package lte;

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.OutputStream;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.MulticastSocket;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.URL;

import java.net.URLConnection;

import java.util.Iterator;

import java.util.Map;

import java.util.Properties;

import java.util.Set;

import java.util.StringTokenizer;

import java.util.TreeMap;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.util.Duration;

import javax.swing.JOptionPane;

import org.apache.commons.io.IOUtils;

public class Gatewaysinglereceiver extends Thread {

public String gatewayn, portno, sysno, allvideo = "", neighbourvideo = "", localvideo = "";

String ssss="";

int delcount=0;

Gatewaymultireceiver gmr;

LocalServingGatewayController lsgc;

Observer o = new Observer();

public Vector allvid = new Vector();

public TreeMap allvideocount = new TreeMap();

public Vector supersourcevect = new Vector();

private int VID\_WIDTH = 320;

private int VID\_HEIGHT = 180;

private int PLAYER\_WIDTH = 320;

private int PLAYER\_HEIGHT = 265;

File createpath = new File("");

File locf = new File(createpath.getAbsolutePath()+"\\localgateway");

public Gatewaysinglereceiver(String gateway, String portno, String sysno, Gatewaymultireceiver gmr, LocalServingGatewayController lsgc) {

this.gatewayn = gateway;

this.portno = portno;

this.sysno = sysno;

this.gmr = gmr;

this.lsgc = lsgc;

this.o = lsgc.getObeserver();

locf.mkdir();

start();

}

@Override

public void run() {

try {

ServerSocket ss = new ServerSocket(Integer.parseInt(portno));

while (true)

{

Socket s = ss.accept();

ObjectInputStream ois = new ObjectInputStream(s.getInputStream());

String status = (String) ois.readObject();

if (status.equalsIgnoreCase("VideoData"))

{

System.out.println("video start to receiving");

String videoname = (String) ois.readObject();

byte[] video = (byte[]) ois.readObject();

}

else if (status.equalsIgnoreCase("ShareVideo"))

{

String videoname = (String) ois.readObject();

byte[] video = (byte[]) ois.readObject();

// ==============build the content caching===========

neighbourvideo = neighbourvideo + videoname + "\n";

// File f = new File("D:\\Alaguvigneshwar\\Sample\\" + this.gatewayn);

File f = new File(locf.getAbsolutePath()+"\\"+this.gatewayn);

f.mkdir();

FileOutputStream out = new FileOutputStream(f.getAbsolutePath()+ "\\" + videoname);

out.write(video);

out.close();

Platform.runLater(new Runnable()

{

@Override

public void run()

{

o.setavailablevideo(neighbourvideo);

}

});

try

{

String str = videoname + "$" + this.gatewayn;

InetAddress in = InetAddress.getByName("225.89.67.45");

MulticastSocket ms = new MulticastSocket(4567);

ms.joinGroup(in);

DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), in, 4567);

ms.send(dp);

} catch (Exception e)

{

e.printStackTrace();

}

}

else if (status.equalsIgnoreCase("VideoRequest"))

{

String choose = (String)ois.readObject();

String sourcenode = (String) ois.readObject();

String location = (String)ois.readObject();

String videoreq = (String) ois.readObject();

Platform.runLater(new Runnable()

{

@Override

public void run()

{

o.setreqsource(sourcenode);

}

});

reqtoserver(choose,sourcenode, videoreq,location);

}

else if (status.equalsIgnoreCase("availvideo"))

{

String node = (String) ois.readObject();

String video = gmr.urlcall();

try

{

String sys = gmr.nodesysno.get(node).toString();

String port = gmr.nodeportno.get(node).toString();

s = new Socket(sys, Integer.parseInt(port));

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("userdisplay");

oos.writeObject(video);

oos.close();

s.close();

}

catch (Exception e) {

e.printStackTrace();

}

}

else if (status.equalsIgnoreCase("pathsendtogateway"))

{

String path = (String) ois.readObject();

String internode = (String) ois.readObject();

}

else if (status.equalsIgnoreCase("Costbasedpath"))

{

String choose = (String)ois.readObject();

String path = (String) ois.readObject();

String dest = (String) ois.readObject();

String intermediate = (String) ois.readObject();

String source = (String) ois.readObject();

String videoname = (String) ois.readObject();

if(lsgc.dpathob.get(0).getColumndestination().equals(""))

{

lsgc.dpathob.clear();

}

lsgc.dpathob.add(new Destsplit(source, intermediate, dest));

if(choose.equalsIgnoreCase("PLAY"))

{

stream(path, videoname);

System.out.println("select play");

}

else

{

download(path,videoname);

System.out.println("select download");

}

}

}

}

catch (Exception e)

{

e.printStackTrace();

}

}

//

public void reqtoserver(String choose,String sourcenode, String video,String location) throws IOException

{

JOptionPane.showMessageDialog(null,"Video Download From Server");

File f = new File("");

String proper = f.getAbsolutePath();

String filepath = proper+"\\address.properties";

FileInputStream fs =new FileInputStream(filepath);

Properties pro = new Properties();

pro.load(fs);

String locl = pro.getProperty("IP");

for (int i = 0; i < gmr.vvv.size(); i++)

{

String streamvideo = gmr.vvv.get(i).toString().trim();

if (video.equalsIgnoreCase(streamvideo))

{

try

{

URL url = new URL("http://"+locl+":9999/SAVEP2p/Admin/videos/" + streamvideo);

URLConnection conection = url.openConnection();

conection.connect();

InputStream input = new BufferedInputStream(url.openStream(), 10 \* 1024);

Platform.runLater(new Runnable()

{

@Override

public void run()

{

o.setreqsource(sourcenode);

}

});

localgate(choose,streamvideo, input, sourcenode,location);

//

input.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

}

//

void localgate(String choose,String videoname, InputStream input, String sourcenode,String location) throws FileNotFoundException, IOException {

// ==============build the content caching===========

if(choose.equalsIgnoreCase("play"))

{

File f = new File(locf.getAbsolutePath()+"\\" + this.gatewayn);

if (!allvid.contains(videoname))

{

allvid.add(videoname);

localvideo = localvideo + videoname + "\n";

}

boolean mss = f.exists();

if (mss == true)

{

FileOutputStream out = new FileOutputStream(f.getAbsolutePath()+ "\\" + videoname);

byte[] data = new byte[1024];

int count = 0;

while ((count = input.read(data)) != -1)

{

out.write(data, 0, count);

out.flush();

}

out.close();

Platform.runLater(new Runnable()

{

@Override

public void run()

{

try

{

o.setavailablevideo(localvideo);

if (!supersourcevect.contains(videoname))

{

supersourcevect.add(videoname);

lsgc.createnode(videoname);

}

} catch (IOException ex)

{

Logger.getLogger(Gatewaysinglereceiver.class.getName()).log(Level.SEVERE, null, ex);

}

}

});

}

else

{

f.mkdir();

try

{

FileOutputStream out = new FileOutputStream(f.getAbsolutePath() + "\\" + videoname);

byte[] data = new byte[1024];

int count = 0;

while ((count = input.read(data)) != -1)

{

out.write(data, 0, count);

// Flush output

out.flush();

}

out.close();

}

catch (Exception e)

{

e.printStackTrace();

}

Platform.runLater(new Runnable()

{

@Override

public void run()

{

try

{

o.setavailablevideo(localvideo);

if (!supersourcevect.contains(videoname))

{

supersourcevect.add(videoname);

lsgc.createnode(videoname);

}

}

catch (IOException ex)

{

Logger.getLogger(Gatewaysinglereceiver.class.getName()).log(Level.SEVERE, null, ex);

}

}

});

}

try

{

sleep(2000);

String str = videoname + "$" + this.gatewayn;

System.out.println("string str = "+str);

InetAddress in = InetAddress.getByName("225.89.67.45");

MulticastSocket ms = new MulticastSocket(4567);

ms.joinGroup(in);

DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), in, 4567);

ms.send(dp);

}

catch (Exception e)

{

e.printStackTrace();

}

String innodename = (String)gmr.allinnodes.firstElement();

String insys = (String)gmr.allinsysno.get(innodename).toString();

String inport = (String)gmr.allinport.get(innodename).toString();

Socket s = new Socket(insys, Integer.parseInt(inport));

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("sendtointermediate");

oos.writeObject(sourcenode);

oos.writeObject(location);

oos.writeObject(videoname);

oos.close();

s.close();

try

{

System.out.println("video sending");

// =======video convert byte array===========

InputStream fin = new FileInputStream(f.getAbsolutePath()+ "\\" + videoname);

byte[] byteArr = IOUtils.toByteArray(fin);

// ==========byte array send socket===========

String sysnum = (String) gmr.nodesysno.get(sourcenode);

String portnum = (String) gmr.nodeportno.get(sourcenode);

s = new Socket(sysnum, Integer.parseInt(portnum));

oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("videodesttosource");

oos.writeObject(f.getAbsolutePath());

oos.writeObject(videoname);

oos.writeObject(byteArr);

fin.close();

oos.close();

s.close();

}

catch (Exception e)

{

e.printStackTrace();

}

// ==============share video to all gateway============

if (gmr.allgateway.size() != 0)

{

for (int i = 0; i < gmr.allgateway.size(); i++)

{

String gatewayname = gmr.allgateway.get(i).toString();

if (!this.gatewayn.equalsIgnoreCase(gatewayname))

{

String sysno = gmr.gatewaysysno.get(gatewayname).toString();

String portno = gmr.gatewayportno.get(gatewayname).toString();

FileInputStream fi = new FileInputStream(f.getAbsolutePath()+ "\\" + videoname);

byte[] byteArr = IOUtils.toByteArray(fi);

s = new Socket(sysno, Integer.parseInt(portno));

oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("ShareVideo");

oos.writeObject(videoname);

oos.writeObject(byteArr);

fi.close();

oos.close();

s.close();

}

}

}

}

else

{

JOptionPane.showMessageDialog(null,"Please first play a video");

}

}

// ========================After create a content caching========================

void stream(String path, String videoname) throws FileNotFoundException, IOException, InterruptedException

{

if(!allvideocount.containsKey(videoname))

{

Vector videocount = new Vector();

videocount.add(videoname);

allvideocount.put(videoname,videocount);

}

else

{

Vector videocount = (Vector)allvideocount.get(videoname);

videocount.add(videoname);

allvideocount.put(videoname, videocount);

}

System.out.println(this.gatewayn+"Value is: :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::" +allvideocount);

File f = new File(locf.getAbsolutePath()+"\\"+ this.gatewayn);

String streampath = f.getAbsolutePath()+"\\"+videoname;

System.out.println("string path = = "+streampath);

Vector vvv = new Vector();

StringTokenizer sss = new StringTokenizer(path, "->");

while (sss.hasMoreTokens())

{

String pathelement = sss.nextToken();

vvv.add(pathelement);

}

vvv.remove(0);

String nextnode = (String) vvv.firstElement();

String traffic = gmr.alltraffic.get(this.gatewayn).toString();

FileInputStream out = new FileInputStream(f.getAbsolutePath()+ "\\" + videoname);

byte[] byteArr = IOUtils.toByteArray(out);

//

String sys = (String) gmr.allinsysno.get(nextnode);

String port = (String) gmr.allinport.get(nextnode);

Socket s = new Socket(sys, Integer.parseInt(port));

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("Streampath");

// oos.writeObject(totalsplit);

oos.writeObject(streampath);

oos.writeObject(videoname);

oos.writeObject(vvv);

oos.writeObject(byteArr);

oos.close();

s.close();

if(delcount == 0)

{

videodelete(f);

delcount++;

}

}

public void download(String path,String videoname) throws FileNotFoundException, IOException

{

File f = new File(locf.getAbsolutePath()+"\\"+ this.gatewayn);

String streampath = f.getAbsolutePath()+"\\"+videoname;

if(!allvideocount.containsKey(videoname))

{

Vector videocount = new Vector();

videocount.add(videoname);

allvideocount.put(videoname,videocount);

}

else

{

Vector videocount = (Vector)allvideocount.get(videoname);

videocount.add(videoname);

allvideocount.put(videoname, videocount);

}

Vector vvv = new Vector();

StringTokenizer sss = new StringTokenizer(path, "->");

while (sss.hasMoreTokens())

{

String pathelement = sss.nextToken();

vvv.add(pathelement);

}

vvv.remove(0);

String nextnode = (String) vvv.firstElement();

String traffic = gmr.alltraffic.get(this.gatewayn).toString();

String rpe = traffic.replace("%", "");

int tra = Integer.parseInt(rpe);

int totalsplit;

if(tra<10)

{

totalsplit=1;

}

else

{

totalsplit = tra / 10;

}

File savesplit = SplitVideoFile(streampath,totalsplit);

File[] arrfiles = savesplit.listFiles();

for(int i=0;i<arrfiles.length;i++)

{

FileInputStream fin = new FileInputStream(arrfiles[i]);

byte[] byteArr = IOUtils.toByteArray(fin);

//

String sys = (String) gmr.allinsysno.get(nextnode);

String port = (String) gmr.allinport.get(nextnode);

try

{

Socket s = new Socket(sys, Integer.parseInt(port));

ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());

oos.writeObject("Downloadintermediate");

oos.writeObject(arrfiles[i].getName());

oos.writeObject(totalsplit);

oos.writeObject(streampath);

oos.writeObject(videoname);

oos.writeObject(vvv);

oos.writeObject(byteArr);

oos.close();

s.close();

sleep(1000);

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

File SplitVideoFile(String videoname, int totalsplit)

{

File splitFile=null;

try

{

File file = new File(videoname);//File read from Source folder to Split.

if (file.exists())

{

String videoFileName = file.getName().substring(0, file.getName().lastIndexOf(".")); // Name of the videoFile without extension

splitFile = new File(locf.getAbsolutePath()+"\\" + videoFileName);//Destination folder to save.

if (!splitFile.exists())

{

splitFile.mkdirs();

System.out.println("Directory Created -> " + splitFile.getAbsolutePath());

}

int i = 01;// Files count starts from 1

InputStream inputStream = new FileInputStream(file);

String videoFile = splitFile.getAbsolutePath() + "/" + String.format("%02d", i) + "\_" + file.getName();// Location to save the files which are Split from the original file.

OutputStream outputStream = new FileOutputStream(videoFile);

int totalPartsToSplit = totalsplit;// Total files to split.

int splitSize = inputStream.available() / totalPartsToSplit;

int streamSize = 0;

int read = 0;

while ((read = inputStream.read()) != -1)

{

if (splitSize == streamSize)

{

if (i != totalPartsToSplit)

{

i++;

String fileCount = String.format("%02d", i); // output will be 1 is 01, 2 is 02

videoFile = splitFile.getAbsolutePath() + "/" + fileCount + "\_" + file.getName();

outputStream = new FileOutputStream(videoFile);

// System.out.println("File Created Location: " + fileCount+"\_"+file.getName());

streamSize = 0;

}

}

outputStream.write(read);

streamSize++;

}

inputStream.close();

outputStream.close();

System.out.println("Total files Split ->" + totalPartsToSplit);

}

else

{

System.err.println(file.getAbsolutePath() + " File Not Found.");

}

}

catch (Exception e)

{

e.printStackTrace();

}

return splitFile;

}

void videodelete(File f)

{

String gateway = this.gatewayn;

final Timeline timeline2 = new Timeline();

timeline2.setCycleCount(Timeline.INDEFINITE);

timeline2.getKeyFrames().add(new KeyFrame(Duration.seconds(180),new EventHandler<ActionEvent>()

{

@Override

public void handle(ActionEvent event)

{

ssss="";

int count = 2;

Set set = allvideocount.entrySet();

Iterator it = set.iterator();

while (it.hasNext())

{

Map.Entry me = (Map.Entry) it.next();

String key = me.getKey().toString();

Vector value = (Vector)me.getValue();

if(count>=value.size())

{

allvideocount.remove(key);

System.out.println(key+": : : : : ; : ; allvideocount : : : : : : : : : : : "+allvideocount);

allvid.remove(key);

File ff = new File(f.getAbsolutePath()+"\\"+key);

ff.delete();

try

{

String str = "VIDEODELETE"+"$"+key + "$" + gateway;

System.out.println("string str = "+str);

InetAddress in = InetAddress.getByName("225.89.67.45");

MulticastSocket ms = new MulticastSocket(4567);

ms.joinGroup(in);

DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), in, 4567);

ms.send(dp);

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

for(int i=0;i<allvid.size();i++)

{

String ss = allvid.get(i).toString();

ssss = ssss + ss +"\n";

}

Platform.runLater(new Runnable()

{

@Override

public void run()

{

try

{

o.setavailablevideo(ssss);

}

catch (Exception ex)

{

}

}

});

}

},null));

timeline2.play();

}

}