Architettura client-server UDP per trasferimento file

Traccia 2

1. Componenti del gruppo

Lorenzo Colletta
Email: lorenzo.colletta@studio.unibo.it
Matricola: 0000978714

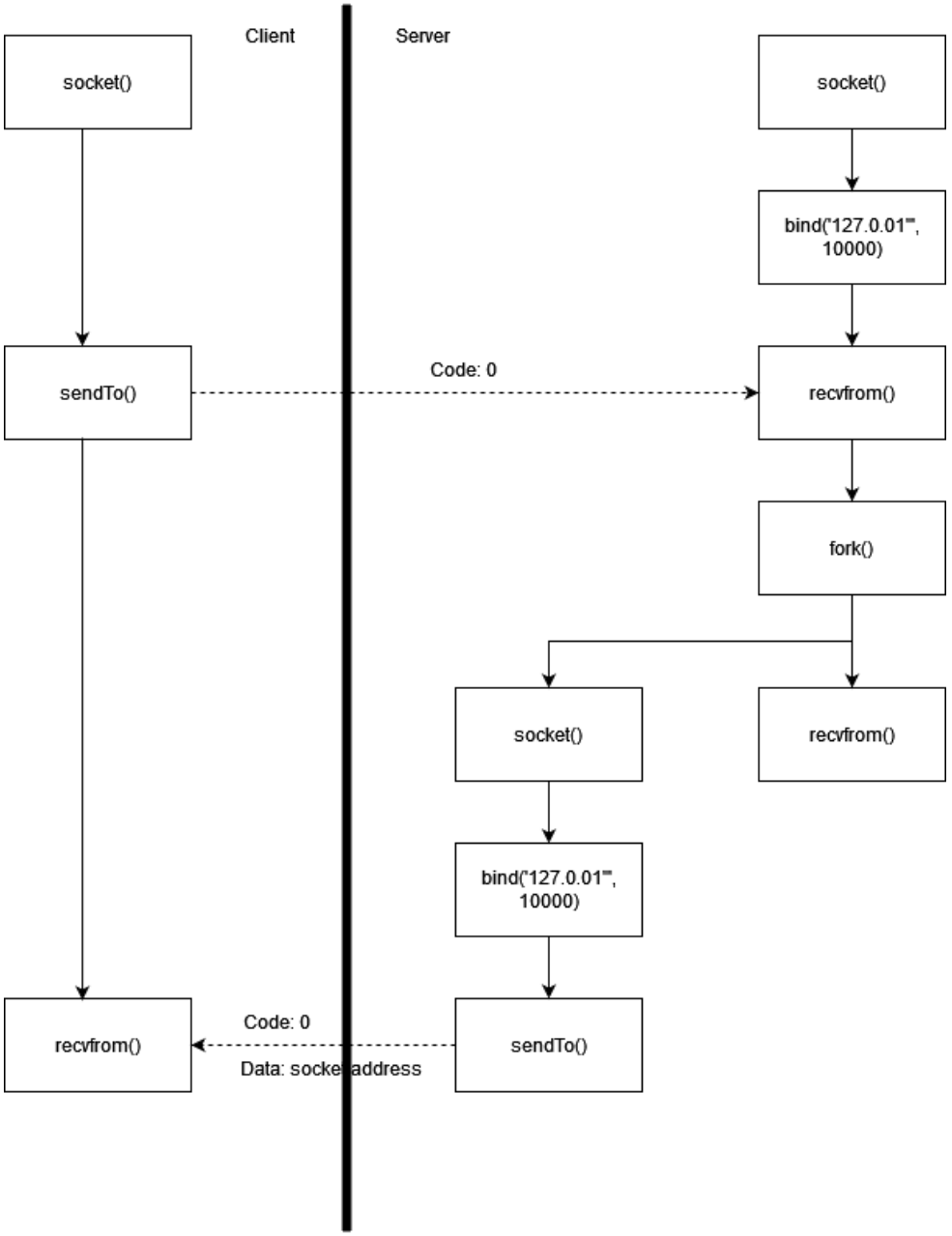
2. Scelte progettuali

connessione

Il protocollo adottato prevede l'instaurazione di una nuova "connessione" a ciascuna richiesta di esecuzione di un comando.
I comandi previsti sono:

- List
- Put
- Get

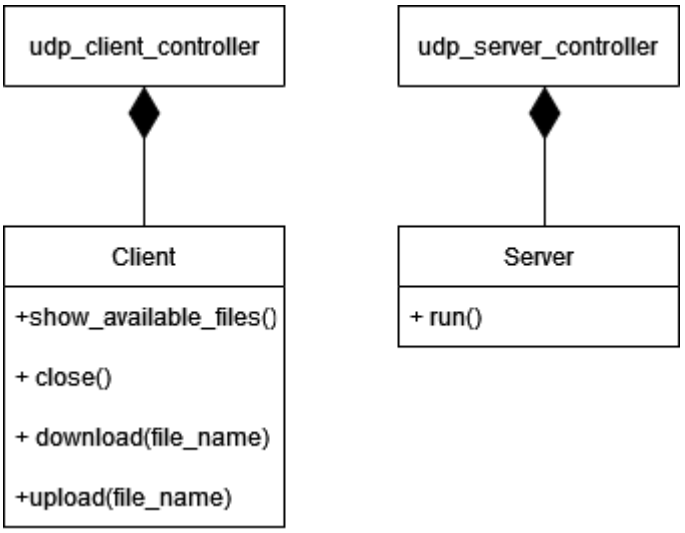
Il comando List prevede la comunicazione da parte del server dei file presenti nella propria cartella corrente. Viene identificato con il codice "1".
Il comando Put prevede il trasferimento di un file da client a server. Viene identificato con il codice 2.
Il comando Get prevede il trasferimento di un file da server a client. Viene identificato con il codice 3.
Dato l'utilizzo del protocollo udp, e della mancata gestione di una connessione da parte di quest'ultimo, la procedura di gestione di una connessione avviene a livello applicativo.
La procedura adottata si limita a verificare la raggiungibilità del server, il client quindi tenta l'invio di un comando per l'instaurazione della connessione, sottoponendo un messaggio con codice "0".
Il server al ricevimento del messaggio si occupa di creare un nuovo socket da utilizzare per il successivo scambio di dati con il client chiamante.
Creato il nuovo socket UDP, il server risponde al client presentando il medesimo messaggio con codice "0" e comunicando il nuovo indirizzo (indirizzo IP + numero di porta del nuovo socket).
A questo punto il server si pone in attesa di un messaggio che includa: un primo carattere che indichi il codice relativo ad un comando da poter eseguire; opzionalmente una stringa che indichi il nome di un file (da ricevere/spedire).



Qui sopra viene riportato uno schema che riassume la procedura di connessione tra client e server.
Avvenuta la connessione, è previsto l'invio di un ulteriore messaggio, dal client al server, che comunichi il comando da eseguire.
Segue una sequenza di messaggi che possono essere rivolti da client a server o viceversa a seconda del comando.

Strutture dati

Nell'applicazione lato server si è reso necessario l'uso di una lista che mantenga traccia dei thread utilizzati per le connessioni con i client.



Qui sopra vengono riportati gli schemi UML delle due applicazioni.

3. Scelte implementative

Buffer

Il protocollo UDP prevede la disposizione di buffer per ciascuna socket istanziata. Tali buffer permettono la memorizzazione temporanea del payload ricevuto prima che venga poi trasmesso dal livello di trasporto al livello applicazione.
Così come lo è il payload di un messaggio UDP, i buffer hanno una capienza piuttosto limitata, risultano talvolta limitati per la trasmissione di file.
Viene dunque prevista una sequenza di trasmissioni qualora l'informazione da inviare superi la capienza dei buffer.

Ricezione

Nelle prime fasi di instaurazione della connessione, si possono verificare due principali problematiche.
La prima, strettamente legata alla natura non affidabile del protocollo UDP, consiste nella possibilità che un eventuale messaggio non venga percepito dal destinatario.
La seconda consiste semplicemente nella possibilità che il server, al momento della richiesta di connessione, non sia in esecuzione nell'host.
Considerati i seguenti scenari, i due host, una volta inviato il proprio messaggio, si pongono in attesa di una risposta che potrebbero non ricevere mai.
Per far fronte a tali problematiche viene impostato un tempo di timeout sul socket, al cui scadere viene terminata l'attesa bloccante avviata dalla chiamata al metodo recvfrom().
Qualora un host rimanga in attesa per 5 secondi, l'host destinatario viene considerato irraggiungibile, si termina così la trasmissione con esito negativo.

Il tempo di timeout viene inoltre adottato per stabilire la fine della trasmissione di un messaggio di risposta al comando richiesto.

Interfaccia

Entrambe le applicazioni sono ciascuna composte di un modulo ed un script.
L'obiettivo di tale progettazione a livello di codice è quello di favorire il riuso.
I due moduli Server.py e Client.py forniscono le classi che implementano il protocollo di trasferimento file a livello applicazione.
I relativi “controller” (udp_client_controller.py e udp_server_controller.py) si occupano dell’interfacciamento con l’utente. Entrambi forniscono un’interfaccia a linea di comando.
L’interfaccia del server si limita a mostrare le richieste ricevute e i loro esiti.
L’interfaccia relativa al client fornisce la possibilità di selezionare il comando da eseguire e la conseguente visualizzazione del risultato.

4. Modalità di esecuzione

Esecuzione del client:

```
python3 udp_client_controller.py
```

Esecuzione del server:

```
python3 udp_server_controller.py
```