Relazione Tecnica - Sistema d'Asta Online usando TCP

Autori:
Alexei Karavan
Andrea Zicarelli
Corso di Computer Networks, Università di Bolzano, Anno Accademico 2025/2026

1. Requisiti di Ingegneria del Software

1.1. Requisiti Funzionali

Il sistema d'asta online, sviluppato in Java, utilizza il protocollo TCP per garantire una comunicazione affidabile tra il server e i client. Di seguito sono riportati i principali requisiti funzionali del sistema:

- **Server TCP multi-thread**: Il server deve essere in grado di gestire più client contemporaneamente, assegnando gli oggetti da mettere all'asta e gestendo le offerte in tempo reale.
- **Client interattivo**: Ogni client può fare offerte sugli oggetti all'asta, inviare messaggi di chat, ricevere aggiornamenti sull'asta, visualizzare l'offerta più alta e il prezzo attuale, e uscire dall'asta in qualsiasi momento.
- **Gestione delle sessioni**: Il server deve permettere l'ingresso e l'uscita dei client in modo sicuro, gestendo gli errori e le disconnessioni in modo che non blocchino il funzionamento del sistema.
- **Asta a tempo**: Ogni oggetto all'asta ha un periodo di tempo definito (ad esempio, 2 minuti), al termine del quale l'oggetto viene assegnato al miglior offerente.

1.2. Requisiti Non Funzionali

- **Scalabilità**: Il sistema deve supportare più client connessi simultaneamente senza compromettere le prestazioni.
- **Robustezza**: Il server deve continuare a funzionare correttamente anche se uno o più client si disconnettono improvvisamente.
- **Usabilità**: Il client deve essere facile da usare, con comandi chiari come `/join`, `/bid`, `/msg`, `/quit`, e `/info`.

1.3. Diagrammi

Diagramma delle Classi

Nel diagramma delle classi, vediamo i seguenti componenti:

- **Server**: Gestisce la connessione con i client e la gestione delle aste.
- **ClientHandler**: Un thread per ogni client connesso, responsabile per ricevere e inviare messaggi.
- **AuctionItem**: Rappresenta un oggetto messo all'asta.
- **AuctionState**: Contiene lo stato attuale dell'asta (prezzo, miglior offerente).

2. Progettazione dell'Algoritmo

2.1. Interazioni Client-Server

Il flusso di interazioni tra i client e il server avviene nel seguente modo:

- **Client connesso al server**: Il client si connette al server e invia il comando `JOIN <nickname>`. Il server verifica se il nickname è già preso e, se valido, accetta la connessione.
- **Asta**: Il server invia un oggetto da mettere all'asta, con il prezzo iniziale e l'incremento minimo per l'offerta. I client possono iniziare a fare offerte usando il comando `/bid <importo>`. Ogni offerta valida aggiorna il prezzo e il miglior offerente.
- **Comunicazione tra client**: I client possono inviare messaggi usando il comando `/msg <messaggio>`, che viene inviato a tutti i client connessi.
- **Timeout e chiusura dell'asta**: Ogni oggetto all'asta ha una durata predefinita (ad esempio, 2 minuti). Quando il timer scade, l'asta si chiude e l'oggetto viene assegnato al miglior offerente.

2.2. Gestione delle Offerte

Il server gestisce le offerte in tempo reale. Quando un client invia un'offerta, il server verifica che l'offerta sia maggiore o uguale al prezzo corrente più l'incremento minimo. Se l'offerta è valida, il server aggiorna lo stato dell'asta e invia la conferma a tutti i client.

3. Implementazione in Java

3.1. Struttura del Codice

Il sistema è implementato utilizzando il linguaggio Java, con comunicazione tra client e server tramite socket TCP.

- **Server.java**:
  - Gestisce la connessione con i client.
  - Gestisce le aste.
  - Gestisce i timeout per ogni oggetto all'asta.
  - Facilita la comunicazione tra i client tramite la chat.
- **Client.java**:
  - Gestisce la connessione al server.

- Invia comandi come `/join`, `/bid`, `/msg`.
- Visualizza le informazioni sull'asta e i messaggi dal server.

4. Test e Comportamenti Inaspettati

4.1. Test Funzionali

- **Connessione al server**: Ogni client si connette al server con un nickname univoco e invia il comando `/join`.
- **Invio offerte**: I client possono inviare offerte con il comando `/bid`, che viene validato dal server.
- **Chat**: I client possono inviare messaggi alla chat usando il comando `/msg`.

4.2. Test di Concorrenza

Poiché il server supporta più client connessi contemporaneamente, sono stati testati scenari con più client e con disconnessioni improvvise, verificando che gli altri client possano continuare a partecipare senza problemi.

4.3. Bug Riscontrati

- **Disconnessione improvvisa di un client**: Se un client si disconnette durante un'asta, il server lo rimuove correttamente, ma sarebbe utile migliorare la notifica della disconnessione agli altri client.
- **Offerte simultanee**: Quando più client fanno offerte nello stesso momento, il server gestisce correttamente la sincronizzazione, ma in alcuni casi si è verificato un piccolo ritardo nella risposta ai client.

5. Possibile Implementazione Alternativa con UDP

5.1. Vantaggi di UDP

- **Maggiore velocità**: UDP, essendo un protocollo senza connessione, elimina l'overhead legato alla gestione delle connessioni, rendendolo più veloce.
- **Minor utilizzo di risorse**: UDP richiede meno risorse, in quanto non necessita di gestire sessioni persistenti.

5.2. Svantaggi di UDP

- **Affidabilità**: UDP non garantisce la consegna dei pacchetti, il che potrebbe comportare la perdita di messaggi o offerte importanti.
- **Ordinamento dei pacchetti**: Poiché UDP non garantisce l'ordine dei pacchetti, dovremmo implementare una logica aggiuntiva per gestire l'ordinamento delle offerte e la sincronizzazione tra i client.

- **Rilevamento degli errori**: UDP non offre meccanismi per la gestione degli errori, quindi dovremmo implementare la ritrasmissione dei pacchetti persi e la gestione degli errori a livello di applicazione.

5.3. Confronto Critico

Caratteristica	TCP	UDP
Affidabilità	Garantita, con ritrasmissioni e controllo	Nessuna garanzia, pacchetti possono essere persi
Velocità	Più lento a causa della gestione delle connessioni	Più veloce, senza overhead di connessione
Complessità	Più semplice da implementare, gestione automatica della connessione	Richiede più lavoro per gestire errori e ordine dei pacchetti
Scalabilità	Buona, ma può soffrire con troppi client simultanei	Molto buona, soprattutto in ambienti con pochi client
Robustezza	Alta, gestisce errori e disconnessioni	Bassa, con possibilità di perdita di pacchetti

In conclusione, sebbene UDP sia più veloce, TCP è la scelta migliore per questo sistema d'asta grazie alla sua affidabilità nella gestione delle offerte e della sincronizzazione tra client e server.

Conclusioni

Il sistema d'asta sviluppato è funzionale e robusto, con una gestione efficiente dei client e una corretta implementazione della logica dell'asta. Funzionalità avanzate come la barra di stato live e il comando `/info` aumentano l'interattività e l'usabilità del client. Il confronto con una possibile implementazione UDP evidenzia i compromessi tra velocità e affidabilità, confermando la validità della scelta di TCP.

Author: Alexei Karavan · © · Free University of Bozen-Bolzano