

# Reti di Calcolatori T

## Esercitazione 1

**GRUPPO 5:**  
Stefano Lentini  
Davide Perinti  
Leone Ruggiero  
Marco Turzi  
Andrea Zanni

Per la risoluzione dell'esercizio viene utilizzato il modello Client/Server e la comunicazione tra queste due entità avviene tramite le DatagramSocket le quali si appoggiano sul protocollo UDP (connection-less).

Il Client richiede lo scambio di due righe in un file remoto sul DiscoveryServer. Il DiscoveryServer crea un thread per ogni file da gestire e informa il Client sulla porta corretta da usare per richiedere lo scambio delle righe effettuato dal SwapRowServer(Thread) del file richiesto.

# SwapRowServer(1)

- Il **COSTRUTTORE** si occupa di inizializzare uno dei Thread SwapRowServer con i parametri (nomefile, porta) passati dal DiscoveryServer, la socket ed il packet utilizzati per ricevere la richiesta di scambioRighe dal Cliente ed inviare il risultato.

```
public SwapRowServer (String filename, int port) {  
    this.filename = filename;  
    this.port = port;
```

```
    try {  
        this.socket = new DatagramSocket(port);  
    } catch (SocketException e) {  
        e.printStackTrace();  
        System.exit(1);  
    }
```

Gestiamo la SocketException nel caso in cui il DiscoveryServer (che comunque controlla il parametro port) passi un parametro non valido.

```
    buf = new byte[256];  
    this.packet = new DatagramPacket(buf, buf.length);  
}
```

# SwapRowServer(2)

- Siccome il nostro SwapRowServer viene invocato come Thread dal DiscoveryServer, l'esecuzione avviene attraverso **il metodo run()** che **si occupa di attendere la richiesta di scambio da un cliente, ottenere il numero delle due righe richieste dal packet ricevuto ed eseguire lo scambio delle righe** tramite il metodo `scambiaRighe(riga1, riga2)` **che restituisce il risultato dello scambio, inviandolo poi al cliente.**

```
packet.setData(buf);
    try {
        socket.receive(packet);
    } catch (IOException e1) {
        e1.printStackTrace();
        System.exit(2);
    }

    ByteArrayInputStream biStream = new ByteArrayInputStream(packet.getData(), 0,
    packet.getLength());

    DataInputStream diStream = new DataInputStream(biStream);
    String richiesta = null;

    try {
        richiesta = diStream.readUTF();
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(2);
    }

    StringTokenizer st = new StringTokenizer(richiesta);
    try {
        riga1 = Integer.parseInt(st.nextToken());
        riga2 = Integer.parseInt(st.nextToken());
    } catch (NumberFormatException e) {
        System.out.println("Errore nel passaggio delle righe");
        result=4;
    } catch (NoSuchElementException e) {
        System.out.println("Errore riga mancante");
        result=4;
    }

    if(result!=4) {
        result = scambiaRighe(riga1, riga2);
    }

    ByteArrayOutputStream boStream = new ByteArrayOutputStream();
    DataOutputStream doStream = new DataOutputStream(boStream);

    try {
        doStream.writeUTF(Integer.toString(result));
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(2);
    }

    data = boStream.toByteArray();
    packet.setData(data);

    try {
        socket.send(packet);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(2);
    }
```

Gestisce il caso in cui il Client formato sia errato, oppure non abbia passato una delle due righe. Evitando di eseguire lo scambio delle righe comunicandolo al Client.

# SwapRowServer(3)

- Il metodo `scambiaRighe` Utilizza un `BufferedReader` per leggere dal file richiesto dal cliente le due righe da scambiare, con i dovuti controlli\* poi crea un file temporaneo nel quale riscrivere il file originale con le righe scambiate. Se tutto va come previsto, elimino il file originale e rinomino il file temporaneo.

```
for(int i=1; i<=(nRiga1>nRiga2 ? nRiga1 : nRiga2); i++) {  
    try {  
        if((line = br.readLine())==null) {  
            System.out.println("Numero riga errata.");  
            return 3;  
        }  
    } catch (IOException e) {  
        System.out.println("Errore nella lettura della linea n° "+ i);  
        try {  
            br.close();  
        } catch (IOException e1) {  
            System.out.println("Errore chiusura BufferedReader");  
        }  
        return 2;  
    }  
    if(i==nRiga1)  
        riga1 = line;  
    else if(i==nRiga2)  
        riga2 = line;  
}
```


Eseguo il ciclo finché non raggiungo l'indice maggiore delle due righe richieste dal Client.

Controllo che il Client abbia richiesto delle righe esistenti nel file richiesto.

# SwapRowServer(3bis)

```
int i=1;
try {
    while((line=br.readLine())!=null) {
        if(i==nRiga1) {
            bw.write(riga2);
            bw.newLine();
        }
        else if(i==nRiga2) {
            bw.write(riga1);
            bw.newLine();
        }
        else {
            bw.write(line);
            bw.newLine();
        }
        i++;
    }
} catch (IOException e) {
    System.out.println("Errore nella SCRITTURA delle linee sul file temporaneo.");
    try {
        bw.close();
        br.close();
    } catch (IOException e1) {
        System.out.println("Errore chiusura BufferedReader/BufferedWriter");
    }
    return 2;
}
```

Trascrivo dal file originale  
**tutte** le righe presenti sul file  
temporaneo, scambiando le  
due righe salvate nel ciclo  
precedente.



**Elimino il file originale** per  
**sostituirlo** con il file temporaneo  
in cui le due righe sono state  
scambiate nel ciclo precedente.

```
try {
    Files.delete(Paths.get(this.filename));
} catch (IOException e) {
    System.out.println("Errore durante l'eliminazione del file originale.");
    return 2;
}
temp.renameTo(new File(this.filename));
System.out.println("Scambio avvenuto con successo");
return 0;
```

# DiscoveryServer

## Funzionamento

- Il D.S. conserva una tabella in cui ciascun record e' costituito dalla coppia (nomeFile, numero di porta).
- Per ogni elemento della tabella genera un thread RowSwapServer.
- Alla richiesta del Client risponde con il numero di porta del thread che gestisce il file richiesto. In caso il file richiesto non sia presente nella tabella, il D.S. risponde al Client con un intero negativo (-1).

# Discovery Server

## Problematiche

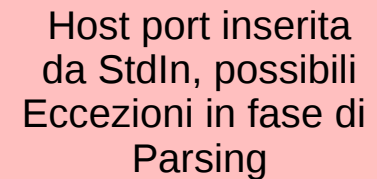
- Per garantire un corretto funzionamento del D.S. e' necessario, come prima cosa, controllare che il contenuto della tabella sia formalmente accettabile.
- Il controllo dell' unicità dei file in tabella e' garantito dall' implementazione (Hashtable), mentre e' necessario controllare che non ci siano numeri di porta ripetuti.
- In caso di errore nella costruzione della tabella, viene stampata a video la causa e il server si chiude.
- Se invece la tabella e' stata costruita correttamente il Server crea i Thread RowSwap, poi entra in un ciclo infinito in cui esegue le seguenti operazioni in sequenza:

```
if(((args.length-1)%2)!=0&&args.length>=3)...
```

```
if(hostPort<1024)...
```

```
catch(NumberFormatException e)
{
...
}
```

Host port inserita  
da StdIn, possibili  
Eccezioni in fase di  
Parsing



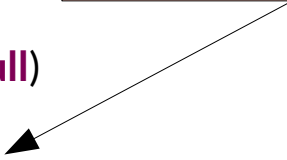
```
if(portNum==hostPort||portNum<1024)...
```

```
else if(table.containsValue(portNum))...
```

- Attesa del pacchetto dal Client.

```
while(true) {  
    packet.setData(buf);  
    Socket.receive(packet);  
    ...}  
}
```

Il Client interpreta il "-1"  
come File not found



- Consultazione della tabella per cercare l'elemento richiesto.

```
if(table.get(nomeFile)==null)  
{  
    doStream.writeUTF("-1");  
}  
else  
{  
    doStream.writeUTF(table.get(nomeFile).toString());  
}
```

- Invio risposta al Client.

- Qualsiasi eccezione non imputabile all'utente che invoca il D.S. (NumberFormatException, Socket, ecc..) viene gestita in questo modo:

```
catch (Exception e)  
{  
    e.printStackTrace();  
    System.exit(CODICE ERRORE);  
}
```



# C L I E N T

- Il client ha il compito di stabilire la connessione con DiscoveryServer indicandolo quale file modificare.
- Dopo aver inviato a DiscoveryServer il nome del file, viene chiesto all'utente di inserire le righe da scambiare. Tale informazione verrà poi inviata all'RSserver che eseguirà lo scambio delle righe.
- Infine attenderà che RSserver abbia finito il suo compito e il ricevimento dell'esito dell'operazione.

# C O N T R O L L I

## Problematiche

- Metodi come writeUTF, parseInt, ecc.. possono lanciare eccezioni
- Il file richiesto potrebbe non esistere nella directory di Discovery.

## Soluzioni

- Risolto il problema tramite l'utilizzo di blocchi try/catch

```
try {  
    doStream1.writeUTF(lineeSwap);  
} catch (IOException e) {  
    System.out.println("Errore writeUTF");  
    e.printStackTrace();  
    System.exit(5);};
```

- Risolto tramite il controllo sul dato ricevuto da Discovery contenente il numero di porta di Rserver.

```
if(portaRS==-1) //il file non esiste  
  
{  
    System.out.println("il file non esiste");  
    System.exit(-1);  
}
```

- L'utente potrebbe non inserire nessuna riga

- RSserver potrebbe commettere degli errori durante lo svolgimento del suo compito

- Risolto inserendo un controllo sulla stringa letta da standard input

```
if (lineeSwap != null ) {
```

```
.....
```

```
}else{
```

```
System.out.println("Errore inserimento righe");
```

```
System.exit(1);
```

```
}
```

- Dalla risposta ricevuta da RS, il programma stampa a video la tipologia di problema che si è presentato e termina.

```
if(risposta.equals("0")) {
```

```
System.out.println("Successo");
```

```
System.exit(0);
```

```
}if (risposta.equals("1")){
```

```
Syste.out.println("File non trovato");
```

```
System.exit(1);
```

```
}if (risposta.equals("2")){
```

```
System.out.println("Errore elaborazione RSserver");
```

```
System.exit(2);
```

```
}if(risposta.equals("3")){
```

```
System.out.println("Errore inserimento righe");
```

```
System.exit(3);
```

```
}if(risposta.equals("4")){
```

```
System.out.println("Errore inserimento righe: è stata inserita una sola riga");
```

```
System.exit(4);
```

```
}
```