Concurrent and Distributed Programming (PCD)

Session 5: an Auction system

Miguel Angel Vico Moya

1. Introduction

In this session we must make a program that implements an Auction system. With a structure like as the previous session, we implement an Agent program and a Bidder program in Erlang that simulates an auction system. The agent program follows the server structure of the previous session (chat system) and the bidder program follows the client structure.

2. Solutions

Following, we describe each program in detail:

- **agent.erl**:

  We can start the agent program with the "agent:start()" call. Then, the program asks the product, initial price, time to start and time to finish the auction to user. When the user enters this data, the process waits to the start time expired, then the auction starts.

  If a bidder enters to the auction before it starts, agent sends a message to bidder with the product and a message saying that the user must wait for auction start. When a bidder enters to the auction after it starts, agent sends a message to bidder with the product, actual maximum bidder and actual maximum bid.

  When a bidder joins or leaves from auction, the agent sends a message to other bidders with this information.

  In the case of bids, when a bidder makes a bid less or equal than actual maximum bid, this bid is ignored (when nobody has made a bid, then the agent permits a bid equal than initial price). If the bid is greater than actual maximum bid, then the agent registers this bid as the maximum and sends this information to other bidders.

  The agent permits to the bidders, leave from auction only in the case that the auction has not started or the bidder is not the maximum bidder, in other case, the agent send a message to the bidder saying that it can not leave from auction because is the maximum bidder.

  When the auction finishes, the agent sends a message with the winner of auction and the maximum bid to all connected bidders, then the process ends. If the auction finishes and nobody have made a bid, then the auction is restarted with the half of actual price.

To do all of these tasks, the process receives these messages:

- **{auction_start}**: When the time to start expires, the timer sends a message to the agent.
- **{auction_finish}**: When the time to finish expires, the timer sends a message to the agent. This timer is started when agent receives {auction_start} message.
- **{bidder_join_req, Name, From}**: When a bidder joins to auction, this message is received.
- **{bidder_leave_req, Name, From}**: When a bidder wants to leave from auction, agent receives this message.
- **{bidder_bid, Name, Bid}**: The agent receives this message when a bidder want to make a bid.

And sends these other:

- **{auction_start, CurrentBid}**: This message indicates that the auction starts with the initial price indicated by "CurrentBid".
- **{auction_finish, CurrentBidder, CurrentBid}**: This message indicates that the auction finished and the winner of auction is "CurrentBidder" with the "CurrentBid" bid.
- **{join, Name}**: This message indicates that the bidder with name "Name" has joined to the auction.
- **{leave, Name}**: This message indicates that the bidder with name "Name" has left from the auction.
- **{leave_req, ok}**: This message indicates that the bidder can leave from the auction.
- **{leave_req, no_ok}**: This message indicates that the bidder can not leave from the auction.
- **{auction_bid, Name, Bid}**: This message indicates that the bidder with name "Name" has made a bid with value of "Bid".
- **{welcome1, Product, CurrentBidder, CurrentBid}**: This message is sent to the bidders that join to the auction and it has been started. This message contains which is the product, the current maximum bidder and the current maximum bid in the auction.
- **{welcome2, Product}**: This message is sent to the bidders that join to the auction and it has not been started. This message contains which is the product in the auction.

- **server.erl**:

  We can start the bidder program with the "bidder:start('agent_process', 'Name of bidder')" call. Then the program creates two processes, one for the communication of the agent and other for interact with the user.

  At the first moment, the process sends to the agent process a message to request a join with the name of the bidder and the internal address (for communications between processes).

  Then, one of the processes only receives messages sent by the agent (see the agent description). All messages are used to print information of the auction except two of them: the {leave_req, ok} message and the {auction_finish, CurrentBidder, CurrentBid} message. In both cases, the bidder program ends.

The other process of the program, interacts with the user to get commands. There are only two commands: “exit” and “bid”. The “exit” command helps the bidder to leave the auction. The “bid” command permits to the bidder make a bid in the auction.