



Università degli Studi di Bologna
Facoltà di Ingegneria

Corso di Reti di Calcolatori T

Esercitazione 2 (proposta) *Socket Java con connessione*

Luca Foschini

Anno accademico 2013/2014

Esercitazione 2 1

Specifica Client

Sviluppare un'applicazione C/S che effettui il **trasferimento di tutti i file di un direttorio dal client al server** (multiple put).

Il client chiede all'utente il **nome del direttorio**, relativo al direttorio corrente dove viene lanciato il client, si connette al server con una socket connessa (con `java.net.Socket`), usandone lo stream di output attraverso cui **inviare informazioni**, e lo stream di input da cui **ricevere il comando di attivazione del trasferimento per ciascun file**: **"attiva"** nel caso un file con quel nome non sia presente sul file system del server con quel nome, esito negativo (ad esempio **"salta file"**) altrimenti.

I file richiesti vengono **salvati nel direttorio corrente** sul server e il protocollo indicato evita che vengano sovrascritti file esistenti che abbiano lo stesso nome.

Esercitazione 2 2

Specifica Server

Il server attende una richiesta di connessione da parte del client (su `java.net.ServerSocket`), usa la socket (`java.net.Socket`) prodotta dalla connessione per creare uno stream di input **da cui ricevere i nomi dei file e il contenuto dei file**, e uno stream di output su cui **inviare il comando di attivazione trasferimento**, anche per più richieste di multiple put dallo stesso client. Si noti che si **deve utilizzare la stessa socket per il trasferimento di tutti i file del direttorio**.

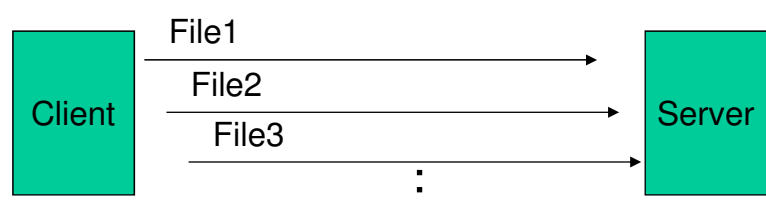
Il server deve essere realizzato come **server concorrente e parallelo**. Per ogni nuova richiesta ricevuta il processo padre, dopo aver accettato la richiesta, attiva un processo figlio a cui affida il completamento del servizio richiesto.

Esercitazione 2 3

Note per la realizzazione

Per il trasferimento dei file bisogna studiare un **protocollo per la gestione del trasferimento multiplo**. Ad esempio il server, **prima dell'invio del singolo file**, può inviare il **nome del file** e il **numero di byte** da cui il file è composto, quindi il server **invia lo stream di byte**. Il server segue questo protocollo per ogni file e utilizza **la medesima socket** per gestire tutte le comunicazioni.

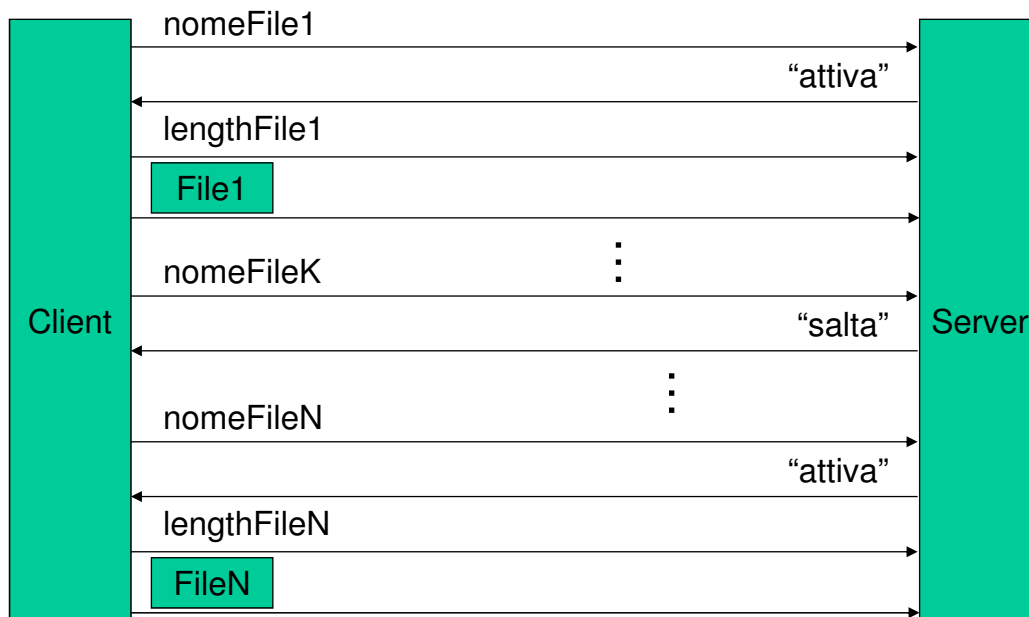
Il ciclo si ripete fino a quando tutti i file nel direttorio richiesto sono stati inviati, quindi il client notifica la fine delle trasmissioni chiudendo la comunicazione e la connessione.



Esercitazione 2 4

Note per la realizzazione

Più in dettaglio, si vuole realizzare il seguente protocollo:



Esercitazione 2 5

Trasferimento di più direttori (versione semplificata)

Per il trasferimento di più direttori bisogna studiare un opportuno **protocollo**. Anzitutto si sottolinea che il direttorio è una struttura locale che **non può** essere trasferita come un qualsiasi altro file.

Se **non ci interessa ricostruire la struttura in direttori (lato server)** è possibile riutilizzare il protocollo proposto in precedenza procedendo, per i vari direttori e i file all'interno di ciascuno di essi come segue: **prima dell'invio del singolo file**, può inviare il **nome del file** e il **numero di byte** da cui il file è composto, quindi il client **invia lo stream di byte**. Il server segue questo protocollo per ogni file e utilizza **la medesima socket** per gestire tutte le comunicazioni e salva tutti i file nel medesimo e unico direttorio corrente.

Il ciclo si ripete fino a l'utente non indica con un EOF, che provoca anche la chiusura della comunicazione col server, l'intenzione di interrompere la sessione

Esercitazione 2 6



Proposta di estensione: Trasferimento più direttori (Client)



Si estenda il programma sviluppato in modo che **gestisca il trasferimento di più direttori dal server al client (multiple get)**.

Protocollo: si estenda il protocollo attuale in modo da abilitare il trasferimento di **diversi direttori** utilizzando la **stessa unica connessione dal cliente al server**.

Per ogni richiesta, **il Client** dovrà creare in locale un direttorio con lo stesso nome di quello richiesto, dove verranno salvati i file inviati dal server, quindi dovrà salvare in tale direttorio i file in arrivo dal server, e mettersi in attesa di una nuova richiesta dell'utente.

Esercitazione 2 7



Proposta di estensione: Trasferimento più direttori (Server)



Il Server, invece, per ogni richiesta, deve inviare tutti i file del direttorio, e notificare la fine della trasmissione del direttorio.

Si gestisca inoltre la **terminazione dell'interazione fra client e server**: il client deve poter indicare al server la propria intenzione di chiusura dell'interazione. Una volta terminata la sessione client e server (*processo figlio del server principale*) terminano la propria esecuzione.

Get e Put Coordinate

Si estenda ulteriormente il programma sviluppato in modo da abilitare il funzionamento del server **in modalità sia get** (trasferimento dal server al client) **che put** (trasferimento dal client al server).

Protocollo: si estendano i protocolli proposti, in modo da gestire la **sessione di lavoro fra client e server** utilizzando sempre una **stessa connessione**.

Per ogni richiesta, **il Client** richiede all'utente il tipo della richiesta che viene inoltrata al server (put o get); poi, **Client e Server** si coordinano per portare a termine l'operazione richiesta. Al termine, il client si pone in attesa di una nuova richiesta dell'utente fino alla terminazione dell'interazione.

Esercitazione 2 9

Consegna

Chi vuole può inviare via email lo svolgimento dell'estensione ai docenti, in modo da discutere la soluzione ed eventualmente pubblicarla sul sito del corso

Esercitazione 2 10