SCELTE PROGETTUALI

PROTOCOLLO TCP

Ho scelto il protocollo TCP come protocollo a livello di trasporto in quanto volevo garantire la corretta trasmissione dei dati e l'ordine di ricezione, a discapito della reattività ottenuta utilizzando il protocollo UDP. Quindi, anche se l'utilizzo del protocollo TCP può comportare la generazione di una quantità significativa di traffico a causa del suo controllo di flusso e della gestione degli errori e, per questo motivo, ha una maggiore latenza rispetto a UDP, ho preferito il suo utilizzo per una trasmissione affidabile dei dati.

Avrei potuto fare un'implementazione ibrida tra i due protocolli per ottenere un giusto equilibrio tra affidabilità e reattività, però questo avrebbe richiesto una progettazione più complicata dovuta alla logica di selezione del protocollo da usare in base alle esigenze della singola richiesta.

I/O MULTIPLEXING

Valutando le esigenze specifiche dell'applicazione, la scelta implementativa migliore è risultata quella dell'I/O Multiplexing invece della fork. Anche se quest'ultima è più facile da implementare e da comprendere, poiché ogni figlio mi gestisce una richiesta indipendentemente dalle altre, è computazionalmente pesante in termini di utilizzo delle risorse di sistema e quindi preferibile quando le richieste da gestire si presentano in un numero limitato.

In conclusione, la scelta più adatta per l'applicazione è l'I/O Multiplexing, che mi consente di gestire molte richieste con un unico processo, anche se più complesso in termini di implementazione.

TEXT PROTOCOLS

Presa in considerazione la quantità e il tipo di dati da trasmettere al server e da ricevere dallo stesso, ho scelto di implementare una comunicazione basata su un protocollo di tipo testuale; ho ottenuto così un'implementazione semplificata del client, il quale ha soltanto il compito di inviare al server quanto letto a riga di comando, a discapito della sicurezza aggiuntiva fornita da un protocollo di tipo binario, il quale, inoltre, è preferibile per applicazioni che richiedono la gestione di grandi flussi di dati.

Il controllo della validità dell'input si limita al conteggio delle parole inserite a riga di comando e alla corrispondenza con i comandi richiesti nelle specifiche del progetto.

ASPETTI AGGIUNTIVI

MUTUA ESCLUSIONE

Per com'è stata implementata l'applicazione, la mutua esclusione non è necessaria per il suo corretto funzionamento.

Non ci sono strutture dati che potrebbero essere suscettibili all'accesso in contemporanea da parte di più processi, in quanto tutte le strutture utilizzabili dal server sono accedute da lui stesso e basta. Grazie anche all'utilizzo dell'I/O Multiplexing, c'è soltanto un processo che gestisce le richieste in arrivo; caso contrario quello in cui avessi utilizzato la fork per la gestione delle richieste.