Reti di Calcolatori 2014 – 2015

Gabriele D'Angelo
<gda@cs.unibo.it>
http://www.cs.unibo.it/gdangelo/

Corso di Laurea Triennale in
Scienze e Tecnologie Informatiche

Cesena

Revisione 1213.1

Esempio: client / server con socket TCP

- **Client:**
  - Socket TCP che usa strutture di invio e ricezione di basso livello
  - OutputStream e InputStream con metodi .read() e .write()
- **Server:**
  - Socket TCP che usa strutture di invio e ricezione di basso livello
  - OutputStream e InputStream con metodi .read() e .write()

Scaricare il codice e commentarlo

•**Esercitazione 1:** client e server TCP

- **Scopo**: uso di costrutti di basso livello con socket TCP

1) Compilare ed eseguire TCPClient e TCPServer:

   a) Testarne il funzionamento

      a) descrivere il comportamento del client?

      b) descrivere il comportamento del server

      c) che cosa è cambiato dalla prima esercitazione?

2) Eseguire due distinti Client:

   a) Accedere contemporaneamente al server con i due client, verificarne il comportamento

      a) Client1 avvio, Client2 avvio, Client1 trasmissione input, Client2 trasmissione input

      b) Client1 avvio, Client2 avvio, Client2 trasmissione input, Client1 trasmissione input

   b) Quale livello di concorrenza è stato realizzato?

•**Esercitazione 2:** server con thread

- **Scopo**: programmazione concorrente del server

1) Compilare ed eseguire TCPClient e TCPServer:

   a) Testarne il funzionamento

      a) descrivere il comportamento del client?

      b) descrivere il comportamento del server

      c) che cosa è cambiato dalla prima esercitazione?

2) Eseguire due distinti Client:

   a) Accedere contemporaneamente al server con i due client, verificarne il comportamento

      a) Client1 avvio, Client2 avvio, Client1 trasmissione input, Client2 trasmissione input

      b) Client1 avvio, Client2 avvio, Client2 trasmissione input, Client1 trasmissione input

   b) Quale livello di concorrenza è stato realizzato?

•**Esercitazione 3:** server con thread

- **Scopo**: interoperabilità tra socket

1) Compilare ed eseguire TCPClient, come server usare il server TCP della prima esercitazione:

   a) Testarne il funzionamento più volte chiedendo stringhe di lunghezza variabile

      a) descrivere il comportamento del client? È deterministico?

      b) descrivere il comportamento del server?

      c) che cosa è cambiato dalla prima esercitazione?

•**Esercitazione 3.1:** server con thread

- **Scopo**: interoperabilità tra socket

1) Modificare il codice ed aggiunere il '\n' finale alla stringa letta da input

   a) *sendStream.write("\n".getBytes()); // aggiugo il CR finale*

2) Testarne il funzionamento più volte chiedendo stringhe di lunghezza variabile

   a) descrivere il comportamento del client? È deterministico?

   b) descrivere il comportamento del server?

   c) che cosa è cambiato dalla prima esercitazione?

•Esercitazione 3.2: client TCP

```
void getResponse ()
{
        try
        {
                int dataSize = 1;
                byte [] recvBuff = new byte [1024];
                response = new String();
                while(dataSize > 0)
                {
                        dataSize = recvStream.read (recvBuff, 0, 1024);
                        String buff_read = new String (recvBuff, 0, dataSize);
                        System.out.println("Buffer read: " + buff_read + '\n');
                        response = response + buff_read;
                }
        }
         catch (…
```

- Cosa è cambiato?
- La stringa è letta correttamente?
- Il client termina correttamente la sua esecuzione?

Esercitazione 3.3: client TCP

```
void getResponse ()
{
        try
        {
                int dataSize = 1;
                byte [] recvBuff = new byte [1024];
                response = new String();
                while(dataSize > 0)
                {
                        dataSize = recvStream.read (recvBuff, 0, 1024);
                        String buff_read = new String (recvBuff, 0, dataSize);
                        System.out.println("Buffer read: " + buff_read + '\n');
                        response = response + buff_read;
                        if (recvBuff[dataSize-1] == '\n') dataSize = -1;
                }

        }
```

- Cosa è cambiato?
- La stringa è letta correttamente?
- Il client termina correttamente la sua esecuzione?

Esercitazione 3: Client TCP

1. Analizzare il comportamento del client e del server
    1. *con i buffer per l'invio e ricezione di stringhe*
    2. *con l'uso dei comandi di più basso livello read e write*
2. il nuovo client è equivalente a quello visto nella prima esercitazione?
3. usare la .readline() per leggere dati provenienti da una socket quale astrazione ci impone. È generale?
4. il server si comporta come aspettato?

Esercitazione 4: socket TCP

- **Scopo**: aggiungere richieste multiple al client TCP

1) Modificare il client in modo che possa continuare ad inviare richieste fino a quando l'utente non invia la stringa "`quit`".

   a) Usare una variabile Boolean controllata ad ogni cliclo while

      a) *Boolean finished = false;*
      b) *while (finished == false)*
      c) *sentence.compareTo("quit") == 0*

2) Verificarne il funzionamento, il client riesce ad inviare più stringhe?

3) Il server risponde correttamente, perché?

Reti di Calcolatori 2014 – 2015

Gabriele D'Angelo
<gda@cs.unibo.it>
http://www.cs.unibo.it/gdangelo/

Corso di Laurea Triennale in
Scienze e Tecnologie Informatiche

Cesena

Revisione 1213.1