Gruppo 17

Università degli Studi di Bologna
Scuola di Ingegneria

Esercitazione 1

Scambio Righe Socket Java senza connessione

Corso di Reti di Calcolatori T

Daniele Magagnoli, Giulia Martina Bal,
Antonio Cassanelli, Gianmiriano Porrazzo

Anno Accademico 2021/2022

Specifiche dell'esercitazione:

- L'applicazione vuole dare la possibilità ai clienti di poter scambiare due righe di un file di testo.
- L'architettura prevede molti Client, residenti su più macchine, e un DiscoveryServer insieme ad una serie di RowSwapServer coresidenti.
- Il DiscoveryServer agisce da main lanciando tutti gli RowSwap (thread) e inoltre fornisce il collegamento di una coppia Client-RowSwapServer.
- Ogni Client fa la richiesta di scambio righe in modo ciclico e sempre per lo stesso file, dunque comunica sempre con lo stesso RowSwap.
- Ogni RowSwap server, al momento della creazione della socket, viene attivato e svolge l'algoritmo di scambio righe sul file indicato dall'utente all'invocazione del DiscoveryServer

DISCOVERY SERVER

Come prima cosa ci occupiamo dell'attivazione di tanti thread RowSwap quanti sono i file in ingresso, tramite metodo start.

```
//attivazione thread rowswap
for ( i=1; i<args.length; i+=2){
    new RowSwapServer( args[i], args[i+1]).start();
}
//fine attivazione thread
```

Dopodiché ci assicuriamo che le porte passate come argomento siano diverse fra loro

```
//Controllo che tutte le porte sono diverse
for ( i=2; i<args.length; i+=2){
    for(j=i+2; j<=args.length; j+=2 ){
        if(args[i].equals(args[j])){
            System.out.println("Porta "+i+" e porta "+j+" sono uguali\n");
            System.exit(1);
        }
    }
}
```

Lettura dal DatagramPacket del file passato dal cliente

```
biStream = new ByteArrayInputStream(packet.getData(), 0, packet.getLength());
diStream = new DataInputStream(biStream);
richiesta = diStream.readUTF();
st = new StringTokenizer(richiesta);
nomeFile = st.nextToken(); //nome file passato dal cliente
```

DISCOVERY SERVER

Controllo che il nome del file inviato dal Client sia un file gestito da un RowSwap verificando che sia presente tra gli argomenti passati in ingresso al Discorvery Server. Se il file non è presente la variabile porta è una stringa che indica l'esito negativo della richiesta

```
for(i=1; i<args.length-1 && ind<0; i+=2) {
    if(args[i].contentEquals(nomeFile)) {
        ind=i+1;
        porta=args[ind];
        }
}
if(ind < 0) {
    System.out.println("File name not found.");
}

boStream = new ByteArrayOutputStream();
doStream = new DataOutputStream(boStream);
doStream.writeUTF(porta);
data = boStream.toByteArray();
packet.setData(data, 0, data.length);
socket.send(packet);
```

Altrimenti la variabile porta conterrà il numero della porta del RowSwap a cui il Client dovrà mandare le righe da scambiare. La porta viene quindi messa nel DatagramPacket e viene fatta una send sulla socket.

ROWSWAP SERVER

Nel RowSwap Server ci occupiamo prima di tutto del recupero delle righe dal Client.

```
biStream = new ByteArrayInputStream(packet.getData(), 0, packet.getLength());
diStream = new DataInputStream(biStream);
richiesta = diStream.readUTF();
st = new StringTokenizer(richiesta);
numriga1 = Integer.parseInt(st.nextToken()); //nome file passato dal cliente
numriga2 = Integer.parseInt(st.nextToken());
```

Poi procediamo all'invocazione del metodo swap e a seconda del successo o meno dell'operazione la variabile esito conterrà una stringa da inviare in un secondo momento al Client.

```
if(this.swap(filename, numriga1, numriga2)) {
    esito="Scambio avvenuto con successo!\n";
} else esito="Errore nello scambio\n";

// preparazione della linea e invio della risposta
try {

    boStream = new ByteArrayOutputStream();
    doStream = new DataOutputStream(boStream);
    doStream.writeUTF(esito);
    data = boStream.toByteArray();
    packet.setData(data, 0, data.length);
    socket.send(packet);
}
catch (IOException e) {
```

```java
private boolean swap(String fileIn,int rig1,int rig2)throws IOException,FileNotFoundException {
    boolean res=true;
    if(rig1==rig2) {//stessa riga, non serve cambiare
        return false;
    }
    BufferedReader read=new BufferedReader(new FileReader(fileIn));
    int n=1;
    String riga=null;
    String r1=null;
    String r2=null;

    while(((riga=read.readLine()) != null) || (r1==null && r2==null)) { //se trovo r1 e r2 esco dal ciclo
        if(n==rig1) {
            r1=riga;
        }else if(n==rig2) {
            r2=riga;
        }
        n++;
    }
    read.close();

    read = new BufferedReader(new FileReader(fileIn));
    n=1;
    StringBuilder s=new StringBuilder();
    if(r1==null|| r2==null) {
        res=false;
    }else {
        while((riga = read.readLine()) != null) {
            if(n == rig1) {
                s.append(r2 + "\n");
            }else if(n == rig2) {
                s.append(r1 + "\n");
            }else {
                s.append(riga + "\n");
            }
            n++;
        }
        read.close();
        BufferedWriter write = new BufferedWriter(new FileWriter(fileIn));
        write.write(s.toString());
        write.close();
    }
    return res;
}
```

ROWSWAP SERVER

La funzione swap effettua lo scambio righe passategli dal Client. Legge una prima volta il file trovando le righe da scambiare, poi dopo averle memorizzate rilegge una seconda volta il file incapsulando tutte le righe in uno StringBuilder con le righe scambiate di posizione. Infine il file viene sovrascritto. Inoltre facendo test con dei file molto grandi, a causa dell'uso della StringBuilder, viene lanciata l'eccezione OutOfMemory.

SWAPCLIENT

Il Client apre una socket per comunicare con il Discovery Server: gli passa il nome del file su cui operare per poi ricevere la porta del RowSwap relativa al file che ha chiesto. Se la porta è negativa significa che il DS non ha trovato il file.

```
try {
    socket = new DatagramSocket();
    socket.setSoTimeout(30000);
    packet = new DatagramPacket(buf, buf.length, addr, port);
    System.out.println("Creata la socket con il DS: " + socket);

    boStream = new ByteArrayOutputStream();
    doStream = new DataOutputStream(boStream);
    doStream.writeUTF(args[2]); //invio del nome del file
    packet.setData(boStream.toByteArray());
    socket.send(packet);
    System.out.println("Richiesta inviata a " + addr + ", " + port);

} catch (IOException e) {
    System.out.println("Problemi nella creazione della socket o nell'invio del pacchetto:
    e.printStackTrace();
    System.exit(1);
}

try {
    String porta = null;
    packet.setData(data);
    socket.receive(packet);
    //recupero esito del DS
    biStream = new ByteArrayInputStream(packet.getData(), 0, packet.getLength());
    diStream = new DataInputStream(biStream);
    porta = diStream.readUTF();
    System.out.println(porta);
    portRS = Integer.parseInt(porta);
    if(portRS < 0) {
        System.out.println("Errore nella richiesta della porta.\n");
        socket.close();
        System.exit(1);
    }
```

```
System.out.println("Numero delle righe da scambiare? ");
while ((righe=stdIn.readLine()) != null) {
    // interazione con l'utente e invio richiesta
    try {
        boStream = new ByteArrayOutputStream();
        doStream = new DataOutputStream(boStream);
        doStream.writeUTF(righe);
        buf = boStream.toByteArray();
        packet.setData(buf);
        socket.send(packet);
        //System.out.print("Numero delle righe da scambiare? ");

        //System.out.println("Richiesta inviata a " + addr + ", " + portRS);

    } catch (Exception e) {
        System.out.println("Problemi nell'invio della richiesta: ");
        e.printStackTrace();
        System.out
          .println("\n^D(Unix)/^Z(Win)+invio per uscire");
        continue;
    }

    try {
        //attesa risposta
        packet.setData(data);
        socket.receive(packet);

        //stampa esito
        biStream = new ByteArrayInputStream(packet.getData(), 0, packet.getLength());
        diStream = new DataInputStream(biStream);
        System.out.println(diStream.readUTF() + "\n");

    } catch (IOException e) {
        System.out.println("Problemi nella ricezione del datagramma: ");
        e.printStackTrace();
        System.out.println("\n^D(Unix)/^Z(Win)+invio per uscire");
        continue;
        // il client continua l'esecuzione riprendendo dall'inizio del ciclo
    }


    // tutto ok, pronto per nuova richiesta
    System.out.println("Numero delle righe da scambiare?");
    System.out.println("\n^D(Unix)/^Z(Win)+invio per uscire");
}
```

SWAPCLIENT

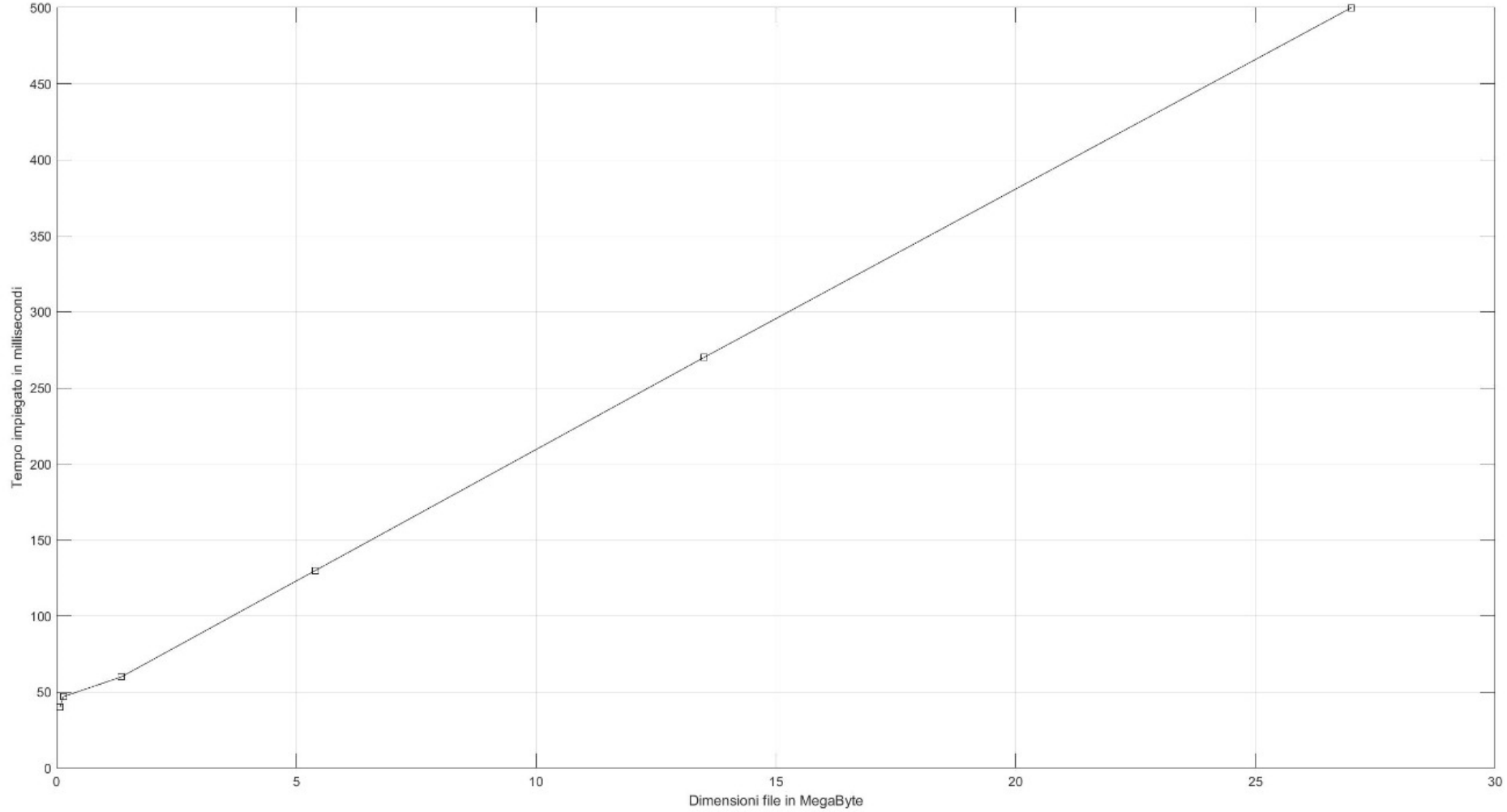
Dopo aver chiuso la socket con il DS il client apre una seconda socket con il RowSwap sulla porta che ha ricevuto dal DS per potergli comunicare le righe da scambiare e ricevere l'esito dell'operazione.
Il Client ha un comportamento ciclico, continua a chiedere lo scambio di righe finché non preme ^D(Unix)/^Z(Win) e invio per uscire.

Osservazioni finali:

Eseguendo vari test sul progetto, considerando un client alla volta che dialoga con il server e prendendo il tempo nel RowSwap Server si ottengono i seguenti dati:

- File di 1,35 Mbyte: primo scambio 200 ms
  richieste successive 60 ms
- File di 5,4 Mbyte: primo scambio 400 ms
  richieste successive 130 ms
- File di 13,5 Mbyte: primo scambio 700 ms
  richieste successive 270 ms
- File di 27 Mbyte: primo scambio 1300 ms
  richieste successive 500 ms

Eseguiti su macchina virtuale 4gb RAM, i7-770HQ, 2.80 ghz


Tempo impiegato in millisecondi
Dimensioni file in MegaByte
0
50
100
150
200
250
300
350
400
450
500
0
5
10
15
20
25
30

GRAZIE PER L’ATTENZIONE