Carpeta aparte audioRecorder:

import javax.sound.sampled.*\**;

import java.io.*\**;

import java.util.concurrent.TimeUnit;

public class AudioRecorderPlayer {

    private static  int SAMPLE\_RATE = 16000; *// Frecuencia de muestreo en Hz*

    private static  int SAMPLE\_SIZE\_IN\_BITS = 16; *// Tamaño de muestra en bits*

    private static  int CHANNELS = 1; *// Mono*

    private static  boolean SIGNED = true; *// Muestras firmadas*

    private static  boolean BIG\_ENDIAN = false; *// Little-endian*

    public static void main(String[] args) {

*//Iniciar variables y objetos necesarios para definir formato y buffer donde se guardara el audio*

        int duration = 5; *//cuantos segundos vamos a grabar?*

        AudioFormat format = new AudioFormat(SAMPLE\_RATE, SAMPLE\_SIZE\_IN\_BITS, CHANNELS, SIGNED, BIG\_ENDIAN);

        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();

*//iniciar objeto de grabacion de audio*

        RecordAudio recorder = new RecordAudio(format, duration,byteArrayOutputStream);

        Thread recorderTrh   = new Thread(recorder);

        recorderTrh.start();

*//esperar a que la grabacion termine*

        try{

            recorderTrh.join();

        }catch(Exception e){

*//TODO*

        }

*// Reproducir el audio grabado*

        byte[] audioData = byteArrayOutputStream.toByteArray();

        PlayerRecording player = new PlayerRecording(format);

        player.initiateAudio(audioData);

    }

}

import java.io.*\**;

import java.net.*\**;

import javax.sound.sampled.*\**;

public class PlayerRecording {

    private AudioFormat format;

    private byte[] audioData; *//datos de entrada*

    private SourceDataLine out;  *//salida a la tarjeta de audio*

    private AudioInputStream in;

    public PlayerRecording(AudioFormat format) {

*this*.format=format;

    }

    public void initiateAudio(byte[] audioData) {

        try {

            in = new AudioInputStream(new ByteArrayInputStream(audioData), format,

                    audioData.length / format.getFrameSize());

*// Abrir línea de salida de audio*

            out = AudioSystem.getSourceDataLine(format);

            out.open(format);

            out.start(); *// Comenzar la reproducción de audio*

            playAudio();

        } catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

    private void playAudio() {

        byte[] buffer = new byte[1024];

        int count;

        try {

            System.out.println("Reproduciendo...");

            while ((count = in.read(buffer)) != -1) {

                    out.write(buffer, 0, count);

            }

            out.drain();

            out.stop();

            out.close();

            in.close();

        } catch (Exception e) {

            System.out.println(e.getMessage());

        }

    }

}

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.util.concurrent.TimeUnit;

import javax.sound.sampled.AudioFormat;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.DataLine;

import javax.sound.sampled.TargetDataLine;

public class RecordAudio implements Runnable{

    private AudioFormat format;

    private int duration;

    private  ByteArrayOutputStream out;

    public RecordAudio(AudioFormat format, int duration, ByteArrayOutputStream out){

*this*.format = format;

*this*.duration= duration;

*this*.out    = out;

    }

    @Override

    public void run(){

        int bytesRead;

        try {

*// Abrir línea de captura de audio*

            DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

            TargetDataLine targetLine = (TargetDataLine) AudioSystem.getLine(info);

            targetLine.open(format);

            targetLine.start(); *// Comenzar la captura de audio*

            System.out.println("Grabando durante " + duration +" segundos...");

*// grabar audio durante t segundos*

            byte[] buffer = new byte[targetLine.getBufferSize() / 5];

            long startTime = System.currentTimeMillis();

            while (System.currentTimeMillis() - startTime < TimeUnit.SECONDS.toMillis(duration)) {

                    bytesRead = targetLine.read(buffer, 0, buffer.length);

                    out.write(buffer, 0, bytesRead);

                }

                targetLine.stop();

                targetLine.close();

        }catch(Exception e){

*//TODO: handle exception*

        }

    }

}

Carpeta aparte senderReceiver:

Subcarpeta receiver:

import java.io.*\**;

import java.net.*\**;

import java.nio.ByteBuffer;

import javax.sound.sampled.*\**;

public class MusicReceiver {

*//debe pasar el nombre de la cancion en los argumentos*

    public static void main(String[] args) throws Exception {

        InetAddress IPAddress = InetAddress.getByName("localhost"); *// Dirección IP del servidor en la red local*

        int PORT = 6789;

        int BUFFER\_SIZE = 1024 + 4;

        DatagramSocket clientSocket = new DatagramSocket();

        PlayerThread playerThread;  *//rrepductor usando un hilo para almacenar paquetes y no saturar la tarjeta de audio*

*// Configurar el reproductor de audio*

        AudioFormat audioFormat = new AudioFormat(16000, 16, 1, true, false);

        playerThread = new PlayerThread(audioFormat,BUFFER\_SIZE);

        playerThread.start();

*// contactar al servidor para que inicie el envio de la cancion*

        String mensaje = "Hola servidor, enviame una cancion... #" ;

        byte[] sendData = mensaje.getBytes();

        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, PORT);

        clientSocket.send(sendPacket);

        byte[] buffer = new byte[BUFFER\_SIZE];

*// Recibir los paquetes y reproducir el audio*

       int count = 0;

       while (true) {

           DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

           clientSocket.receive(packet);

           buffer = packet.getData();

           ByteBuffer byteBuffer = ByteBuffer.wrap(buffer);

           int packetCount = byteBuffer.getInt();

           if (packetCount == -1) {

               System.out.println("Received last packet " + count);

               break;

           } else {

               byte[] data = new byte[1024];

               byteBuffer.get(data, 0, data.length);

               playerThread.addBytes(data);

               System.out.println("Received packet " + packetCount + " current: " + count);

           }

           count++;

       }

       clientSocket.close();

    }

}

import java.util.concurrent.ArrayBlockingQueue;

import java.util.concurrent.BlockingQueue;

import javax.sound.sampled.AudioFormat;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.SourceDataLine;

public class PlayerThread extends Thread {

    private static int MAX\_ITEMS\_IN\_QUEUE = 3;

    private int secondsBuffer = 280;

    BlockingQueue<byte[]> buffer;

    private SourceDataLine sourceDataLine;

    private int count = 0;

    private int packes = 0;

    public PlayerThread(AudioFormat audioFormat, int BUFFER\_SIZE) {

        try {

            MAX\_ITEMS\_IN\_QUEUE = (int) audioFormat.getSampleRate() \* secondsBuffer \*

                    audioFormat.getFrameSize()

                    / BUFFER\_SIZE;

            System.out.println("Max items in queue: " + MAX\_ITEMS\_IN\_QUEUE);

            buffer = new ArrayBlockingQueue<>(MAX\_ITEMS\_IN\_QUEUE, true);

            sourceDataLine = AudioSystem.getSourceDataLine(audioFormat);

            sourceDataLine.open(audioFormat);

            sourceDataLine.start();

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

*//agregar bytes a la cola, esto deberia permitir que no se pierdan paquetes*

    public void addBytes(byte[] bytes) {

        try {

            count++;

            buffer.put(bytes);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

*//hilo principal, toma los bytes de la cola y los pone en la tarjeta de audio*

    public void run() {

        while (true) {

            try {

                if (buffer.isEmpty()) {

                    if (packes > 0) {

                        System.out.println("Packets: write " + packes + " add Count: " + count);

                        packes = 0;

                        count = 0;

                    }

                    Thread.yield();

                    continue;

                }

                byte[] bytes = buffer.take();

                packes++;

                sourceDataLine.write(bytes, 0, bytes.length);

            } catch (Exception e) {

                e.printStackTrace();

            }

        }

    }

}

Subcarpeta sender:

import java.io.*\**;

import java.net.*\**;

import javax.sound.sampled.*\**;

public class MusicSender {

*//debe pasar el nombre de la cancion en los argumentos*

    public static void main(String[] args) throws Exception {

        InetAddress IPAddress = InetAddress.getByName("localhost"); *// Dirección IP del servidor en la red local*

        String song = "songs/Song1\_16k.wav";

        int PORT = 6789;

        DatagramSocket serverSocket = new DatagramSocket(PORT,IPAddress);

        System.out.println("Esperando solicitud del cliente!!");

*//esperar cliente*

        byte[] receiveData = new byte[1024];

        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

        serverSocket.receive(receivePacket);

        System.out.println("Cliente conectado!! \nEnviando Cancion!!");

*//crear objeto e iniciar envio*

        PlayerSender sender = new PlayerSender(song,receivePacket,serverSocket);

        sender.sendAudio();

    }

}

import java.io.*\**;

import java.net.*\**;

import javax.sound.sampled.*\**;

import java.nio.ByteBuffer;

public class PlayerSender {

    AudioInputStream in; *//datos de entrada*

    private String route;

    private InetAddress clienteIPAddress;

    private int         clientPort;

    private DatagramSocket socket;

    private int BUFFER\_SIZE = 1024;

    public PlayerSender(String route, DatagramPacket receivePacket, DatagramSocket serverSocket) {

*this*.route=route;

*this*.clienteIPAddress = receivePacket.getAddress();

*this*.clientPort = receivePacket.getPort();

*this*.socket = serverSocket;

    }

*//metodo principal para leer el audio y enviarlo*

    public void sendAudio() {

        byte[] buffer = new byte[BUFFER\_SIZE];

        int bytesRead;

        ByteBuffer byteBuffer = ByteBuffer.allocate(BUFFER\_SIZE+4);

        try {

            File file = new File(route);

            in = AudioSystem.getAudioInputStream(file);

*// Leer los datos de audio y enviarlos en paquetes*

            int count = 0;

            while ((bytesRead = in.read(buffer, 0, buffer.length)) != -1) {

                byteBuffer.clear();

                byteBuffer.putInt(count);

                byteBuffer.put(buffer, 0, bytesRead);

                byte data[] = byteBuffer.array();

                sendPacket(data);

                System.out.println("Sent packet " + count++);

            }

            byteBuffer.clear();

            byteBuffer.putInt(-1);

            byte data[] = byteBuffer.array();

            sendPacket(data);

            socket.close();

        } catch (Exception e) {

*// TODO: handle exception*

        }

    }

*//armar el datagram y ponerlo en el socket*

    public void sendPacket(byte[] audioData) throws Exception {

        DatagramPacket packet = new DatagramPacket(audioData, audioData.length, clienteIPAddress, clientPort);

        socket.send(packet);

    }

}

Ayudame por favot te lo pido