Real-Time Bid War System with Live Video Streaming and Dynamic Host Switching

1 Introduction

In traditional online auctions, users place bids through a centralized server, often facing issues such as:

- High latency
- Lack of transparency
- Security risks
- Limited interactivity

Existing systems rely heavily on web interfaces and pre-recorded content, making them less engaging and prone to fraudulent activities. Additionally, the static nature of auctioneers (hosts) limits flexibility, preventing users from dynamically switching roles between bidders and hosts in real-time.

2 Problem Statement

The goal of this project is to develop a real-time bid war system that enables users to participate in live auctions while integrating the following key features:

2.1 Real-Time Bidding System

- Clients can connect to the auction server and place bids in real-time.
- The server processes bids, updates the auction status, and broadcasts changes instantly to all connected clients using WebSockets.
- Bid validation mechanisms prevent bid fraud and ensure fair competition.

2.2 Live Video Streaming for Auctions

- The auction host streams a live video feed using WebRTC/FFmpeg, allowing bidders to see the item in real-time.
- Clients receive and decode the video stream with minimal latency.
- Adaptive bitrate streaming ensures optimal video quality based on network conditions.

2.3 Client-to-Host Switching (Dynamic Role Assignment)

- A bidder can request to become the host dynamically.
- The server validates the request and seamlessly transitions the new host.
- Existing clients reconnect to the new host's video stream without interruption.

2.4 Secure and Optimized Communication

- End-to-End Encryption (TLS, AES) for secure bid transmissions and video streams.
- DDoS Prevention to protect against auction flooding attacks.
- Low-Latency Network Optimization using compression and packet prioritization techniques.

2.5 Scalable and Efficient Auction Management

- Multiple users can join an auction simultaneously.
- A centralized `server.py` efficiently handles auction states and client connections.
- In-memory storage (Redis) speeds up bid processing, reducing network overhead.

3 Technical Scope & Constraints

- The project will be entirely developed in Python, focusing on network functionality without a frontend.
- Core components will be limited to `server.py`, `client.py`, and `video_stream.py` to ensure modularity.
- WebSockets (`websockets` library) will handle bid transmissions, and WebRTC (`aiortc`) will enable live video streaming.
- The system should support high-concurrency, meaning multiple users can place bids and watch live streams without lag.

4 Expected Outcome

- A fully functional, real-time auction system that efficiently processes bids and live video.
- A robust, network-focused implementation with client-server interactions and peer-to-peer streaming.
- A scalable and secure architecture ensuring smooth user experience under varying network conditions.