# Exercise 3

Construct a topology with 3 processes, use 1000 ms latency between the nodes. Experiment with the following two scenarios:

1:D2200:P1-1

2:D2000:P1-2

3:D2000:P1-3

and

1:D2000:P1-1

2:D2200:P1-2

3:D2200:P1-3

What value is decided for Paxos instance 1 in these two scenarios? Give reasoning in your report why that particular value has been choosen.

## Answer

We implemented following scenarios and topology:

Topology topologyEx3 = **new** Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

node(3, "127.0.0.1", 22033);

defaultLinks(1000, 0);

}

};

Scenario scenarioEx3\_1 = **new** Scenario(Assignement4Main.**class**) {

{;

command(1, "D2200:P1-1");

command(2, "D2000:P1-2");

command(3, "D2000:P1-3");

}

};

Scenario scenarioEx3\_2 = **new** Scenario(Assignement4Main.**class**) {

{;

command(1, "D2000:P1-1");

command(2, "D2200:P1-2");

command(3, "D2200:P1-3");

}

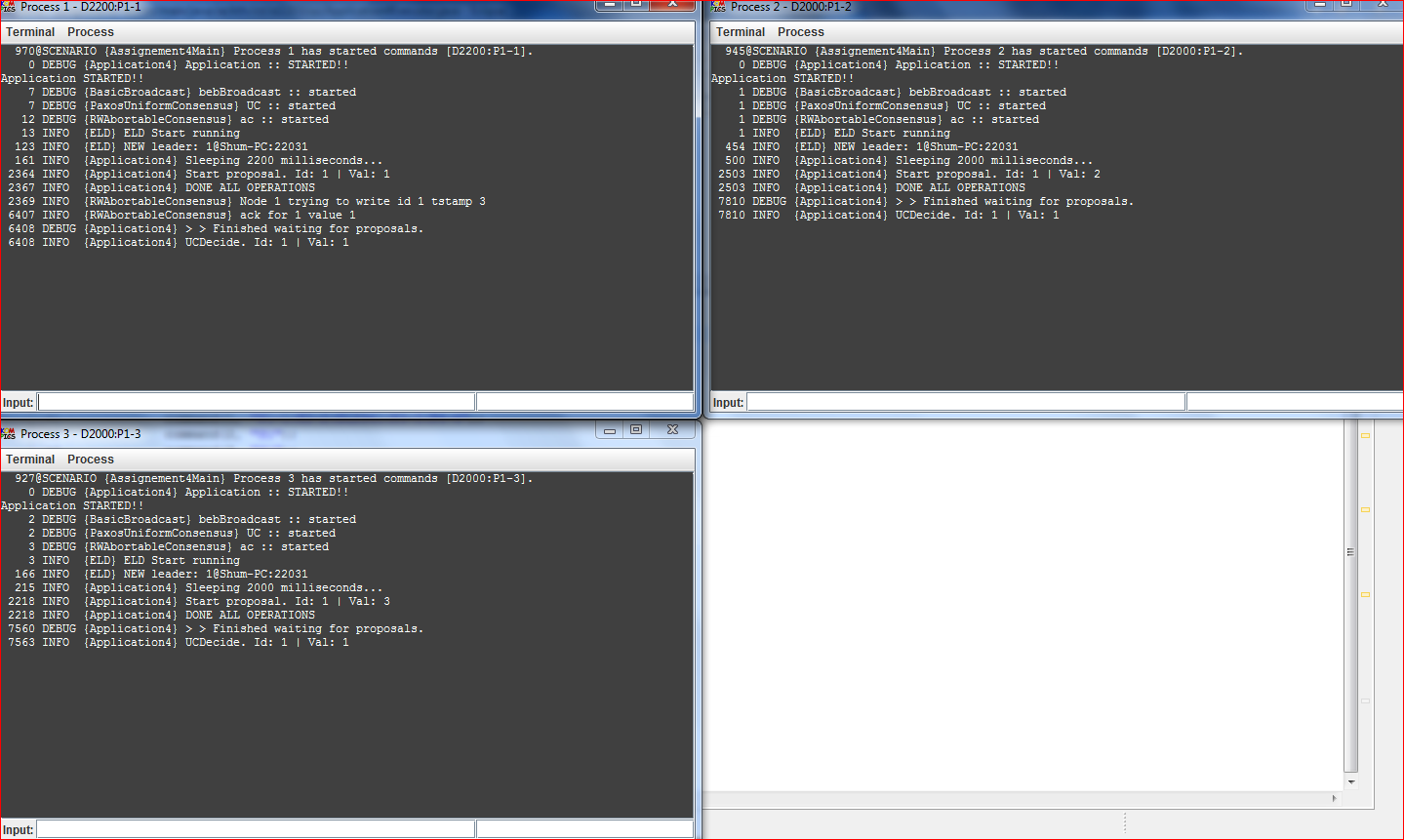
};

We also used following timings for ELD:

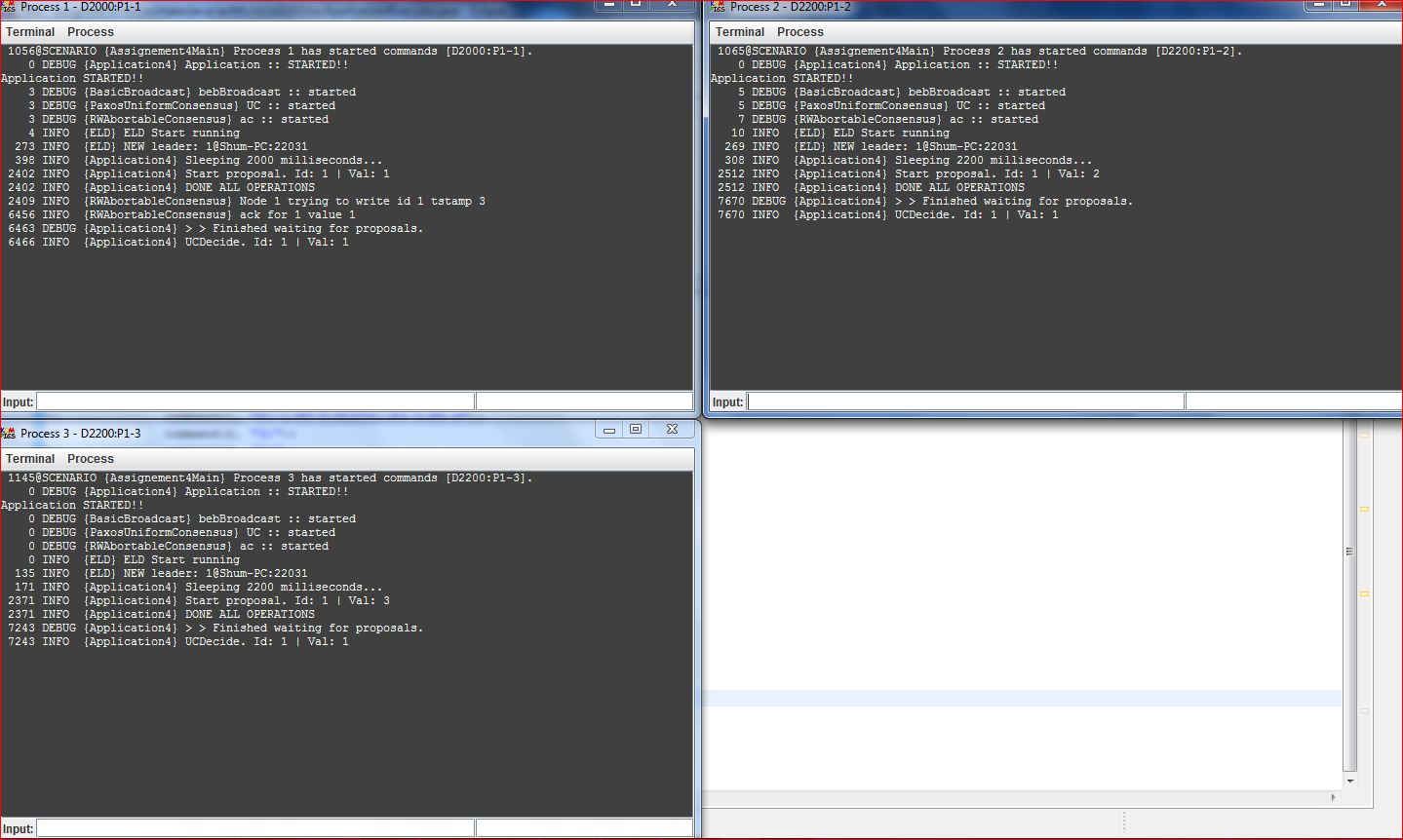
**private** **static** **long** *delta* = 1000;

**private** **static** **long** *timeDelay* = 2000;

Results of the first scenario are:



Results of the second scenario are:



We can see that in both cases nodes decided on value 1 and it didn’t depend on initial delay. This value was chosen because node 1 has been chosen as a leader and this is the only node that actually tries to write its value. Therefore initial delays do not influence on the execution of the algorithm.

# Exercise 5

In the original presentation of the Paxos algorithm, processes can have diﬀerent roles: proposer, acceptor, and learner. For each event handler of AC and PUC say processes with what role are meant to execute the respective handler. In other words, what types of messages are sent and expected (and what events are triggered and expected) by each process role. Of course one process can act in more than one role. In fact, in our case, processes act in all roles, but interestingly, they need not to. Please refer to lecture 11 and the Paxos paper available on the course webpage.

## Answer

For AC:

Upon event ⟨ acPropose | id, v ⟩ do – proposer

Upon event ⟨ bebDeliver | pj ,[Read,id,ts] ⟩ do – acceptor

Upon event ⟨ pp2pDeliver | pj ,[ ,id] ⟩ do – proposer

Upon event ⟨ pp2pDeliver | pj ,[ReadAck,id,ts,v,sentts] ⟩ do – proposer

Upon event ⟨ bebDeliver | pj ,[Write,id,ts,v] ⟩ do – acceptor

uponevent ⟨ pp2pDeliver | pj ,[WriteAck,id,sentts] ⟩ do – proposer

For PUC

Upon event ⟨ trust | pi ⟩ do – proposer

Upon event ⟨ ucPropose | id, v ⟩ do – proposer

Upon event ⟨ acReturn | id,result ⟩ do – proposer

Upon event ⟨ bebDeliver | pi,[Decided,id,v] ⟩ do - learner