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: sempice HTTP client/server

- **Client:**
  - Semplice client HTTP che esegue richieste a server web
    - *Uso di socket TCP*
- **Server:**
  - Semplice server HTTP che invia pagine html a browser di uso comune
    - *Uso di socket TCP*

Scaricare il codice e commentarlo

Applicazione d'esempio: **client HTTP**

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

public class HTTPClient {
  public static void main(String[] args) {

if ((args.length != 1) && (args.length != 2))
        System.out.println("Wrong number of arguments");

OutputStream to_file;
if (args.length == 2)
        to_file = new FileOutputStream(args[1]);
else
        to_file = System.out;

URL url = new URL(args[0]);
String protocol = url.getProtocol();
if (!protocol.equals("http"))
        System.out.println("URL must use 'http:' protocol");

String host = url.getHost();
int port = url.getPort();
if (port == -1) port = 80;
String filename = url.getFile();
```

controllo i parametri

crea l'output su file o system.out

controlla che il protocollo sia http

estrae la porta, se non esiste 80

Applicazione d'esempio: **client HTTP**

creo la socket

buffer di ingresso

invia la request

buffer ricezione

chiudo socket e file

```
String filename = url.getFile();

Socket socket = new Socket(host, port);

InputStream from_server = socket.getInputStream();
PrintWriter to_server =
new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));

to_server.println("GET " + filename);
to_server.flush();

byte[] buffer = new byte[4096];
int bytes_read;
while((bytes_read = from_server.read(buffer)) != -1)
to_file.write(buffer, 0, bytes_read);

socket.close();
to_file.close();

  }
}
```

•**Esercitazione 1:** clientHTTP

- **Scopo**: richiedere una pagina web con una socket TCP

1) Eseguire il codice Client:

   a) Testarne il funzionamento richiedendo una pagina
      a) *java HTTPClient http://137.204.107.67/ test.html*
   b) Verificare che nel file test.html ci sia la pagina richiesta?

2) Eseguire wireshark e catturare il traffico

   a) Identificare le parti componenti della richiesta/risposta in TCP
      a) *connessione, scambio dati, termine connessione*
      b) *Identificare il pacchetto con il bit di psh=1*

•**Esercitazione 2:** clientHTTP

- **Scopo**: richiedere una pagina web con una socket TCP

1) Aggiungere il codice:

   a) stampare a video l'host e il filename richiesto al web server

   b) tradurre il nome di dominio a indirizzo IP

      a) InetAddress IPAddress = InetAddress.getByName(host);

2) Eseguire il codice Client e catturare il traffico:

   a) Testarne il funzionamento richiedendo una pagina

      a) java HTTPClient http://unibo.it/it unibo.html

   b) Identificare le parti componenti della richiesta/risposta in TCP

      a) connessione, scambio dati, termine connessione

      b) Identificare il pacchetto con il bit di psh=1

Esempio: sempice HTTP client/server

- **Client:**
  - Semplice client HTTP che esegue richieste a server web
    - *Uso di socket TCP*
- **Server:**
  - Semplice server HTTP che invia pagine html a browser di uso comune
    - *Uso di socket TCP*

Scaricare il codice e commentarlo

Applicazione d'esempio: **server HTTP**

Socket server

.accept() e creo i buffer

leggo la richiesta

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

public class HTTPServer {

public static void main(String[] args) throws Exception {
        // creazione della socket
        int port = 6789;
        ServerSocket serverSocket = new ServerSocket(port);
        System.err.println("Sever web avviato sulla porta: " + port);

        while (true) {
                Socket clientSocket = serverSocket.accept();
                System.err.println("Nuova richiesta");

                BufferedReader in = new BufferedReader(
                        new InputStreamReader(
                                clientSocket.getInputStream()));
                BufferedWriter out = new BufferedWriter(
                        new OutputStreamWriter(
                                clientSocket.getOutputStream()));

                String s;
                while ((s = in.readLine()) != null) {
                        System.out.println(s);
                        if (s.isEmpty()) {
                                break;
                        }
                }
```

Applicazione d'esempio: **server HTTP**

Scrivo sul buffer di uscita una pagina HTTP (protocollo applicativo)

```
out.write("HTTP/1.0 200 OK\r\n");
out.write("Date: Fri, 31 Dec 1999 23:59:59 GMT\r\n");
out.write("Server: Apache/0.8.4\r\n");
out.write("Content-Type: text/html\r\n");
out.write("Content-Length: 49\r\n");
out.write("Expires: Sat, 01 Jan 2000 00:59:59 GMT\r\n");
out.write("Last-modified: Fri, 31 Dic 1999 23:59:59 GMT\r\n");
out.write("\r\n");
out.write("<TITLE>Exemple</TITLE>");
out.write("<P>Pagnina di esempio</P>");
```

chiusura

```
System.err.println("Richiesta terminata");
out.close();
in.close();
clientSocket.close();
        }
    }
}
```

•**Esercitazione 3:** serverHTTP

- **Scopo**: semplice server web che usa le socket TCP per inviare pagine HTML

1) Eseguire il server:

   a) Testarne il funzionamento usando un browser e analizzare la richiesta del client

      a) iexplorer, firefox, crome, etc…

2) Eseguire più richieste al server per differenti pagine html

   a) Analizzare come cambia il protocollo HTTP alla modifica della pagina richiesta

Esercitazione 4: serverHTTP

```
String content = null;
String FileName = null;
<…>
String[] request = s.split(" ");
// se la stringa inizia con GET significa che è la start line
// prendo il file richiesto dal client e tolgo il '/' iniziale
if (request[0].equals("GET") ) FileName = request[1].substring(1);
 <…>
File file =new File(FileName );
double filelength = file.length();
int bytes = (int) filelength;
FileReader reader = new FileReader(file);
char[] chars = new char[bytes];
reader.read(chars);
content = new String(chars);
reader.close();

// Modificare il codice per inserire la pagina letta nel buffer di uscita
out.write("...
out.write("...
out.write("...
…
```

- Aggiungere il codice per:
  - fare il parsing della richiesta e trovare il file html da restituire
  - leggere il file dal fiesystem ed inviarlo al client

•**Esercitazione 4:** serverHTTP

- **Scopo**: semplice server web che usa le socket TCP per inviare pagine HTML

1) Eseguire il server:

   a) Testarne il funzionamento usando un browser richiedendo la pagina unibo.html precedentemente scaricata

      a) *http://localhost:6789/unibo.html*

2) Testare il server con richieste di pagine strutturate nel flesystem

   a) Creare una cartella del filesystem di nome 'it' e copiare il file unibo.html e testare la richiesta

      a) *http://localhost:6789/it/unibo.html*

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