯示的大小等微調選項。

2. 問題：圖片在無法只能單方面傳送至對方，傳完一次後，對方傳送的檔案便無法正確接收。

   解決辦法：這個錯誤一開始是由於多個錯誤所引起的，如：rcvimg 的 thread 沒有加上 while(true)，導致只會執行一次就停止了。修正之後，發現還是無法雙向傳輸照片。上網查詢後發現，若將 datainputstream 關閉，就等同將整個 socket 關閉，將其改為 sc1.shutdownInput(); 後，只關閉需要關閉的部分即可修正錯誤。

3. 問題：因不熟悉直接將存入 buffer 的圖檔資料直接顯示於聊天視窗中，因此決定先將收到的圖檔先存為本地檔案後，在將本地檔案顯示於聊天室中，但若傳輸多於一張照片，會有圖片同名的問題。

解決方法：為解決圖片同名的問題，在接收圖片的 thread 中加入計數器，並將計數器的數值加入圖檔名稱中，若為 server 端接收，圖檔名稱為：imgs + 計數器數值 .png，若為 client 端接收，圖檔名稱為：imgc+計數器數值.png，如此便可解決檔案同名的問題。

心得

原以為本次的作業因是使用 TCP，不會有傳輸後沒收到的問題，將變的單純一些，但事實卻不是如此。雖然少了需重複傳送以提高接收率的步驟，但因為題目要求可傳送文字與圖片，在圖片部分卡關了好久。幸好由於上次的 UDP 作業，我即是使用 JFrame 撰寫呈現，因此在視窗化的過程中，並沒有遇到太大的困難。本次作業中，我認為另一個有困難的部分為執行緒的實作。在此之前，我甚至連執行緒這個詞都沒聽過。因此在實作前，花了點時間上網搜尋了 java 多執行緒的應用與實作，才搞懂該如何在 class 中 extend Thread 與 new 一個 thread。在程式撰寫過程中，圖片部分因會使用到 socket 的 file 與 data 的 input/ output stream，在決定何時要開啟/關閉上因多次嘗試無效而感到十分挫折。雖然花了許多時間，甚至比上次的 UDP 還要更複雜些，但最後看見文字與圖片能正確傳出與接收，成就感也頗高。

附錄

一、 程式 demo 附於壓縮檔案內，亦上傳 Google Drive，網址為 https://drive.google.com/file/d/1u7q32aLlgaG05dHGnwmkxVUfLV34mAEw/view?usp=sharing