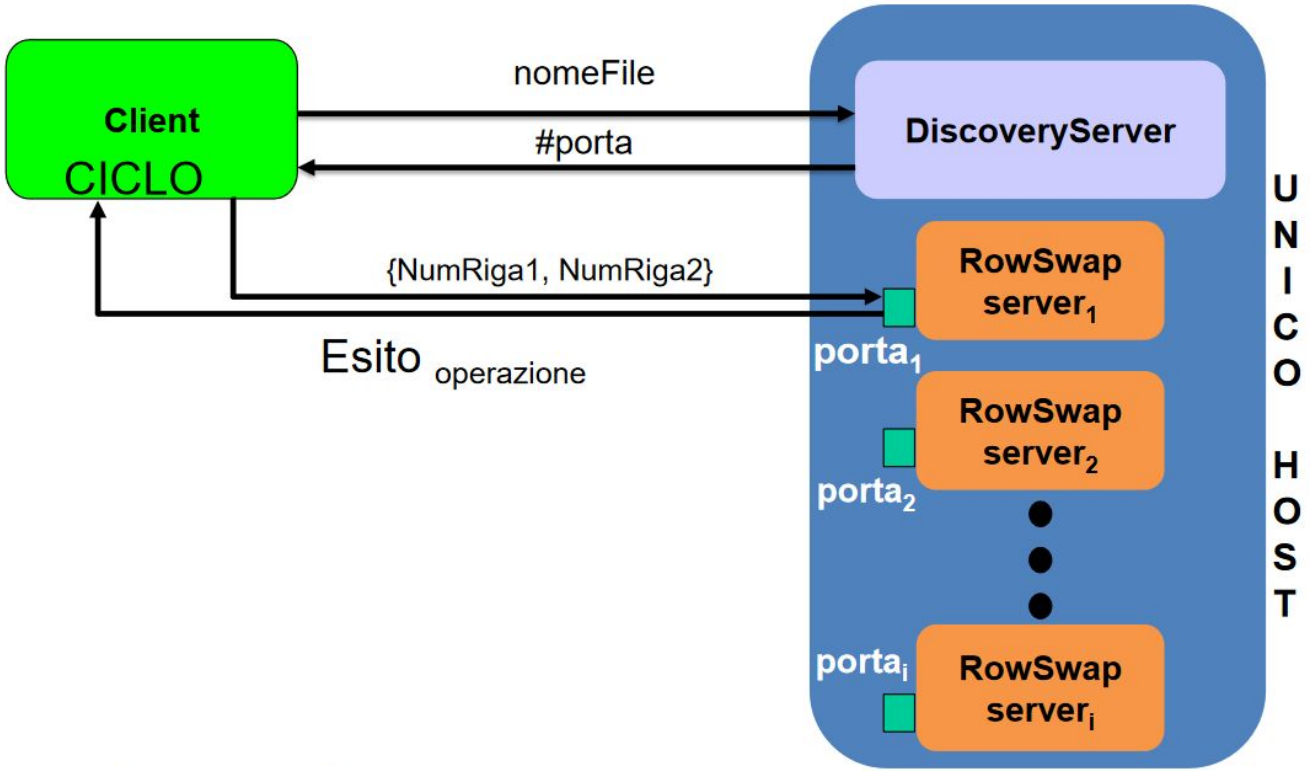
Esercitazione Lab 1

Socket Java senza connessione

19 ottobre 2021

Bonantini Alessandro
Galeone Christian
Garcia Fernandez Diego
Piras Gabriele
Predieri Nicholas

Architettura Problema



Discovery Server

- attiva un RS thread per ogni file passato come argomento
- invia al Client il **numero della porta** sulla quale è in esecuzione il RowSwap Server di interesse o **esito negativo**, se il file indicato non è disponibile

Client

- invia al Discovery Server il **nome del file** di cui vuole scambiare le righe
- invia al RowSwap Server i **numeri** relativi alle **righe** che vuole scambiare

RowSwap Server

- esegue la procedura ***scambiaRighe*** nel file indicato dal client
- stampa a video l'esito dell'operazione

Proposta Codice

Cliente (1)

```
/* Controllo che tutti file e le porte siano diversi/creazione pacchetto con i due
numeri di riga da inviare a RS Server */

            try {

                socketRS = new DatagramSocket();

                socketRS.setSoTimeout(30000);

                packetRS = new DatagramPacket(buf, buf.length, inetAddress,
portRS);

                System.out.println("\nRSS interrogato");

                System.out.println("Creata la socket: " + socketRS);

            }
```

Cliente (2)

```
// invio pacchetto a RS Server
try {
                boStream = new ByteArrayOutputStream();
                doStream = new DataOutputStream(boStream);
                doStream.writeUTF(richiesta);
                data = boStream.toByteArray();
                packetRS.setData(data);
                socketRS.send(packetRS);
                System.out.println("Richiesta inviata a " + inetAddress + ":" +
portRS);
            }
```

Discovery Server (1)

```
    RowSwapServer RSServers[] = new RowSwapServer[k]; //  Creazione k RowSwapServers
    try {    for(int i = 0; i < k; i++){
            //  Run di tutti i thread associati ai files
            RSServers[i] = new RowSwapServer(files[i], new DatagramSocket(ports[i]));
            RSServers[i].start();
            System.out.println("RSServer #" + i + ": avviato su porta " + ports[i] +"
con file " + files[i]);
        }
    } catch (SocketException s) {
        System.out.println("Errore Socket");}
```

Discovery Server (2)

```
//Controllo che tutte le porte siano diverse (concetto simile per i file)
private static boolean errorePorte(int[] ports) {
       boolean isErrore=false;
       int[] copyPorts=Arrays.copyOf(ports,ports.length);
       Arrays.sort(copyPorts);
       for(int i = 1; i< copyPorts.length; i++){
           if(copyPorts[i]== copyPorts[i-1] || copyPorts[i]<=1024 ||
copyPorts[i]>65535){
               isErrore= true;
               return isErrore;
           }
        }
        return isErrore;
}
```

Funzione *ScambiaRighe (1)*

```
int numLinee = 0; //  Conteggio dimensione necessaria per inizializzare l'array
                while ((linea=bf.readLine())!=null){
                    numLinee++; }
                linee = new String[numLinee];
                if(indexLine1<=0 ||indexLine1>numLinee){
                    System.out.println("Errore"); }
                if(indexLine2<=0 || indexLine2>numLinee){
                    System.out.println("Errore");
}
```

Funzione *ScambiaRighe (2)*

```
String temp;  //  Scambio

temp = linee[indexLine1];

linee[indexLine1] = linee[indexLine2];;

linee[indexLine2] = temp;
     for(int i = 0; i < linee.length; i++) { //  Scrittura file di output

             fileWriter.write(linee[i]);

     }

//  Stampa su stdout

boStream = new ByteArrayOutputStream();

doStream = new DataOutputStream(boStream);
```

Funzione *ScambiaRighe (3)*

```
    for (int i = 0; i<linee.length; i++) {
                    doStream.writeChars(linee[i]);
                    data = boStream.toByteArray();
                    packet.setData(data, 0, data.length);
                    socket.send(packet);
          }
bf.close();
fileReader.close();
fileWriter.close();
System.out.println("Success");
```