WebSocket과 SSE를 활용한 실시간 경매 시스템

조규철[O], 오주헌[*]
[O]인하공업전문대학 컴퓨터정보공학과
[*]인하공업전문대학 컴퓨터정보공학과
e-mail: kccho@inhatc.ac.kr[O], 202447003@itc.ac.kr[*]

Implementation of Real-time Auction System Based on WebSocket and Server-Sent Events

Cho Kyu Cheol[O], Oh Ju heon[*]
[O]Dept. of Computer Science, Inha Technical College,
[*]Dept. of Computer Science, Inha Technical College

요 약

본 논문에서는 WebSocket과 Server-Sent Events(SSE)를 활용한 실시간 경매 시스템을 설계하고 구현하였다. 기존의 HTTP 기반 경매 시스템들이 가지고 있던 이벤트기반 구현의 한계와 서버 부하 문제를 해결하기 위해, WebSocket을 통한 양방향 실시간 입찰 처리와 SSE를 통한 단방향 알림 시스템을 구축하였다. 입찰 데이터의 효율적인 처리를 위해 Redis를 인메모리 데이터베이스로 활용하였으며, 영속성이 필요한 데이터는 MariaDB에 저장하는 하이브리드 구조를 채택하였다. 백엔드는 Spring Boot를 사용하여 REST API를 구현하였고, 프론트엔드는 React를 활용하여 사용자 인터페이스를 구현하였다. 실험 결과, 제안된 시스템은 다수의 동시 사용자 환경에서도 안정적인 실시간 입찰 처리가 가능함을 확인하였다.

▸ Keyword : 실시간 양방향 통신(Real-time Bidirectional Communication), 웹 소켓(WebSocket), SSE(Server-Sent Events), 인메모리 데이터베이스(In-memory Database)

I. Introduction

최근 전자상거래 시장의 급격한 성장과 함께 실시간 경매 시스템에 대한 수요가 증가하고 있다. 특히 미술품, 한정판 상품 등 다양한 분야에서 실시간 경매의 활용이 확대되면서, 신뢰성 있고 효율적인 경매 시스템의 필요성이 대두되고 있다[1]. 그러나 기존의 HTTP 기반 경매 시스템들은 실시간 데이터 처리와 사용자 경험 측면에서 여러 한계점을 보인다. 특히 주기적인 HTTP 요청을 통한 폴링 방식은 서버 부하 증가와 실시간 구현에 한계가 있다[2].

본 연구에서는 이러한 문제점을 해결하기 위해 WebSocket과 Server-Sent Events(이하, SSE)를 활용한 실시간 경매 시스템을 제안한다. WebSocket의 양방향 통신을 통해 즉각적인 입찰 처리와 경매 상태 변경을 구현하고, SSE를 활용하여 사용자에게 경매 알림을 효율적으로 전달한다. 또한 Redis를 활용한 인메모리 데이터 처리를 통해 대규모 동시 접속 상황에서도 안정적인 성능을 보장한다.

II. Preliminaries

2.1 WebSocket과 Server-Sent Events (SSE)

WebSocket은 클라이언트와 서버 간의 양방향 통신을 지원하는 프로토콜이다[3]. 기존의 HTTP 통신과 달리 연결이 한 번 수립되면 지속적으로 유지되어 실시간 데이터 교환이 가능하다. 특히 경매 시스템에서 입찰가 갱신과 같은 실시간 정보 교환에 필수적인 기술이다.

SSE는 서버에서 클라이언트로의 단방향 실시간 이벤트 스트림을 제공하는 기술이다. WebSocket보다 가벼우며, 서버에서 클라이언트로의 알림이나 업데이트를 전송하는 데 최적화되어 있다. 경매 종료 알림, 새로운 입찰 알림 등의 기능 구현에 활용된다.

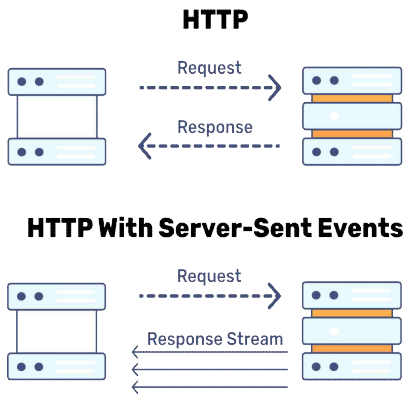

Fig. 1. HTTP vs HTTP With Server-Sent Events[5]

2.2 기존 경매 시스템의 한계와 개선 방안

야후 옥션은 사용자가 페이지를 수동으로 새로고침해야만 현재 입찰가가 갱신되는 방식을 사용했으며, 이는 경매 마감 직전의 중요한 입찰 정보를 놓치는 문제가 있었다. 본 연구에서는 이러한 문제점을 해결하기 위해 WebSocket 기반의 실시간 양방향 통신을 도입하여 페이지 새로고침 없이도 즉각적인 입찰 정보 갱신이 가능하도록 하였다. 이를 통해 사용자들은 보다 능동적이고 실시간적인 경매 참여가 가능하며, 특히 경매 마감 직전의 치열한 입찰 경쟁 상황에서도 모든 참여자에게 신속한 입찰 기회를 제공할 수 있다.

III. Database Scheme

<그림 2>은 본 시스템의 구조를 나타낸 객체-관계 모델(ERD)이다. 테이블은 거래, 경매, 사용자, 알림, 입찰, 이미지, 카테고리, 서버생명주기로 구성되어 있다.

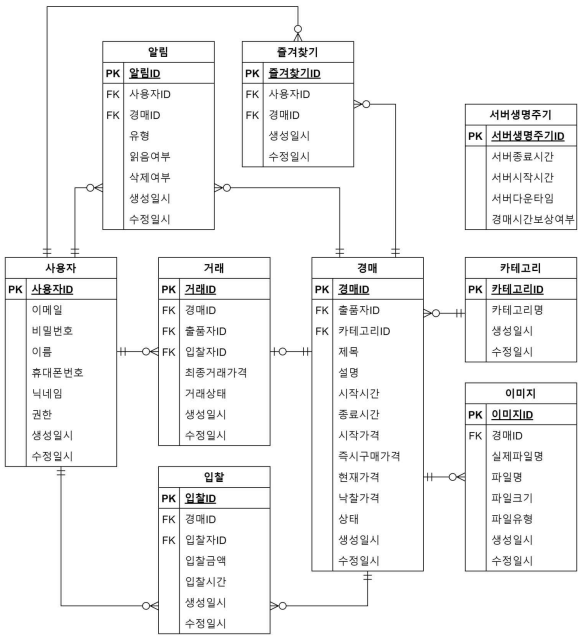

Fig. 2. Entity Relationship Diagram

3.1 관계형 데이터베이스 스키마

경매 테이블은 경매의 기본 정보를 저장하며, 사용자와 카테고리 테이블을 참조(FK)하여 판매자와 카테고리 정보를 관리한다. 입찰 테이블은 경매ID와 사용자ID를 참조(FK)하여 각 경매의 입찰 기록을 저장한다. 거래 테이블은 낙찰된 경매의 거래 정보를 관리하며, 경매ID와 출품자ID(사용자ID) 입찰자ID(사용자ID)를 참조(FK)하여 거래 당사자 정보를 저장한다. 이미지 테이블은 경매ID를 참조(FK)하여 경매 상품의 이미지 정보를 관리한다. 알림 테이블은 사용자ID를 참조(FK)하여 사용자별 알림 정보를 저장하며, 즐겨찾기 테이블은 사용자ID와 경매ID를 참조(FK)하여 사용자의 관심 경매 정보를 관리한다. 서버생명주기 테이블은 서버의 운영 기록을 저정하여 시스템 장애 시 경매 시간 보상을 위한 정보를 제공한다.

3.2 Redis 데이터 구조

실시간 처리가 필요한 데이터는 Redis에 저장되어 관리된다. 입찰 정보는 실시간 처리를 위해 Redis에 우선 저장되며, 주기적으로 관계형 데이터베이스와 동기화된다. 또한 사용자 인증을 위한 Refresh Token도 Redis에 저장되어 효율적인 세션 관리가 가능하다. 이러한 하이브리드 데이터베이스 구조는 정규화된 엔티티 설계를 통해 데이터 중복을 최소화하고, 외래 키 제약 조건을 통해 데이터 무결성을 유지한다. 특히 실시간 처리가 필요한 입찰 데이터는 Redis를 활용하여 처리하며, Redis의 인덱싱 기능을 통해 데이터를 빠르게 검색 가능하도록 구현하였다.

IV. Main Function

본 시스템의 주요 기능은 크게 실시간 입찰 처리, 입찰 모니터링, 알림 시스템으로 구분된다. <그림 3>는 입찰 처리 흐름을 보여준다.

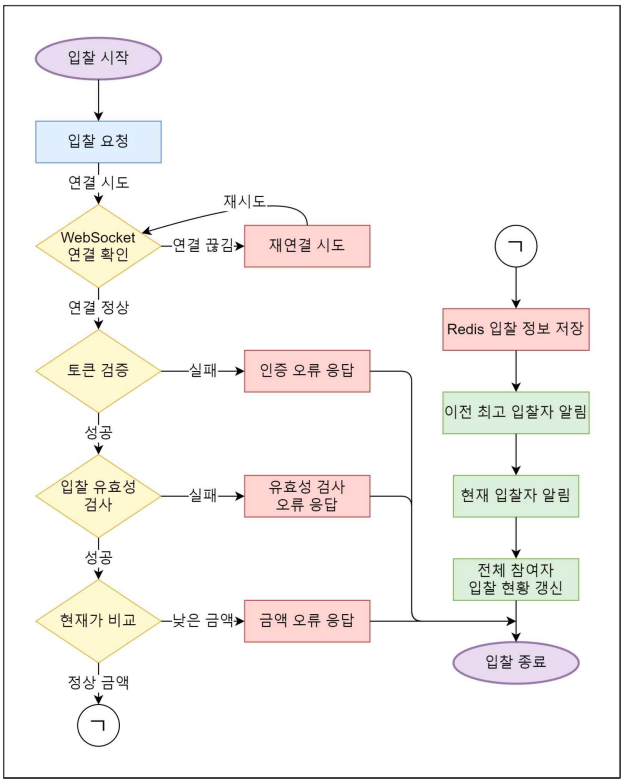

Fig. 3. Bid Flow Chart

실시간 입찰 처리 기능은 WebSocket 프로토콜을 기반으로 양방향 실시간 통신을 구현하였다. 입찰 처리 과정에서는 최소 입찰가 제한, 입찰 단위 검증, 본인 경매 참여 제한, 현재 최고 입찰가 대비 상회 금액 검증 등 다양한 유효성 검증을 수행한다.

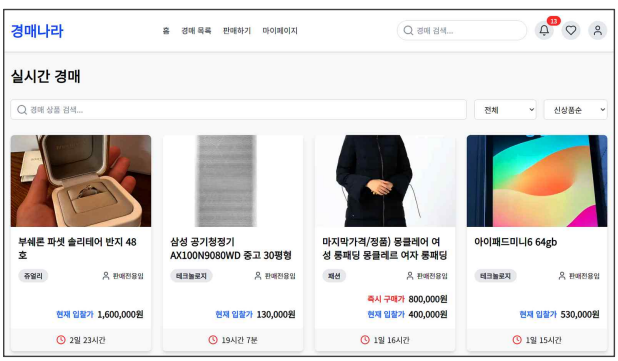

Fig. 4. Auction List

동시성 제어를 위해 ConcurrentHashMap을 활용한 경매당 세션 관리와 트랜잭션 처리를 위해 데이터 정합성을 보장하며, WebSocket을 통해 입찰 성공/실패 여부를 즉시 통보하고 모든 참여자에게 상태 변경을 실시간으로 전달한다.

```
private final Map<Long, Set<WebSocketSession>> auctionRooms = new ConcurrentHashMap<>  ;
private final ObjectMapper objectMapper = new ObjectMapper  ;

@Override
public void afterConnectionEstablished @NonNull WebSocketSession session  throws Exception
    Long auctionId = getAuctionId session ;
    Set<WebSocketSession> auctionRoom = auctionRooms.computeIfAbsent auctionId,
            key -> new CopyOnWriteArraySet<>  ;


    auctionRoom.add session ;
```

Fig. 5. WebSocket Handler Code

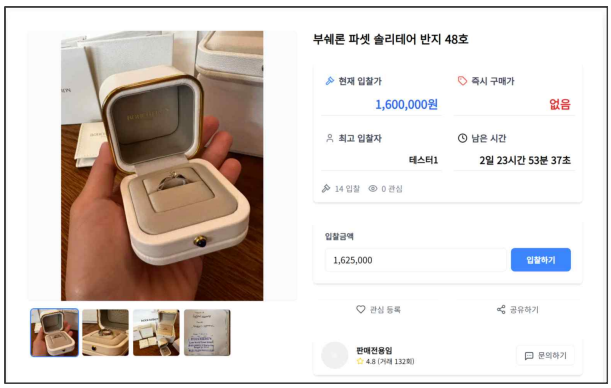

Fig. 6. Auction Detail Page

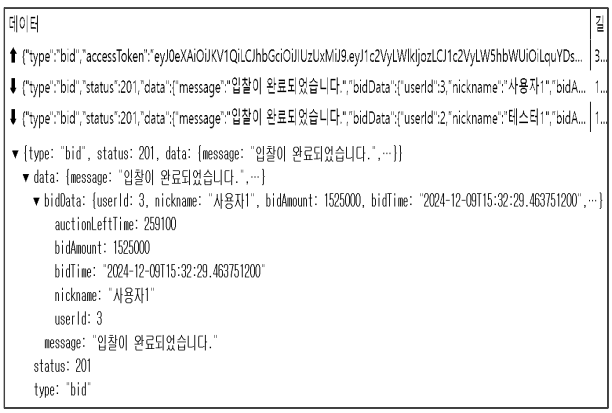

Fig. 7. WebSocket Message

입찰 모니터링 기능은 사용자 경험 향상을 위해 Chart.js 라이브러리를 활용하여 실시간 입찰 데이터 시각화를 구현하였다. 입찰가 변동 추이를 실시간으로 그래프화하고 시계열 데이터를 실시간으로 렌더링하며, 사용자별 입찰 포인트를 차별화하여 표시한다. 또한 최고 입찰자 정보와 경매 잔여 시간을 실시간으로 업데이트하고, 입찰 이력을 실시간으로 표시하여 사용자들이 경매 진행 상황을 직관적으로 파악할 수 있도록 구현하였다.

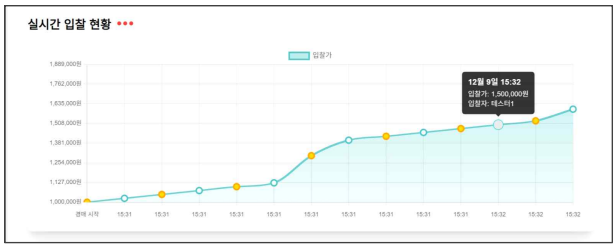

Fig. 8. Bidding Chart using Chart.js

실시간 알림 시스템은 Server-Sent Events(SSE) 기술과 JPA를 활용하여 실시간 알림 처리와 알림 이력 관리를 구현하였다. 알림 시스템은 입찰(BID), 낙찰(WIN), 상회입찰(OUTBID), 마감임박(REMINDER) 네 가지 상태의 알림을 제공하며, 각 알림은 경매 정보, 입찰 정보, 시간 정보 등을 포함한다.

Fig. 8. Auction Notification using SSE

IV. Conclusions

본 연구에서는 WebSocket과 SSE를 활용한 실시간 경매 시스템을 설계하고 구현하였다. WebSocket을 활용한 양방향 실시간 통신으로 입찰 처리의 즉시성을 확보하였으며, ConcurrentHashMap을 통한 세션 관리와 트랜잭션 처리로 동시성 제어를 구현하여 다수의 사용자가 동시에 입찰하는 상황에서도 안정적인 운영이 가능함을 확인하였다. Redis를 인메모리 데이터베이스로 활용하여 실시간 입찰 데이터를 효율적으로 처리하였고, MariaDB와의 하이브리드 구조를 통해 데이터의 영속성을 보장하면서도 빠른 응답 속도를 달성하였다. SSE를 활용한 단방향 알림 시스템으로 서버 리소스를 효율적으로 사용하면서 입찰, 낙찰, 상회입찰, 마감임박 등 다양한 상황에 대한 알림을 실시간으로 전달할 수 있게 되었으며, Chart.js를 활용한 실시간 데이터 시각화로 입찰 추이를 실시간 그래프로 표현하여 사용자 경험을 향상시켰다.

향후 연구과제로는 블록체인 기술을 도입하여 거래의 투명성과 신뢰성을 강화하고, 머신러닝을 활용한 경매 가격 예측 모델 개발이 필요할 것으로 보인다. 또한 시스템의 확장성을 고려한 마이크로서비스 아키텍처로의 전환도 검토할 필요가 있다.

References

[1] K-ARTMARKET, (2020), Online Auction Analysis https://k-artmarket.kr/member/board/Report_BoardView.do?boardId=BRD_ID0001316

[2] Wallarm, (2024), WebSocket vs HTTP: How Are These Two Different. https://www.wallarm.com/what/websocket-vs-http-how-are-these-2-different

[3] mdn web docs, (), WebSockets API https://developer.mozilla.org/ko/docs/Web/API/WebSockets_API

[4] JAVASCRIPT.INFO, (2022), Network requests: WebSocket https://ko.javascript.info/websocket

[5] Bunny.net Academy, (n.d.), What are (Server-Sent Events) and how do they work? https://bunny.net/academy/http/what-is-sse-server-sent-events-and-how-do-they-work