Programmazione di sistemi distribuiti

ANNO 2024/25

Laboratorio 3

In questa attività di laboratorio, siete invitati a fare pratica con l'API socket TCP/IP, l'API standard de facto per l'accesso ai servizi di rete dai livelli 4-3-2 di Internet.

L'attività comprende i seguenti compiti:

- implementazione di un'applicazione server socket TCP/IP (in Java);
- implementazione di un'applicazione client socket TCP/IP (in Java).

Lo strumento consigliato per lo sviluppo della soluzione è:

- *IDE Eclipse per sviluppatori Java aziendali* per lo sviluppo di server e client socket TCP/IP.

Contesto dell'attività

Il servizio *Converter* è un server TCP concorrente, in ascolto sulla porta TCP numero 2001, che può eseguire la conversione dei tipi di supporto dei file immagine. I tipi di supporto supportati da questo servizio sono tre: PNG, JPEG e GIF. Il server è in grado di stabilire connessioni TCP con più client; ogni richiesta è gestita da un thread diverso.

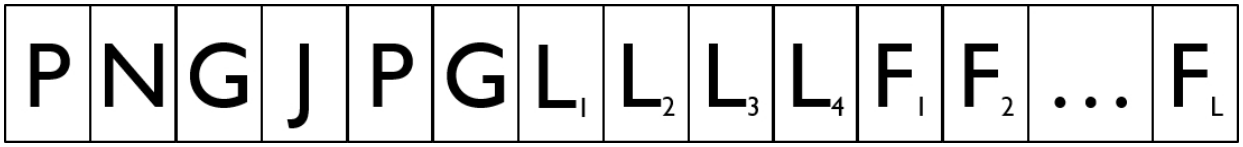
Il client *ConversionRequest* è un'applicazione che deve interagire con il servizio *Converter* per eseguire la conversione di un file immagine. Questa applicazione riceve tre parametri dalla riga di comando:

1. il tipo di supporto originale del file immagine;
2. il tipo di supporto di destinazione in cui il file immagine deve essere convertito;
3. il percorso del file immagine nel file system locale del client *ConversionRequest*.

Dopo aver verificato l'esistenza del file, il client stabilisce una connessione TCP con il servizio *Con- verter.* In questa connessione viene seguito il seguente protocollo. Quando la connessione TCP viene stabilita con successo, il client invia al server:

- tre caratteri ASCII (1 byte per ogni carattere), che rappresentano il tipo di supporto originale del file immagine che deve essere convertito. Le sequenze di caratteri ASCII consentite sono PNG, JPG e GIF;
- tre caratteri ASCII (1 byte per ogni carattere), che rappresentano il tipo di supporto in cui il file immagine deve essere convertito. Le sequenze di caratteri ASCII consentite sono PNG, JPG e GIF;
- un numero intero a complemento a 2 di 4 byte in ordine di byte di rete, che rappresenta la lunghezza in byte del file immagine da convertire;
- i byte del file immagine che devono essere convertiti.

L'immagine seguente illustra un esempio di richiesta di conversione di un file immagine da PNG a JPEG. I byte $_{L1}$, $_{L2}$, $_{L3}$, $_{L4}$ rappresentano la lunghezza del file; i byte $_{F1}$, $_{F2}$, ..., $_{FL}$ rappresentano il contenuto del file da convertire.



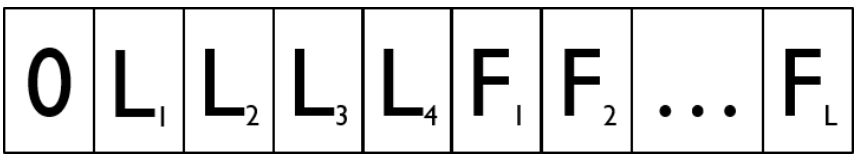
Dopo aver ricevuto tutte queste informazioni dal client, il server converte il file nel tipo di supporto richiesto. Quindi, lo invia al client:

- il carattere ASCII "0" (1 byte) che rappresenta l'esito positivo della ricezione. e le operazioni di conversione;
- un numero intero a complemento a 2 di 4 byte in ordine di byte di rete, che rappresenta la lunghezza in byte del file immagine convertito;
- i byte del file immagine convertito.

Infine, il server avvia la procedura per la chiusura di grazia della connessione.

Dopo aver ricevuto il file e averlo salvato localmente, il client completa la chiusura della connessione TCP.

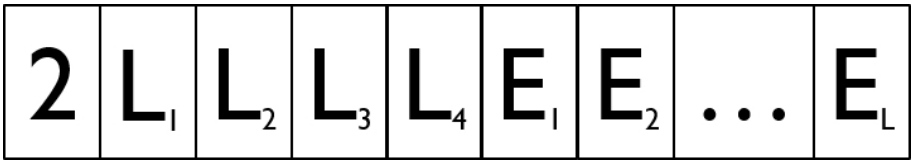
La figura seguente illustra un esempio di risposta del server nel caso in cui tutte le operazioni siano andate a buon fine. I byte $_{L1}$, $_{L2}$, $_{L3}$, $_{L4}$ rappresentano la lunghezza del file convertito; i byte $_{F1}$, $_{F2}$, ..., $_{FL}$ rappresentano il contenuto del file immagine convertito.



Invece, in caso di problemi nella ricezione del messaggio inviato dal client o nella conversione del file immagine, il server invia al client:

- un carattere ASCII (1 byte) che rappresenta un numero maggiore di 0 in caso di esito negativo dell'operazione di ricezione o conversione (in particolare, il carattere ASCII è "1" per una richiesta errata, "2" per un errore interno del server);
- un numero intero di 4 byte in ordine di byte di rete, che rappresenta la lunghezza in byte di una stringa che descrive l'errore verificatosi;
- i byte di una stringa ASCII che descrive l'errore verificatosi.

La figura seguente illustra un esempio di risposta del server nel caso in cui l'operazione di ricezione o di conversione sia fallita a causa di un errore interno del server. I byte L_1, L_2, L_3, L_4 rappresentano la lunghezza del messaggio di errore; i byte E_1, E_2, ..., E_L rappresentano il contenuto del messaggio di errore.



Come testare il client e il server

Provate a convertire file di tipo diverso e verificate che il file convertito ricevuto sia corretto (cioè che possa essere aperto da un visualizzatore senza errori). Provate a convertire un file di grandi dimensioni (ad esempio, 10 MB) e verificate che l'implementazione del protocollo sia ancora corretta ed efficiente nel trasferimento dei file in termini di tempo di trasferimento. Eseguite più client contemporaneamente e verificate come varia il tempo di esecuzione.

Verificate l'interoperabilità testando il vostro client e server rispetto al client e al server di riferimento forniti con il materiale del laboratorio. Per eseguire il client e il server di riferimento, è necessario utilizzare i seguenti comandi nelle rispettive directory locali:

- java -jar client.jar <tipo_originale> <tipo_target> <percorso_immagine>
- java -jar server.jar

Le immagini da convertire devono essere inserite nella cartella "image" della directory in cui si trova il file
client.jar viene eseguito.

Testate e migliorate la robustezza del vostro client e del vostro server

Sia il client che il server dovrebbero utilizzare dei timeout quando attendono input dal peer, per evitare deadlock.

Provare il client e il server in condizioni errate e risolvere eventuali problemi di robustezza:

- Cerca di connettere il client a un indirizzo non raggiungibile.
- Provate a interrompere un client prima che abbia completato la sua operazione (premendo ^C nella sua finestra) e verificate che il server si riprenda correttamente da questo errore.
- Provate ad arrestare il server (premendo ^C nella sua finestra) mentre un client è connesso e verificate che il client gestisca l'errore.

Come vivere questa attività di laboratorio

Siete invitati a completare entrambi i compiti richiesti per svolgere completamente questa attività di laboratorio (cioè lo sviluppo del server TCP e del client). Tuttavia, è possibile riutilizzare il codice Java che esegue la conversione delle immagini dal servizio Java *Converter*, pubblicato come soluzione per la seconda attività di laboratorio.

Per l'implementazione della concorrenza nel lato server, è possibile utilizzare il framework *ExecutorServ ice* fornito dal JDK, poiché semplifica l'esecuzione di task in modalità asincrona (https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Execu- torService.html).

Per evitare problemi di sincronizzazione (ad esempio, condizioni di gara) tra i diversi thread del server, si suggerisce di evitare le operazioni di scrittura sulle variabili condivise.