```
import java.io.*;
import java.net.*;
import java.util.*;

public class StorageNode_UDP extends Thread {
                DatagramSocket ds;
                StorageNode_Multicast multicast;
                Hashtable<Integer, String> nodeTable;
                ScannerPorte scanner;
                int network;
                public StorageNode_UDP (DatagramSocket ds, StorageNode_Multicast multicast, ScannerPorte scanner,
int network) {
                        this.ds = ds;
                        this.multicast = multicast;
                        this.network = network;
                }

                public void run () {
                        StorageNode_UDPMonitor UDPM = new StorageNode_UDPMonitor();
                        List<DatagramPacket> list = new ArrayList();
                        StorageNode_UDPWriter UDPW = new StorageNode_UDPWriter(UDPM, ds, list, ds.getLocalPort());

                        try {
                                        UDPW.start(); //avvio il task assegnato al thread che gestisce le
connessione in ingresso UDP

                                        while (true) {
                                                        UDPM.StartRead(); //se supero questa linea, significa che
ho acquisito la lock
                                                        //quindi ho l'accesso esclusivo sulla lista delle socket
che hanno fatto richiesta
                                                        //leggo tutte le richieste

                                                        for (int i = 0;i < list.size();i++) { //estraggo l'i-esimo
pacchetto dalla lista
                                                                        DatagramPacket dp = list.get(i);
                                                                        ByteArrayInputStream bin = new
ByteArrayInputStream(dp.getData(), 0, dp.getLength());
                                                                        DataInputStream din = new DataInputStream
(bin);
                                                                        //cerco di capire se chi mi ha contattato
è un nuovo nodo oppure un client
                                                                        String type = din.readUTF();
                                                                        int start, end;
                                                                        //se mi ha contattato un nuovo nodo...

                                                                        if (type.equals("node")) {
                                                                                        int file_id = din.readInt
();
                                                                                        boolean alreadyexist =
false; //settata a true se l'id del nuovo nodo è già presente nella DHT
                                                                                        String fakeipSender =
din.readUTF();
                                                                                        //ricavo la porta del nodo
che mi ha contattato
                                                                                        start =
fakeipSender.indexOf("UDPport=");
                                                                                        end = fakeipSender.indexOf
(" TCPport=");
                                                                                        int portSender =
Integer.parseInt(fakeipSender.substring(start + 8, end));
                                                                                        //ricavo l'ip del nodo che
mi ha contattato
                                                                                        fakeipSender =
```

```
fakeipSender.substring(0, start - 1);

aggiornata dall'oggetto di multicast del mio nodo

multicast.getTable();

il predecessore del nodo e ne ricavo gli indirizzi

= nodeTable.keys();

minimo della tabella...


items.nextElement();


(items.hasMoreElements()) {

items.nextElement();


== file_id)

alreadyexist = true;


< min)

= actual;


actual)

= actual;


valore di partenza per il calcolo del pred e del succ




(items.hasMoreElements()) {

items.nextElement();


actual && actual < file_id)

pred = actual;


< actual && actual < succ)

succ = actual;


che il nuovo nodo ha l'id più grande di tutto l'anello

è il nodo con id più piccolo dell'anello
```

```
//ottengo la tabella

nodeTable =

//calcolo il successore e

Enumeration <Integer> items

//calcolo il massimo e il

int min, max, actual;
min = max =


while

                actual =


                if (actual



                if (actual

                        min


                if (max <

                        max

        }

//...e li assegno come

int pred = min, succ = max;

items = nodeTable.keys();

while

                actual =


                if (pred <



                if (file_id


        }

//se max<file_id significa

//quindi il suo successore

if (max < file_id) {
```

```
<min, il nuovo nodo è il nuovo primo elemento dell'anello

predecessore è il nodo con id più grande dell'anello



= "", string;

= 0;

IP e porta TCP del successore

(succ);

IP=");

UDPport=");

(start + 4, end);

TCPport=");

(string.substring(start + 9));

IP e porta TCP del predecessore

(pred);

IP=");

UDPport=");

(start + 4, end);

TCPport=");

(string.substring(start + 9));



//versione locale

InetAddress.getByName("localhost");

///versione di rete

InetAddress.getByName(fakeipSender);

= new ByteArrayOutputStream( );

new ObjectOutputStream(bout);

1"); //comunico al nuovo nodo di ricalcolare il proprio file_id
```

```
                succ = min;
        }

//viceversa, se file_id

//quindi il suo

if (file_id < min) {
                pred = max;
        }

String predIP = "", succIP

int predPort = 0, succPort

//estraggo dalla nodeTable

string = nodeTable.get

start = string.indexOf("

end = string.indexOf("

succIP = string.substring

start = string.indexOf("

succPort = Integer.parseInt

//estraggo dalla nodeTable

string = nodeTable.get

start = string.indexOf("

end = string.indexOf("

predIP = string.substring

start = string.indexOf("

predPort = Integer.parseInt

//invio i dati al nodo
InetAddress ia;

if (network == 0)

        ia =

else

        ia =


ByteArrayOutputStream bout


ObjectOutputStream out =


if (alreadyexist)
        out.writeObject("-
```

```
                                                                    else {

out.writeObject(predIP);

out.writeObject(succIP);

out.writeObject(predPort);

out.writeObject(succPort);

out.writeObject(nodeTable);
                                                                            }

                                                                    byte [] senddata =

bout.toByteArray();

                                                                    dp = new DatagramPacket

(senddata, senddata.length, ia, portSender);

                                                                    ds.send(dp);
                                                                    bout.reset();
                                                            }

                                                        //se mi ha contattato un client devo
contattare (tramite una nuova Socket TCP) il nodo che detiene il dato cercato
                                                        else { //ottengo l'id del dato che vuole
il client e il suo indirizzo ip + porta udp
                                                                    int id_data = din.readInt

();

                                                                    String fakeipClient =

din.readUTF();

                                                                    //ottengo dall'oggetto che
gestisce il multicast del nodo a cui appartengo la tabella aggiornata dei nodi sull'anello
                                                                    nodeTable =

multicast.getTable();

                                                                    //stabilisco quale nodo
detiene il dato cercato dal client

                                                                    Enumeration <Integer> items

= nodeTable.keys();

                                                                    //mi calcolo i nodi con
l'identificatore più piccolo e più grande nell'anello

                                                                    int max, min, actual;
                                                                    max = min =

items.nextElement();

                                                                    while

(items.hasMoreElements()) {
                                                                                actual =

items.nextElement();

                                                                                if (max <

actual)
                                                                                        max

= actual;

                                                                                if (actual

< min)
                                                                                        min

= actual;
                                                                            }

                                                                    int detentore = -1;
                                                                    //caso in cui il dato è
detenuto dal nodo più piccolo

                                                                    if (id_data < min || max <
```

```
id_data)
                                                                        detentore = min;
                                                                    else {
                                                                            items =
nodeTable.keys();
                                                                            detentore =
max; //supponiamo che il dato è detenuto dal nodo con id più grande
                                                                            while
(items.hasMoreElements()) {
actual = items.nextElement();
//se il nodo attualmente in esame è più "vicino" all'id del dato cercato rispetto al presunto detentore...

if (id_data < actual && actual < detentore) {
detentore = actual;
} //...si cambia il presunto detentore del dato
                                                                                }

                                                                            //caso
particolare: identificatore dato = identificatore nodo più grande...
                                                                            //il
detnetore è il nodo più piccolo dell'anello
                                                                            if (id_data
== max)
detentore = min;
                                                                        }

                                                                    if (detentore == -1) {
System.out.println("Errore nella determinazione del nodo detentore del dato");
                                                                            return ;
                                                                        }

                                                                    //ottengo gli indirizzi
del nodo detentore
                                                                    String indirizziDetentore =
nodeTable.get(detentore);
                                                                    start =
indirizziDetentore.indexOf("IP=");
                                                                    end =
indirizziDetentore.indexOf(" UDPport=");
                                                                    String fakeipDetentore =
indirizziDetentore.substring(start + 3, end); //mi ricavo l'indirizzo ip del detentore del dato
                                                                    start =
indirizziDetentore.indexOf("TCPport=");
                                                                    int TCPportDetentore =
Integer.parseInt(indirizziDetentore.substring(start + 8)); //e la porta su cui è in ascolto la socket TCP

                                                                    //creo una socket tcp con
cui contattare il detentore del dato
                                                                    Socket socketDetentore =
new Socket();
```

```
                                                                        if (network == 0)
socketDetentore.connect(new InetSocketAddress("LocalHost", TCPportDetentore));
                                                                        else
socketDetentore.connect(new InetSocketAddress(InetAddress.getByName(fakeipDetentore), TCPportDetentore));
                                                                        BufferedWriter writer = new
BufferedWriter(new OutputStreamWriter(socketDetentore.getOutputStream()));
                                                                        writer.write("client");
//comunico al nodo detentore che è un client che lo sta cercando
                                                                        writer.newLine();
                                                                        writer.write(id_data);
//comunico l'id del dato desiderato dal client
                                                                        writer.write(fakeipClient);
//ed anche l'indirizzo su cui rispondergli
                                                                        writer.newLine();
                                                                        writer.flush();
                                                                        socketDetentore.close();
                                                                } //fine else
                                                        } //fine for

                                                //elimino gli elementi presenti nella lista
                                                if (list.size() != 0)
                                                        for (int i = list.size() - 1;i >= 0;i--)
                                                                list.remove(i);

                                                UDPM.EndRead(); //rilascio la lock
                                        } //fine while(true)

                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                }
```