AUCTION IN DALI

Description

This DALI project aims to implement an auction among three players and an auctioneer in which some players can form an alliance to win if they did not win after a certain number of auction.

In detail…

There are two main roles in the auction: Auctioneer and Player. In this project we have one Auctioneer and three Player with different behavior.

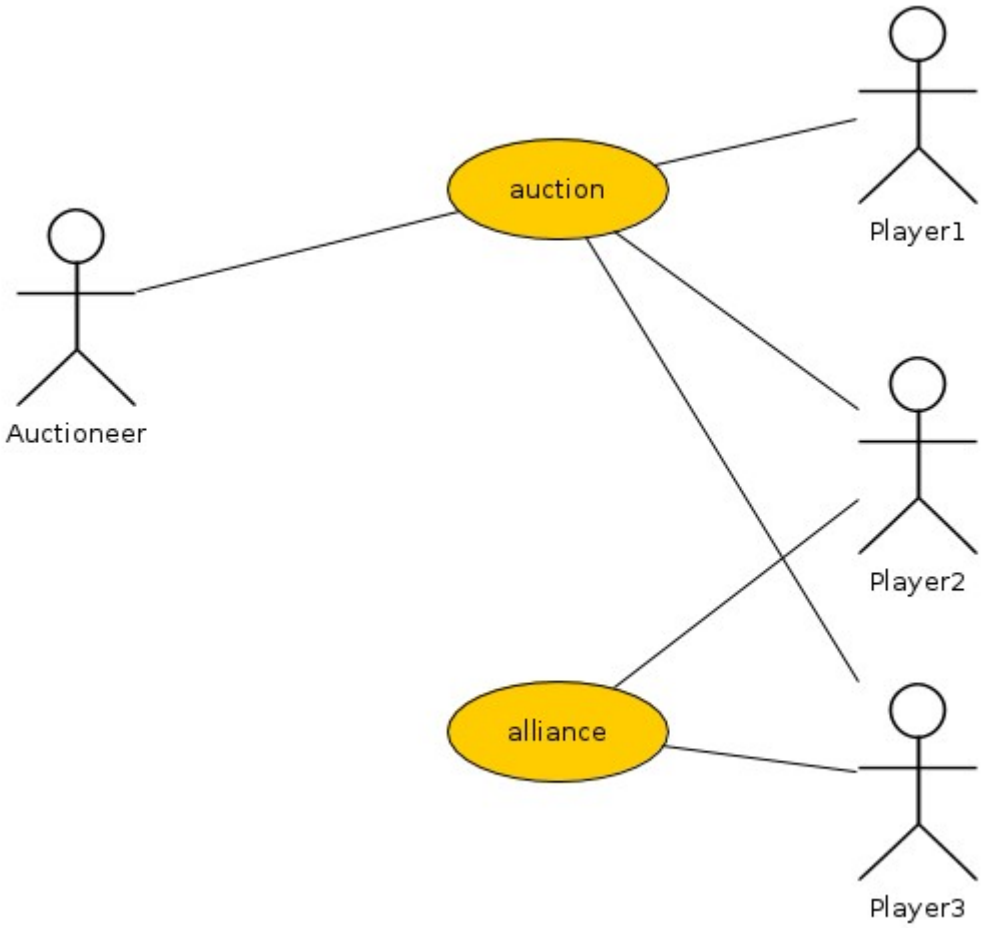


Figure 1 : Use Case

As illustrated in Figure 1 each player takes part in the auction and Player2 and Player3 can take part in an alliance.

Behavior

Auctioneer Agent

The auction starts when the *active_user* sends a start auction message to the *auctioneer.* The agent

then sends a new_auction message to all players agent to start the offering phase and then collect all the bids from them. Once every player has bidded it calculates the winner and announces it with a broadcast message.

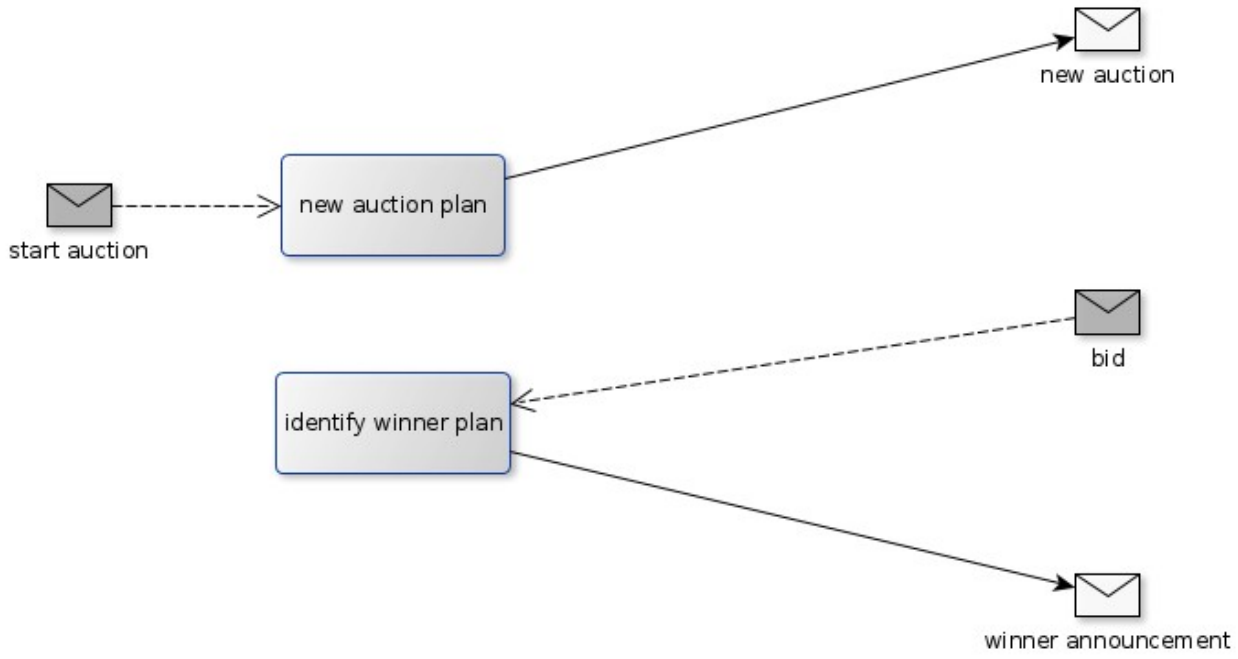


Figure 2 : Auctioneer Behavior

AUCTIONEER	Name	Preconditions
External Events	start_auction, place_bid	Sender = user
Internal Events	identify_winner	All players bidded in the current auction
Actions	winner_announcement [confirm]	identify_winner
	store_winner	Higher offer value than current max value

Player1 Agent

This Player always bids a fixed number (i.e.; 6) when it receives a new_auction message. At a winner announcement message it exults if it is the winner, otherwise it just prints the winner.

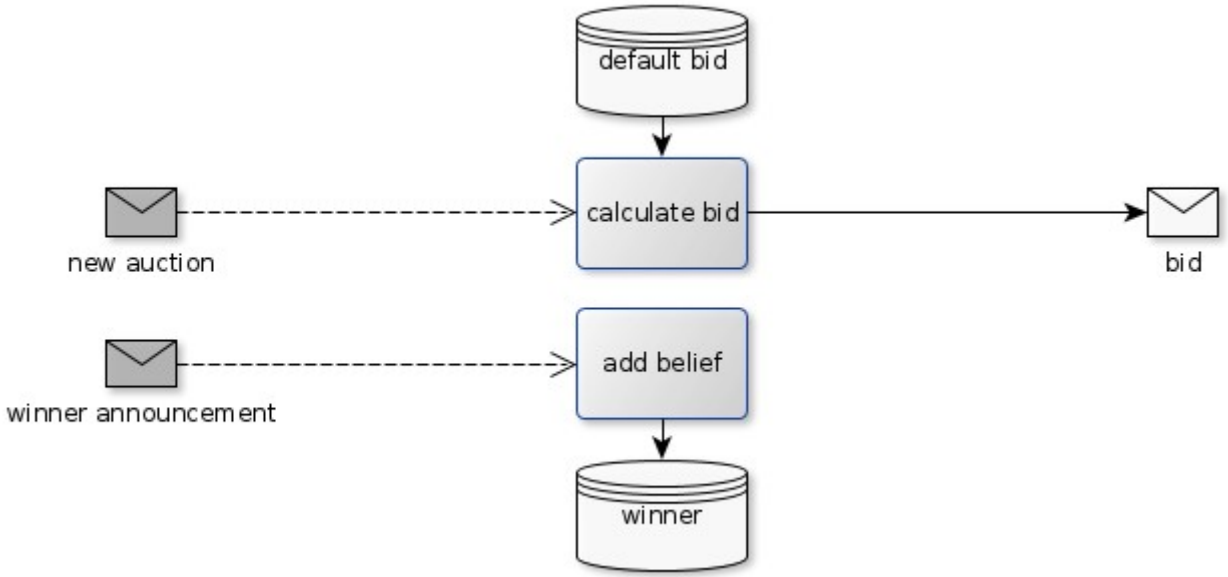


Figure 3: Player1 Behavior

PLAYER1	Name	Precondition
External Events	new_auction(N)	Sender = auctioneer
Internal Events	exult(N)	The agent is the winner
Actions	place_bid(6,M,N)	new_auction received

Player2 Agent

This Player bids value 4 by default, but if it receives an alliance proposal it bids 0 for the next auctions. At a winner announcement message it exults if it is the winner, otherwise it just prints the winner.

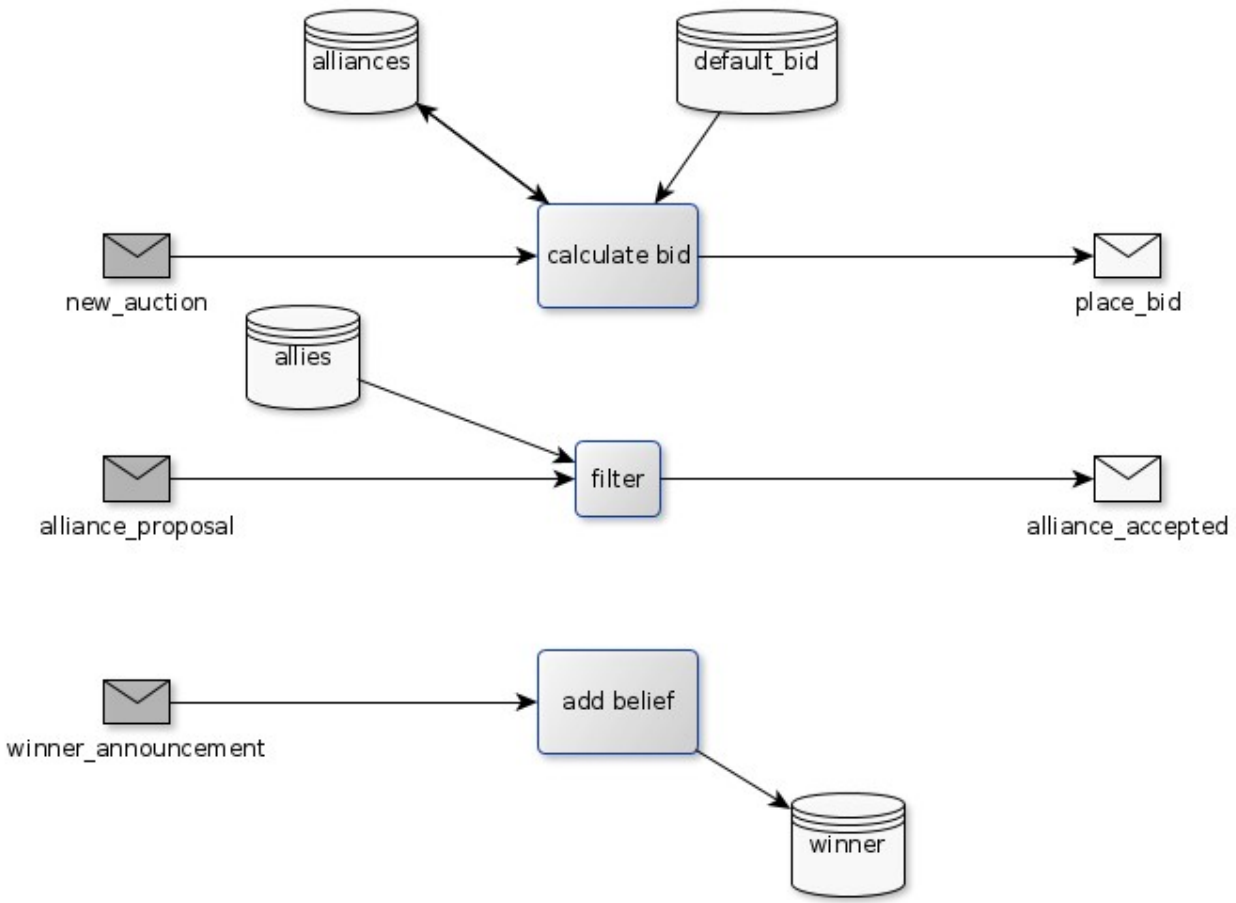


Figure 4: Player2 Behavior

PLAYER2	Name	Precondition
External Events	new_auction(N)	Sender = auctioneer
	alliance_proposal(A)	Sender is a potential ally
Internal Events	exult(N)	The agent is the winner
Actions	place_bid(0,M,N)	new_auction received, with alliance
	place_bid(V,M,N)	new_auction received, no alliance

Player3 Agent

This Player bids 3 by default, but after it looses 3 consecutive auctions it proposes an alliance to player2 to try to win summing up its default bid value and the player2's bid value. As usual at a winner announcement message it exults if it wins, otherwise it just prints the winner's name.

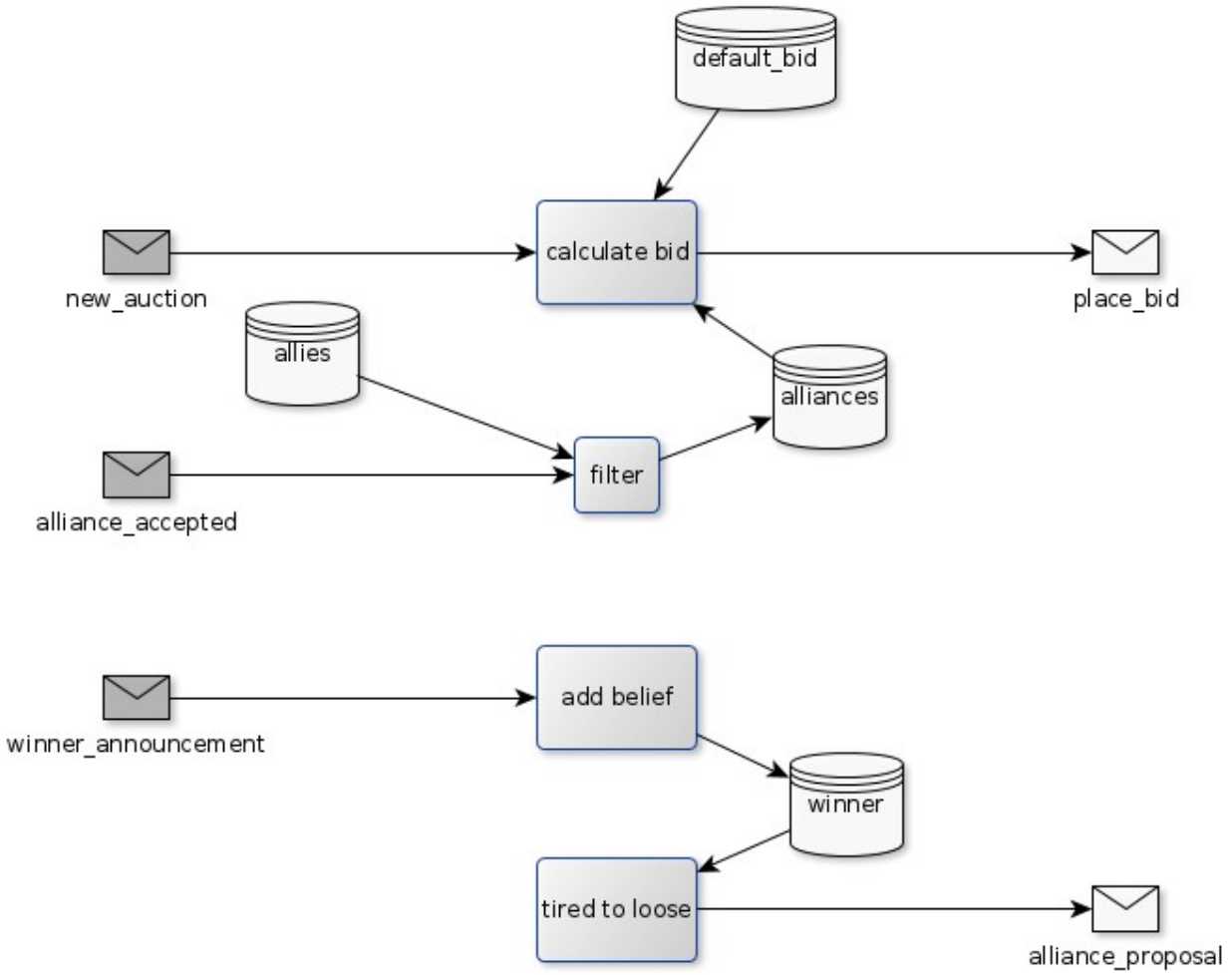


Figure 5: Player3 Behavior

PLAYER3	Name	Precondition
External Events	new_auction(N)	Sender = auctioneer
	alliance_accepted(A)	Sender is a potential ally
Internal Events	exult(N)	The agent is the winner
	tired_to_loose	Tree consecutive losses
Actions	place_bid(0,M,N)	new_auction received, with alliance
	place_bid(V,M,N)	new_auction received, no alliance

Deployment

The communication policy among agents is realized by modifiyng the *communication.con* file, so for an easy installation it was decided to pack all the project files and directory structure.

For GNU/Linux

The project is made up of:

- conf directory – configuration directory in which it is possible to change the communication rules between agents;
- *work directory* – in which the interpreter puts the instancieted agents;
- *log directory* – where the agent put in its own file the actions made during the execution like the sended messages;
- *mas directory* – contains other two directories: types and instaces in which are placed the source code agents' files;
- *4 type files* - one for each agent, respectively auctioneerType, player1Type, player2Type, player3Type;
- *4 instance files* – one for each agent, that contains the referred agent type: auctioneer.txt → auctioneerType, sue.txt → player1Type, jim.txt → player2Type, bob.txt → player3Type;
- *launcher* – script which contains path configuration and starts the MAS.

In a working DALI installation create a directory in the DALI root and name it “Auction”. The “GNU_linux” project folder must be putted inside this new directory.

Note: changes on the launcher file may be done for Sicstus path location.

For Windows

The structure remains the same except for the mas directory which contain directly all the source code agents file respectively: auctioneer.txt, player1.txt, player2.txt, player3.txt.

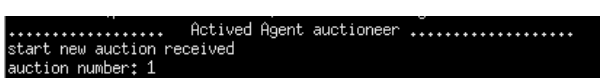
In a working DALI installation create a directory in the DALI root and name it “Auction”. The “Windows” folder must be putted inside this new directory.

Run the Project

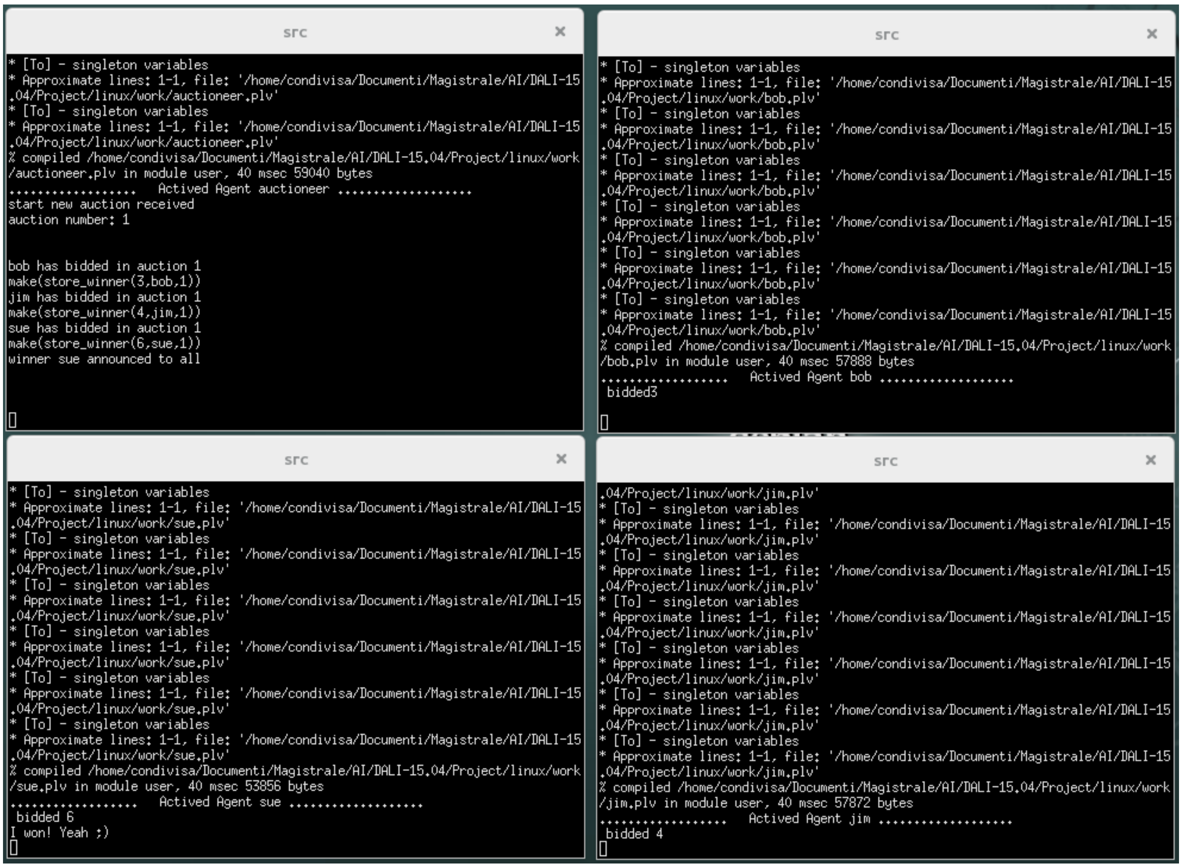
To start the project execute the launcher *startmas.sh* or *startmas.bat* file.

Start an auction

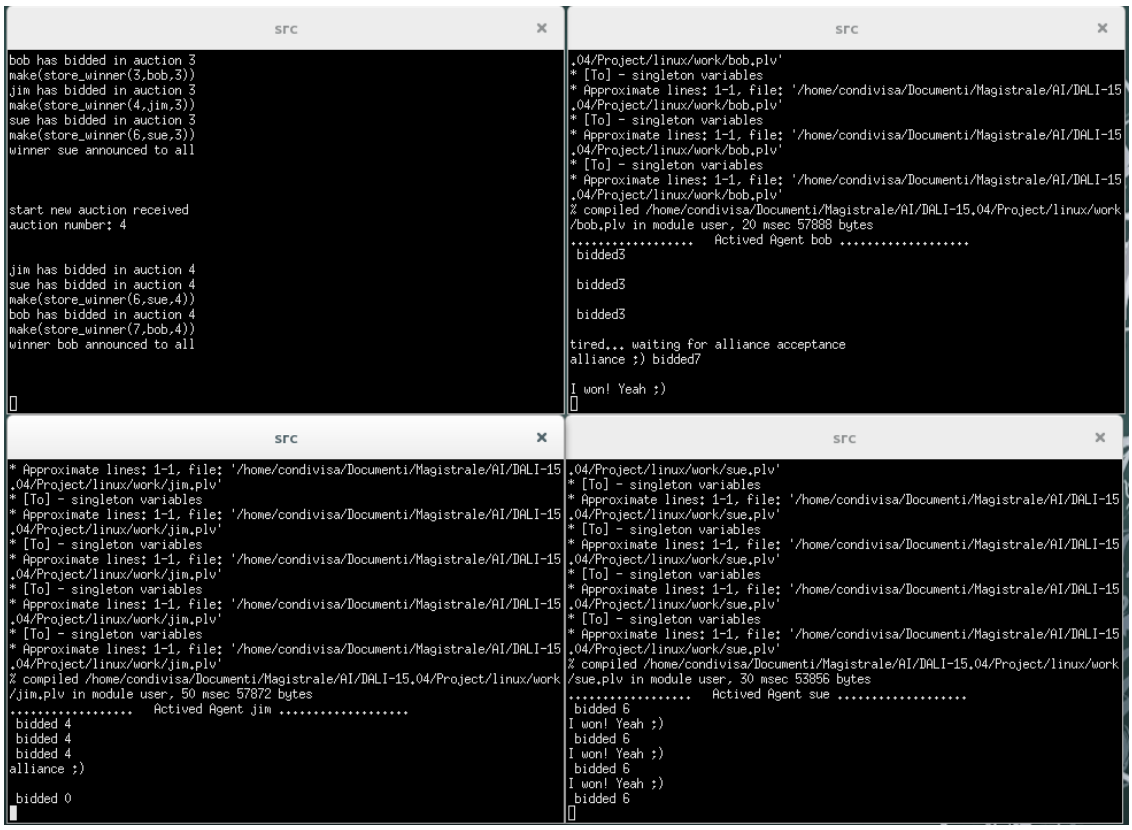
Once all the agents are activated, a new auction can be triggered by the user using a send_message to the auctioneer from the *active_user.pl* agent like below:



At this stage the agents start the bid phase and communicate their bid value to the auctioneer. No user interaction is needed to go to the winner identification phase. Now the auction ended and the agents printed all the information regarding their actions.



The Player agent “sue” won because its value is the greatest. After three auction (see **start an auction** section above) the agent “bob” raise an alliance request to agent “jim” that accept as expected. From now “jim” will bid 0 and “bob” 7 (3+4) and bob will win the succeeding auctions.



The alliance is kept until the alliance past events are discarded from the agents “jim” and “bob” (default 60 seconds), after that the agents behave like the first auction and the agent “bob” will become tired again after other 3 auctions.

Issues

The announce_winner message should be realized with a *confirm* message, but in this project it was found a problem with such type of FIPA message in the sense that the message doesn't add the winner belief in the receiving agents. So it was replaced by a *send_message* and it was introduced an announce_winnerE external event on the Player agents. The Player's *exult* internal event has the same logic but the winner predicate is asserted in the body of the new external event instead of be provided by the auctioneer without a DALI event.
The introduction of an external event for each Player agent may use more HW resoures than the intended SW project with the confirm message.