Design Document

A Publish-Subscribe Distributed Auctioning System

Anindita Das
dasanuiit@gmail.com

About This Document

This document describes the design of a publish-subscribe middleware system implementing distributed auctioning. We start with a description of the high-level design, and then discuss how the different aspects of the design are mapped to our implementation.

Design Architecture

The distributed auctioning system consists of three agents: (1) Sellers, (2) Buyers, and (3) Brokers. The buyers and sellers publish and subscribe to different events pertaining to the auctioning functionality, and these publications are propagated across the system by the brokers. Implementing the system involves implementing the functionalities of these three agents and how they communicate with each other. However, for the purpose of designing the system, we find easier to consider the *flow* of events pertaining to any item i, that correspond to the different system functionality as they are passed along by different agents. In particular, we decompose the flow of events into the following three groups.

1. **Readiness:** For any item i, the *readiness events*, we refer to the events that must be published by some agent (seller or buyer) before the core events (*viz.*, bidding and sale in this case) for item i can occur. In particular, the seller must make item i available through an `available item` publication and the buyer must publish an `interest` that matches i. The `available item` publication is generated by the user interface of the seller, as is the `interest` publication of the buyer.

2. **Bidding:** Bidding for any specific item i for sale is initiated with some buyer b_0 publishing a `bid` event for the item. A pre-requisite for the bidding flow is

that b_0 actually knows the `item id` for the item to be bidden on. In our design, this is achieved typically by the interest specification activity. On its publication, the responsibility of the broker is to propagate the `bid` event to the appropriate seller s; the broker also subscribes b_0 as for `bid update` events for the item. On receiving a bid, the seller publishes the corresponding `bid update` event which is propagated to all subscribers to bids on item i. The seller can also close the sale of item i by publishing a `sale finalized` event. Each subscriber receiving the `bid update` can continue the bid, and the sequence of events repeats until the seller eventually publishes a `sale finalized`.

3. **Exits:** A seller or buyer can leave the market at any time. When a seller leaves, all his publications are deleted; when a buyer leaves, all his subscriptions are deleted. These can be resolved as follows. When a client (seller or buyer) leaves, he publishes an `Exit` event. A broker receiving an `Exit` publication from a seller s deletes all his publications. Any buyer subscribing to a publication from s is notified that the items are no longer available. When a broker receives an `Exit` publication from a buyer b, b is dropped from all subscriptions. If b is in fact the highest bidder of some item, then that item is returned to the market as “new”.

The descriptions above suggest a certain ordering on the activites (and publications) of the different system components. Fig. 1 gives an overview of how the events are executed. In particular, we design the clients (sellers and buyers) as systems that go through the following steps for each item i, assuming the clients do not leave during the duration when i enters the market and when i is sold.

Seller:

- Receives data on item i through the user interface.
- Publishes an `available item` for i and subscribes to `bid updates` for i.
- Periodically checks the current bid for i and determines whether to send a `sale finalized` event for i.
- Sends the `sale finalized` event for i.

Buyer:

- Publishes an interest through user interface.
- Receives publication on i because one of his interests matches i.
- Determines (through user interface) whether to bid for i or not, and if so, whether the bidding is to be manual and automatic.

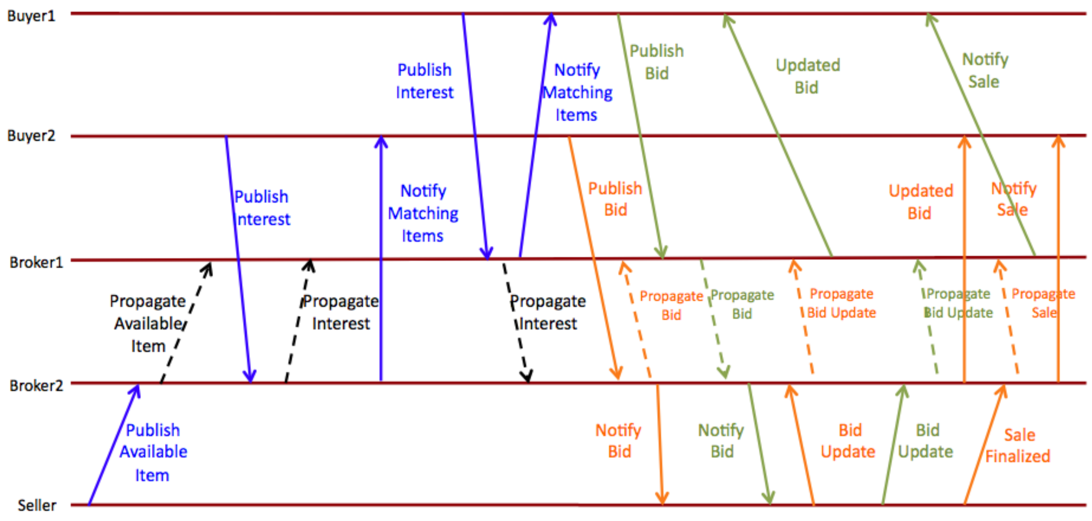

Figure 1: Process Diagram showing the flow of events for the Publish-subscribe Auctioning system. For simplicity, we show two buyers and one seller. Both Buyer2 and Seller are assumed to be connected to Broker2.

- Goes through a loop bidding and receiving updated bid on i, until the item is sold or he is no longer interested.

The above description also makes the role of the broker clear. A broker connected to buyer b must (1) receive the interests from b, (2) find an item i matching the interests for b, and (3) receive bids from b on i, (4) propagate the interest to the seller that owns the item i, and (5) receive publications from the seller on i and notify b.

Implementation

We implement the three processes (buyer, seller, and broker) to faithfully conform to the process diagram above. Each client (buyer and seller) is implemented with two threads. One thread implements the user interface, while the other listens to the port in which the client is connected to the corresponding broker. The user interface thread waits for the user to provide a directive (see below). When a directive is provided (*e.g.*, for the buyer the user has suggested an automatic bid for an item), the system goes through the steps specified in the process diagram (*e.g.*, in this case, sending the bid to the broker).

A broker b has $l + m$ threads, where l is the number of other brokers connected to b and m is the number of clients. Each thread listens to a specific process (either another broker or client), and performs appropriate action whenever an event is received from its port. One part of the action is to “propagate” the event to other brokers connected to *b*; this is done by broadcasting the event. Other actions depend on the type of event. For instance, if the event received is `sale finalized`, *b* checks if there is any subscriber for item being sold connected to *b*; such subscribers then receive a notification of the sale.

User Interface

The user interface for the clients (buyers and sellers) is defined as a simple menu. There is no user interface for the brokers. The top-level user interface for the seller looks like the following.

```
Options :
 1. Publish a New Available Item.
 2. View Items Offered for Sale.
 3. Publish a Finalized Sale.
 4. Leave the Market.
```

When the user enters an option, the menu enters a sub-menu corresponding to that option. For instance, if the user uses Option 1, the sub-menu permits adding a new item, with name, attributes, minimum bid, etc. The menu is designed with an interactive, human user in mind. A thread in the process listens to the user interface and when an entire event is provided by the user (*e.g.*, the seller providing an available item with its name, attributes, and minimum bid), an event is generated and published.