**Università degli Studi di Bologna**
**Scuola di Ingegneria**

# Corso di
# Reti di Calcolatori T

## *Esercitazione 2 (proposta)*
## *Socket Java con connessione*

**Luca Foschini**

Anno accademico 2016/2017



## Specifica Client

Sviluppare un'applicazione C/S che effettui il **trasferimento dal client al server di tutti i file di un direttorio, in particolare quelli il cui nome di file inizia con un carattere vocale e termina con un carattere numerico** (**multiple put con match di nome**).

Il client chiede all'utente il **nome del direttorio**, relativo al direttorio corrente dove viene lanciato il cliente e **un prefisso**, si connette al server con una socket connessa (con **`java.net.Socket`**), usandone lo stream di output per **inviare informazioni**, e lo stream di input per **ricevere il comando di attivazione del trasferimento per ciascun file**.

Il server fornisce un **"attiva"** nel caso un file con quel nome non sia presente sul file system del server con quel nome, esito negativo altrimenti (ad esempio **"salta file"**). 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.

# Specifica Server

Il server attende una richiesta di connessione da parte del client (sulla server socket di ascolto **`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.



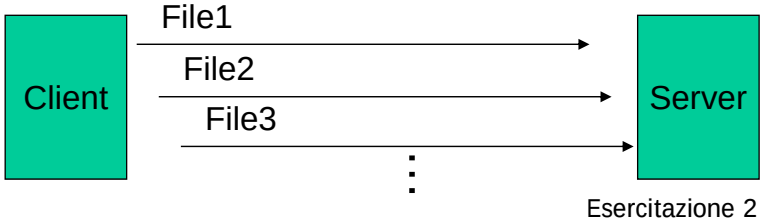
# Note per la realizzazione

Per il trasferimento dei file bisogna studiare un **protocollo per la gestione del trasferimento multiplo**.

Ad esempio il client, **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.

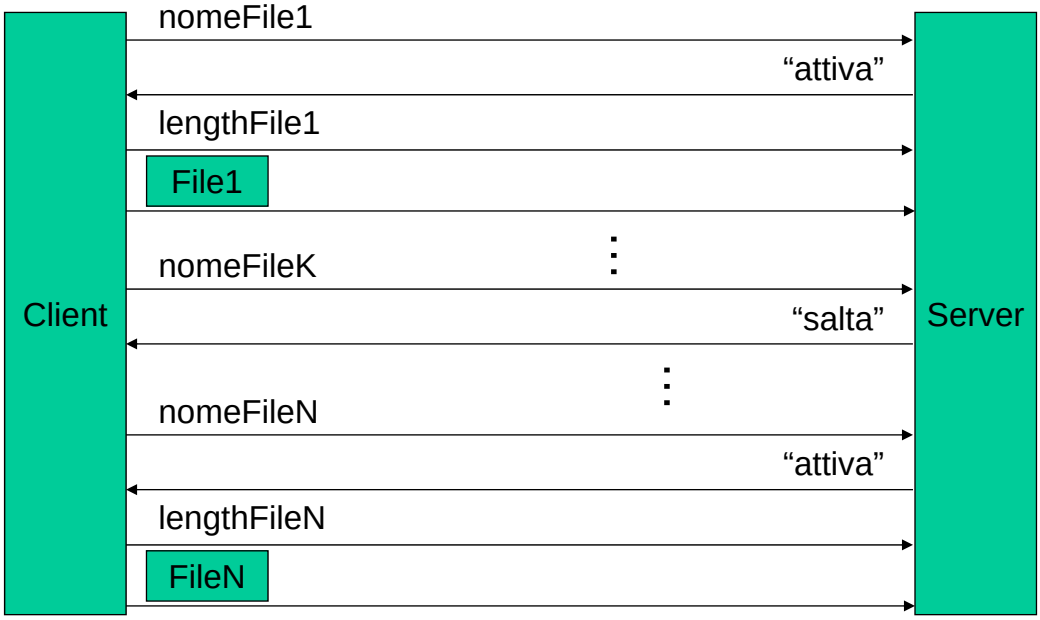
Il ciclo si ripete fino a quando tutti i file nel direttorio richiesto sono stati inviati, quindi il client notifica la fine delle trasmissioni chiudendo la comunicazione e la connessione.

**Trovare un modo per indicarlo.**



# Note per la realizzazione

Più in dettaglio, si vuole realizzare il seguente protocollo:






# 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 dati.

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 che con un carattere vocale e termina con un carattere numerico** , 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 un **fine file**, che provoca anche la chiusura della comunicazione col server, l'intenzione di interrompere la sessione.

# Proposta di estensione: Trasferimento direttori (mget)

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

**Protocollo**: si estenda il protocollo 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.



# Proposta di estensione: Trasferimento più direttori (mget)

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

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.



# Consegna

*Chi vuole può inviare lo svolgimento ai docenti, con uno strumento che verrà specificato sul sito del corso a breve*