Report: A Distributed Auction System

Vicki Ventzel, Sofia de Simón, Tomás Caporale

November 2024

1 Appendix: Corrections

We had implemented the following corrections on the system.

Token passing mechanism: Now the Token passing mechanism supports multiple peer nodes by using a slice peerNodes. It maintains currentIndex to track the next node in the ring, and pendingBids to queue bids when the token isn't held.

Auction handling: Processes bids through processBid - to ensure higher bids are accepted. Differentiates between holding the token and not.

Crash resilience: Detects and handles node failures during token passing. Implements methods getNextNodeAddr and GetNextNode to dynamically adjust the ring.

We have replaced nextNode with a slice peerNodes to handle multiple nodes. We use currentIndex to track the next node in the ring, and pendingBids to queue bids when not holding the token.

2 Github Repository

The code repository for this project is available at GitHub Repository.

3 Introduction

In this report, we describe the implementation of a distributed Auction System that uses the token ring algorithm to manage access to a Critical Section (CS). The system consists of multiple peer nodes that communicate by using gRPC and protocol buffers, which ensures safe and fair access to the Critical Section (requirements from the previous mandatory activity). The Distributed Auction System is based on the code from the previous mandatory assignment. Some of the additional features are dynamic ring handling, bid processing with retry mechanisms, structured logging, and secure communications protocols.

4 Architecture

4.1 Node

The system consists of multiple nodes, each with the ability to request access to a Critical Section (CS) used to manage the auction operations. Each node communicates with its peers in the network to request and pass a token, which ensures that only one node can access the CS at any given time. When a node receives the token, it enters the CS to perform auction-specific tasks, such as processing incoming bids, updating the highest bid, or determining the auction winner. After completing its operations within the CS, the node passes the token to the next node in the ring, thereby ensuring that the CS is accessible to all participants in a circular manner.

4.2 Client and gRPC Services

Clients interact with the system through two gRPC services:

- **TokenRing Service**: Handles token passing between nodes.
- **AuctionService**: Manages bid submissions and auction queries.

Communication uses Protocol Buffers for structured messages and gRPC for efficient and reliable messaging.

4.3 Auction State and Bid Processing

Each node maintains a shared *AuctionState*, tracking the highest bid, bidder, and auction status. Bid processing occurs through two methods:

- **Bid Channel**: A buffered channel enqueues bids for sequential processing.
- **Bid Forwarding**: Forwards bids to the next node if the current node lacks the token.

4.4 Token Ring Algorithm

The Token Ring Algorithm ensures mutual exclusion by circulating the token among nodes. Only the token-holding node can access the CS. The deterministic ring structure guarantees eventual access for all nodes, maintaining fairness.

5 Correctness 1

Sequential Consistency is a memory model where all operations appear in a consistent order across nodes, with each node's operations maintaining their program order.

Our system satisfies sequential consistency because:

- Nodes process operations (e.g., token passing, bids) in the same order.
- The token mechanism ensures exclusive access, preventing conflicts.

The system does not satisfy the stricter **linearisability** due to delays from token passing, but sequential consistency is sufficient for this use case.

6 Correctness 2

6.1 Without Failures

The protocol ensures:

- **Mutual Exclusion**: Only the token holder accesses the CS.
- **Fairness**: The ring guarantees eventual access for all nodes.
- **Determinism**: Token passing is predictable, preserving consistent auction states.

6.2 With Failures

The system handles failures by:

- **Bid Retry**: Retries forwarding bids to avoid loss.
- **Token Recovery**: Regenerates lost tokens to resume operations.
- **Timeouts**: Detects and handles token delays or node failures.

These mechanisms ensure robustness and maintain correctness under transient or permanent failures.