TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA Công Nghệ Thông Tin

BỘ MÔN: MẠNG & TT

ĐỀ THI VÀ BÀI LÀM - Đề 1

Tên học phần: Lập Trình Mạng
Mã học phần: Hình thức thi: *Tự luận có giám sát*
Thời gian làm bài: 30 phút *(không kể thời gian phát đề và nộp bài)*
Được sử dụng tài liệu khi làm bài. Không chia sẻ bài cho nhau, nếu phát hiện sẽ chia đều số điểm.

Họ tên: Đoàn Kiều Ngân **Lớp**: 23T_DT1 **MSSV**:102230201

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam:

Câu 1 (*8 điểm*): Hãy xây dựng chương trình chia sẻ màn hình từ server cho các client. Lưu ý cần tối ưu cho mỗi client về cả đường truyền lẫn chất lượng hình ảnh.

 # Trả lời:

Dán code server vào bên dưới

```
package Buoi08;


import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.net.*;
import java.util.Iterator;
import javax.imageio.*;
import javax.imageio.plugins.jpeg.JPEGImageWriteParam;
import javax.imageio.stream.ImageOutputStream;


public class ScreenServer {
    public static void main(String[] args) {
        new ScreenServer();
    }


    public ScreenServer() {
```

```
        Screen s = new Screen();
        s.start();
        try {
            ServerSocket server = new ServerSocket(2345);
            System.out.println("Server listening on port 2345...");
            while (true) {
                Socket soc = server.accept();
                ScreenProcessing sp = new ScreenProcessing(soc);
                sp.start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

class Screen extends Thread {
    static byte[] tmp;
    static int count = 0;
    static boolean isFull = true;
    static BufferedImage lastFrame = null;

    public void run() {
        Robot r = null;
        Rectangle capture = null;
        try {
            r = new Robot();
            capture = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        while (true) {
            try {

                BufferedImage img = r.createScreenCapture(capture);

                int newW = img.getWidth() / 2;
                int newH = img.getHeight() / 2;
                BufferedImage scaled = new BufferedImage(newW, newH,
BufferedImage.TYPE_INT_RGB);
                Graphics2D g = scaled.createGraphics();
                g.drawImage(img, 0, 0, newW, newH, null);
                g.dispose();

                if (lastFrame == null) {
                    tmp = encodeJpeg(scaled, 0.5f);
                    isFull = true;
                    lastFrame = scaled;
                } else {

                    ByteArrayOutputStream bos = new ByteArrayOutputStream();
                    DataOutputStream dos = new DataOutputStream(bos);
                    int w = scaled.getWidth();
                    int h = scaled.getHeight();
                    dos.writeInt(w);
                    dos.writeInt(h);
                    int changes = 0;

                    for (int y = 0; y < h; y++) {
                        for (int x = 0; x < w; x++) {
                            int rgbNew = scaled.getRGB(x, y);
                            int rgbOld = lastFrame.getRGB(x, y);
                            if (rgbNew != rgbOld) {
```

```
                        dos.writeInt(x);
                        dos.writeInt(y);
                        dos.writeInt(rgbNew);
                        changes++;
                    }
                }
            }

            if (changes > 0 && changes < (w * h / 10)) {
                tmp = bos.toByteArray();
                isFull = false;
                lastFrame = scaled;
            } else {
                tmp = encodeJpeg(scaled, 0.8f);
                isFull = true;
                lastFrame = scaled;
            }
        }

        count++;
        if (count % 30 == 0) {
            System.out.println("Frame " + count + " size=" + tmp.length + (isFull ? " (FULL)" : "
(DELTA)"));
        }

        Thread.sleep(50);
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
  }
```

```
    private byte[] encodeJpeg(BufferedImage img, float quality) throws IOException {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        Iterator<ImageWriter> writers = ImageIO.getImageWritersByFormatName("jpg");
        ImageWriter writer = writers.next();
        ImageOutputStream ios = ImageIO.createImageOutputStream(bos);
        writer.setOutput(ios);

        JPEGImageWriteParam jpegParams = new JPEGImageWriteParam(null);
        jpegParams.setCompressionMode(JPEGImageWriteParam.MODE_EXPLICIT);
        jpegParams.setCompressionQuality(quality);

        writer.write(null, new IIOImage(img, null, null), jpegParams);
        ios.close();
        writer.dispose();
        return bos.toByteArray();
    }
}

class ScreenProcessing extends Thread {
    Socket soc;
    int countNow;

    public ScreenProcessing(Socket soc) {
        this.soc = soc;
    }

    public void run() {
        try {
            DataOutputStream output = new DataOutputStream(soc.getOutputStream());
            while (true) {
                if (countNow == Screen.count) {
                    Thread.sleep(1);
```

```
                continue;
            }
            byte[] tmp = Screen.tmp;
            boolean isFull = Screen.isFull;
            countNow = Screen.count;

            output.writeBoolean(isFull);
            output.writeInt(tmp.length);
            output.write(tmp);
            output.flush();
            Thread.sleep(1);
          }
      } catch (Exception e) {
          e.printStackTrace();
      }
   }
}
```

Dán code client vào bên dưới

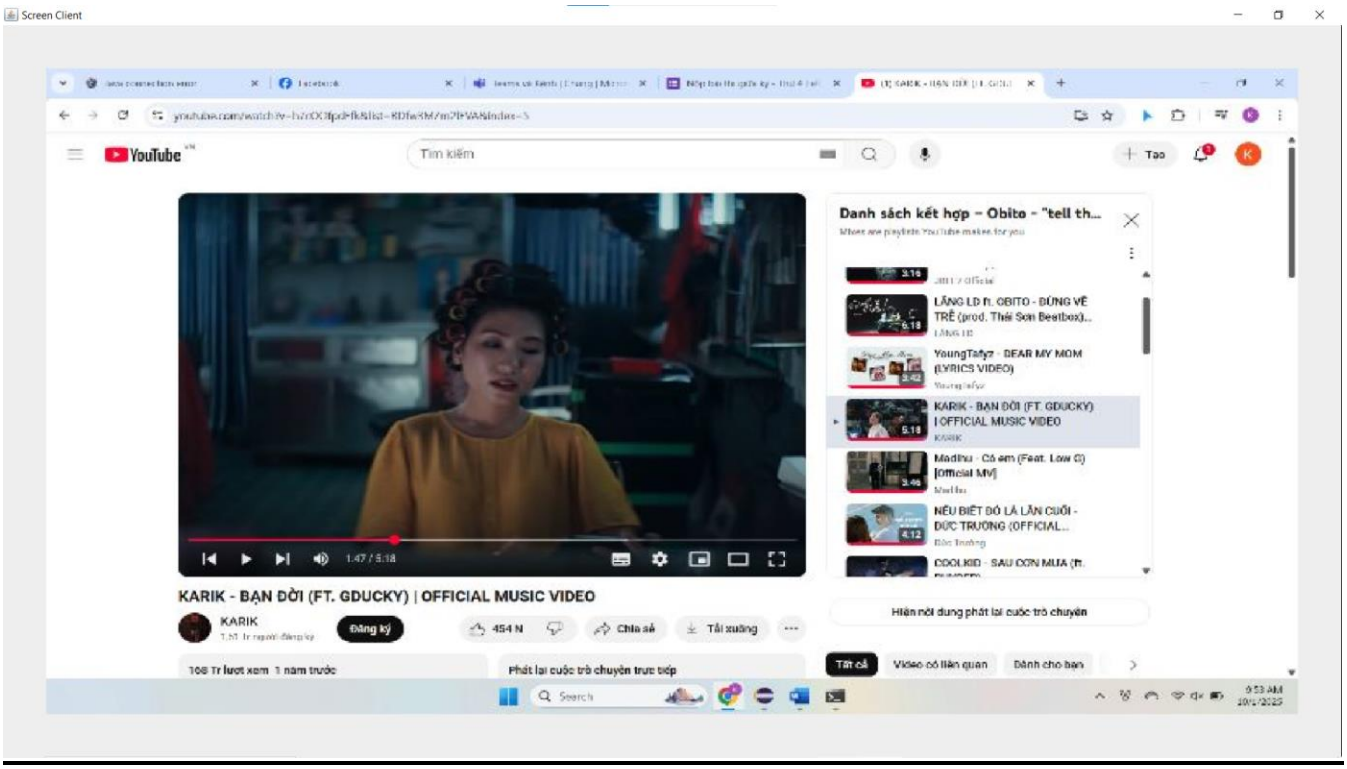
```
package Buoi08;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.net.Socket;
import javax.imageio.ImageIO;
import javax.swing.*;

public class ScreenClient extends JFrame {
   Socket soc;
```

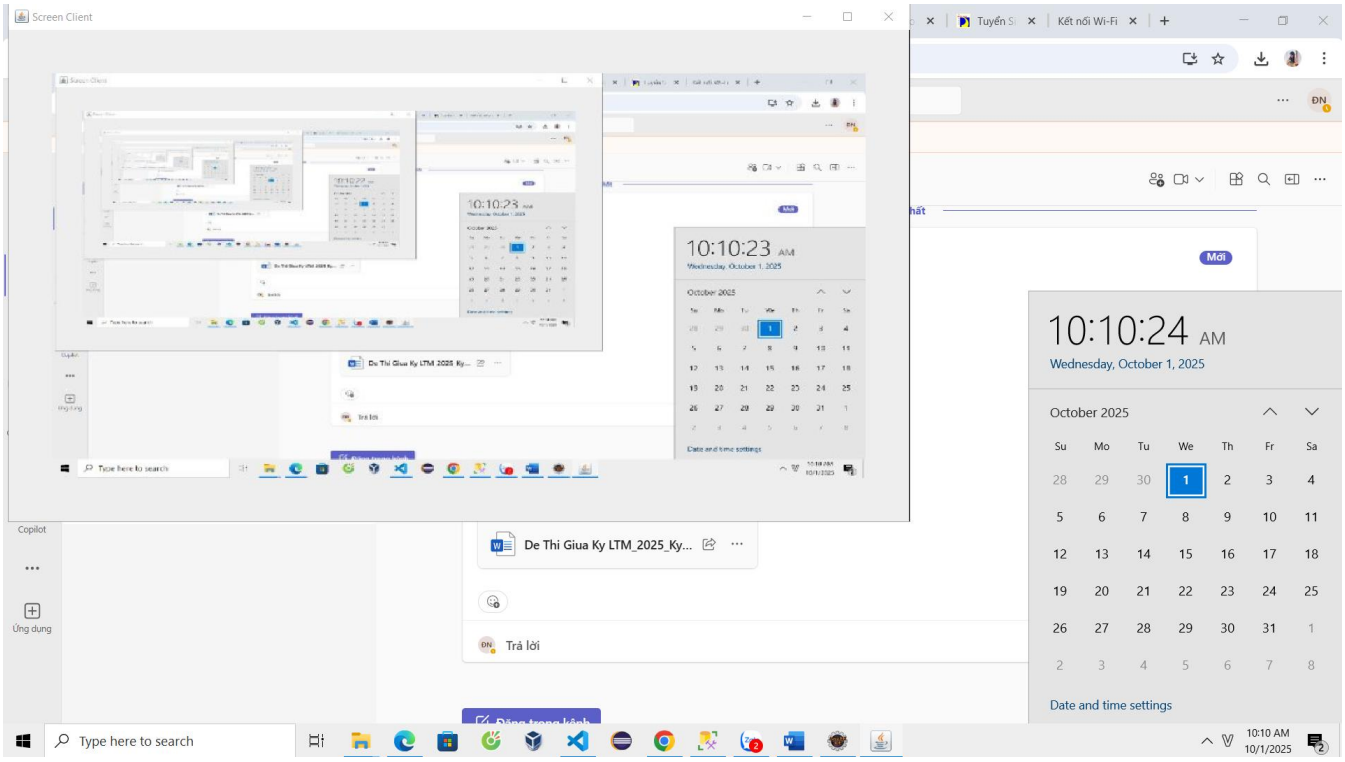
```
BufferedImage lastFrame = null;
int off = 50;

public static void main(String[] args) {
    new ScreenClient();
}

public ScreenClient() {
    this.setTitle("Screen Client");
    this.setSize(800, 600);
    this.setDefaultCloseOperation(3);

    try {
        soc = new Socket("192.168.56.1 ", 2345);
        Receiver r = new Receiver(soc);
        r.start();
    } catch (Exception e) {
        System.exit(1);
    }

    JPanel panel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            if (lastFrame != null) {
                int w = getWidth() - 2 * off;
                int h = getHeight() - 2 * off;
                Image img2 = lastFrame.getScaledInstance(w, h, Image.SCALE_SMOOTH);
                g.drawImage(img2, off, off, getWidth() - off, getHeight() - off,
                        0, 0, w, h, null);
            }
        }
```

```
    };

    this.add(panel);
    this.setVisible(true);

    new Timer(50, e -> panel.repaint()).start();
}

class Receiver extends Thread {
    Socket soc;

    public Receiver(Socket soc) {
        this.soc = soc;
    }

    public void run() {
        try {
            DataInputStream dis = new DataInputStream(soc.getInputStream());
            while (true) {
                boolean isFull = dis.readBoolean();
                int n = dis.readInt();
                byte tmp[] = dis.readNBytes(n);

                if (isFull) {
                    ByteArrayInputStream bis1 = new ByteArrayInputStream(tmp);
                    lastFrame = ImageIO.read(bis1);
                } else {
                    if (lastFrame != null) {
                        DataInputStream deltaIn = new DataInputStream(new ByteArrayInputStream(tmp));
                        int w = deltaIn.readInt();
                        int h = deltaIn.readInt();
                        while (deltaIn.available() > 0) {
```

```
                    int x = deltaIn.readInt();
                    int y = deltaIn.readInt();
                    int rgb = deltaIn.readInt();
                    if (x < w && y < h) {
                        lastFrame.setRGB(x, y, rgb);
                    }
                }
            }
        }
    } catch (Exception e) {
        System.exit(0);
    }
  }
}
```

Dán một số các kết quả thực thi vào bên dưới cho thấy mỗi client có tốc độ đường truyền cũng như chất lượng hình ảnh khác nhau





Câu 2 (*2 điểm*): Hãy mô tả bằng lời rằng em đã làm gì trong đoạn code ở trên để tối ưu về tốc độ đường truyền, chất lượng hình ảnh **cho mỗi client**. **Bạn nào dùng AI để trả lời coi như không có điểm**.

 # Trả lời:

- Để tối ưu tốc độ đường truyền thì em đã làm những việc sau:
  + Giảm kích thước hình ảnh màn hình xuống còn một nửa kích thước gốc => giảm dung lượng hình ảnh.
  + So sánh từng pixel giữa frame mới và frame trước để xác định vùng thay đổi nếu số lượng pixel thay đổi ít hơn 10% thì chỉ gửi phần thay vì toàn bộ ảnh => tiết kiệm băng thông đáng kể.
  + Điều chỉnh tốc độ gửi khoảng 20 frame/giây => giúp tránh quá tải mạng và CPU.
- Để tối ưu chất lượng hình ảnh:
  + Nâng chất lượng nén từ 0.5f lên 0.8f => giúp ảnh rõ nét hơn, ít bị mờ hoặc mất chi tiết
  + Lưu lại frame trước (lastFrame) để phục vụ việc so sánh => đảm bảo rằng delta được tính chính xác, tránh gửi dữ liệu thừa

Đà Nẵng, ngày tháng năm 2025