

Relazione Tecnica TEPsIT

Progetto Client-Server: Simil-FTP

Applicazione Java Socket per il trasferimento file su rete TCP/IP

Gruppo di Lavoro:

Mazzinghi Alessandro

Betti Paolo

Gracci Niccolò

Classe 5^a A Informatica

Anno Scolastico 2025/2026

1 Introduzione

Il progetto consiste nella realizzazione di un'applicazione distribuita Client-Server basata sul linguaggio Java, che simula le funzionalità essenziali di un server FTP (*File Transfer Protocol*). L'obiettivo è permettere a un utente remoto di connettersi a un server centrale, visualizzare il contenuto delle directory (comando **LS**), navigare nel file system (comando **CD** e **CD ..**) e scaricare file (comando **GET**).

L'applicazione è stata progettata per operare in due scenari distinti:

1. **Ambiente Locale:** Test su *localhost* (127.0.0.1) tramite l'ambiente di sviluppo Eclipse.
2. **Ambiente WAN:** Deployment reale con Server su rete domestica (dietro NAT) e Client su rete geografica esterna.

2 Architettura del Sistema

Il sistema utilizza un'architettura centralizzata basata su **Socket TCP** (*Stream Socket*) per garantire l'affidabilità della trasmissione dati, fondamentale quando si trasferiscono file binari.

2.1 Il Server

Il Server adotta un modello **multithreaded**. Il Main Thread rimane in ascolto passivo sulla porta 5500. Ad ogni richiesta di connessione (**accept()**), viene istanziato un nuovo thread (**ThreadConnessioneFTP**). Questo approccio garantisce la gestione concorrente di più client: se un utente sta scaricando un file pesante, il server rimane libero di accettare nuove connessioni.

2.2 Il Client

Il Client è un'applicazione console che interagisce con l'utente, invia i comandi al server tramite **DataOutputStream** e riceve le risposte testuali o i flussi di byte dei file.

3 Analisi del Protocollo (Livello Applicazione)

Per la comunicazione è stato definito un protocollo testuale personalizzato operante sopra TCP.

3.1 Tabella dei Comandi

Di seguito la specifica dei pacchetti scambiati tra Client e Server.

| Comando | Descrizione | Pacchetto Richiesta (Client) |
|---------|---|-------------------------------|
| LS | Richiede la lista dei file nella cartella corrente. | Stringa UTF: "LS" |
| GET | Richiede il download di un file specifico. | Stringa UTF: "GET <nomefile>" |
| CD | Cambia la directory di lavoro sul server. Supporta .. per risalire. | Stringa UTF: "CD <cartella>" |
| QUIT | Termina la sessione e chiude il socket. | Stringa UTF: "QUIT" |

Tabella 1: Definizione dei comandi del protocollo Simil-FTP

3.2 Flusso di Trasferimento File (GET)

Il trasferimento di file binari richiede una sincronizzazione precisa per evitare la corruzione dei dati. Il protocollo segue questi step:

1. **Client:** Invia `GET nomefile.ext`.
2. **Server:** Verifica l'esistenza del file.
 - *Se esiste:* Invia stringa "OK", seguita da un long che rappresenta la dimensione in byte del file.
 - *Se non esiste:* Invia stringa "ERRORE".
3. **Server:** Inizia l'invio dello stream di byte leggendo il file a blocchi (buffer da 4KB).
4. **Client:** Legge esattamente il numero di byte dichiarati nella dimensione, ricostruendo il file in locale.

4 Implementazione Software

Rispetto alla versione iniziale, il codice è stato aggiornato per gestire la navigazione tra cartelle e la validazione dei comandi. L'intero codice sorgente del progetto è disponibile sulla repository GitHub:

https://github.com/alepk0007/Progetto-Client_Server_Simil_FTP

Di seguito il frammento significativo della classe `ThreadConnessioneFTP` che gestisce la logica decisionale:

```

1 // Gestione comandi nel metodo run() del Server
2 if (comando.equals("LS")) {
3     inviaLista();
4 }
5 else if (comando.equals("GET")) {
6     if (parti.length > 1) inviaFile(parti[1]);
7     else out.writeUTF("ERRORE: Specifica un file");
8 }

```

```

9  else if (comando.equals("CD")) {
10     // Nuova implementazione navigazione
11     if (parti.length > 1) cambiaDirectory(parti[1]);
12     else out.writeUTF("ERRORE: Specifica directory");
13 }
14 else if (comando.equals("CD..")) {
15     // Gestione scorciatoia senza spazio
16     cambiaDirectory("..");
17 }
18 else {
19     out.writeUTF("Comando non riconosciuto. Riprova.");
20 }
21
22 // Metodo per il cambio directory
23 private void cambiaDirectory(String nomeDir) throws IOException {
24     File nuovaDir;
25     if (nomeDir.equals("..")) {
26         nuovaDir = directoryCorrente.getParentFile();
27     } else {
28         nuovaDir = new File(directoryCorrente, nomeDir);
29     }
30
31     if (nuovaDir != null && nuovaDir.exists() && nuovaDir.
        isDirectory()) {
32         directoryCorrente = nuovaDir.getCanonicalFile();
33         out.writeUTF("Directory corrente: " + directoryCorrente.
            getName());
34     } else {
35         out.writeUTF("ERRORE: Directory non valida.");
36     }
37 }

```

Listing 1: Gestione comandi e navigazione nel Thread Server

5 Configurazione di Rete

Per rendere il server accessibile dalla rete esterna (es. dalla rete scolastica verso il server domestico), è stata necessaria una configurazione avanzata degli apparati di rete.

5.1 Gestione NAT e Port Forwarding

Poiché il server si trova dietro un router domestico, dispone di un indirizzo IP privato. È stato configurato il Port Forwarding (DNAT):

- **IP Pubblico (WAN):** 79.20.112.29 (VDSL2 Dynamic IP).
- **IP Server (LAN):** 192.168.1.242 (Statico).
- **Porta:** 5500 TCP (Esterna ed Interna).

5.2 Firewall

Sul server Windows è stata creata una *Inbound Rule* nel Windows Defender Firewall per permettere il traffico TCP in entrata sulla porta 5500, altrimenti bloccato di default.

6 Conclusioni

Il progetto ha permesso di approfondire le dinamiche della comunicazione di rete. L'implementazione del protocollo personalizzato sopra TCP ha chiarito l'importanza della gestione degli stream di I/O. Inoltre, l'aggiunta delle funzionalità di navigazione (CD) e la gestione degli errori hanno reso l'applicazione più robusta e simile a un reale server FTP. Il deployment in ambiente reale ha infine evidenziato le problematiche tipiche delle reti WAN, come la configurazione del NAT.