# Exercise 1

## experiments

We tried the PFD with both 2 and 4 nodes in fully connected topologies.

Topology topology1 = **new** Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

link(1, 2, 4000, 0).bidirectional();

}

};

Topology topology2 = new Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

node(3, "127.0.0.1", 22033);

node(4, "127.0.0.1", 22034);

defaultLinks(1000, 0);

}

};

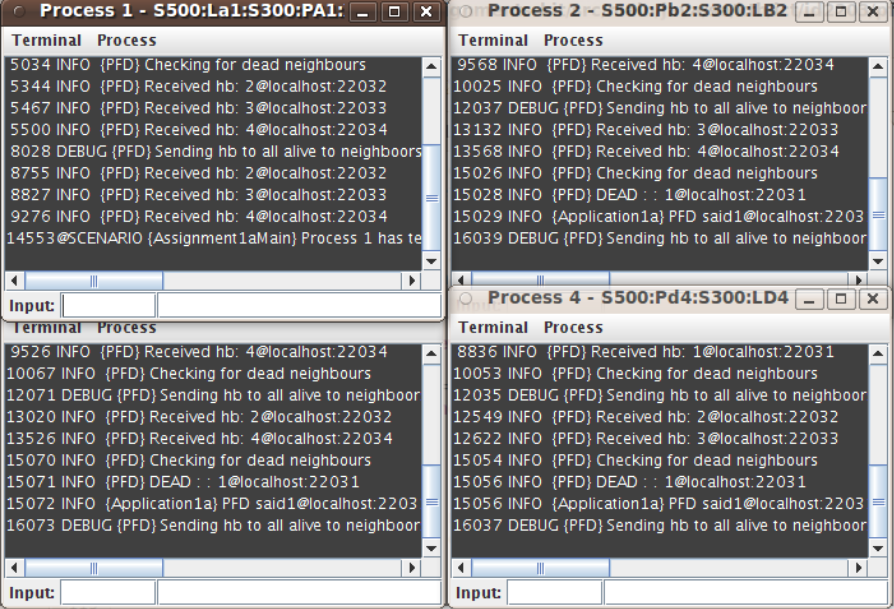
With the following values of γ and δ:

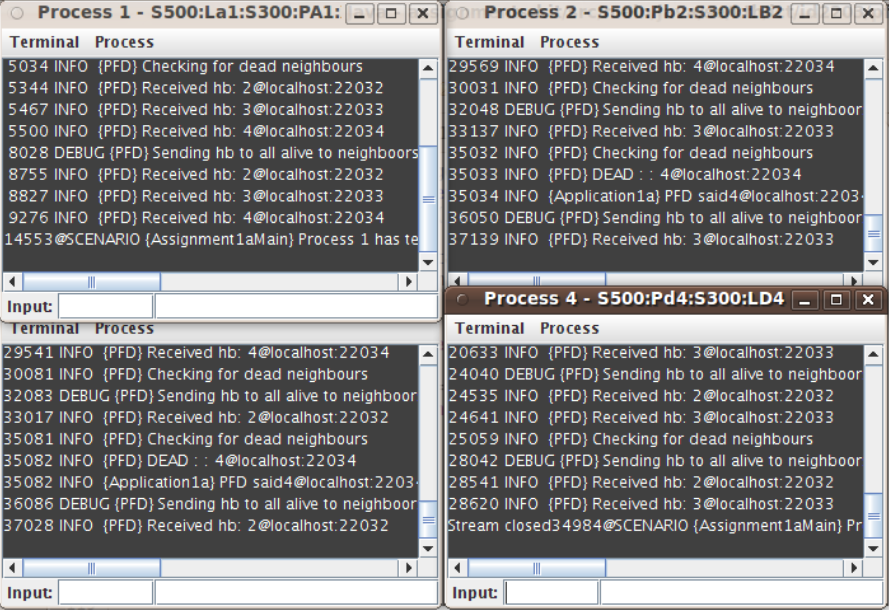
**private** **static** **long** *GAMMA* = 4000;

**private** **static** **long** *DELTA* = 1000;

## observations

The PFD worked as intended, reporting the crash of a node as you can observe in the following screenshots:





## conclusions

Our test showed that the algorithm and the implementation of the PFD works correctly and is consistent to the completeness property of it.

# Exercise 2

## experiments

We tried the EPFD with both link delay shorter than the initial heartbeat period and longer than this. Also, in the first case, we also killed one process to test that the other PFD is detecting the failure.

Topology topology1 = **new** Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

link(1, 2, 3213, 0).bidirectional();

}

};

Topology topology2 = **new** Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

link(1, 2, 4000, 0).bidirectional();

}

};

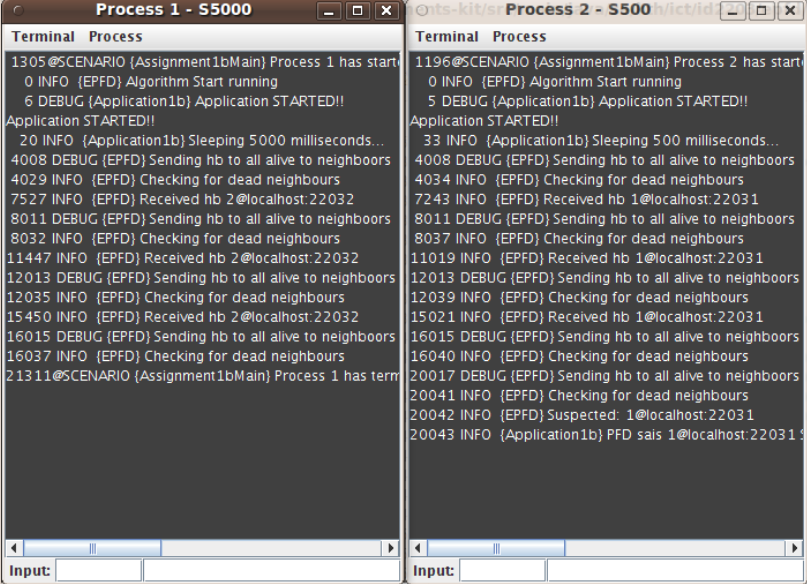
With the following values of TimeDelay and Δ:

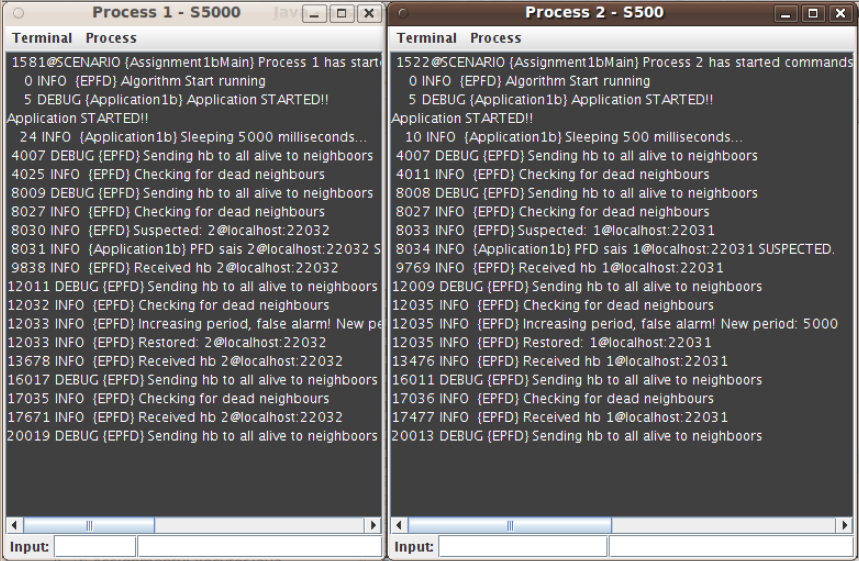
**private** **static** **long** *TIMEDELAY* = 4000;

**private** **static** **long** *DELTA* = 1000;

## Observations

The EPFD worked as intended, reporting the crash of a node in the 1st case and stabilizing in the 2nd case, as you can observe in the following screenshots:





## conclusions

Our test showed that the algorithm and the implementation of the EPFD in the specific test cases are not that good. What is happening is that after the 1st stabilization of the period time, the heartbeat events keep coming every TimeDelay ms, so a second stabilization does not happen. Actually, this is not exactly a problem, because the EPFD correctly do not suspect the nodes, since they are not crashed.

# Exercise 3

In this exercise we used EPFD with FairLoss links to simulate loss of the messages. We used the following topology:

Topology topology1 = **new** Topology() {

{

node(1, "127.0.0.1", 22033);

node(2, "127.0.0.1", 22034);

link(1, 2, 1500, 0.7).bidirectional();

}

};

Latency = 1500 Ms;

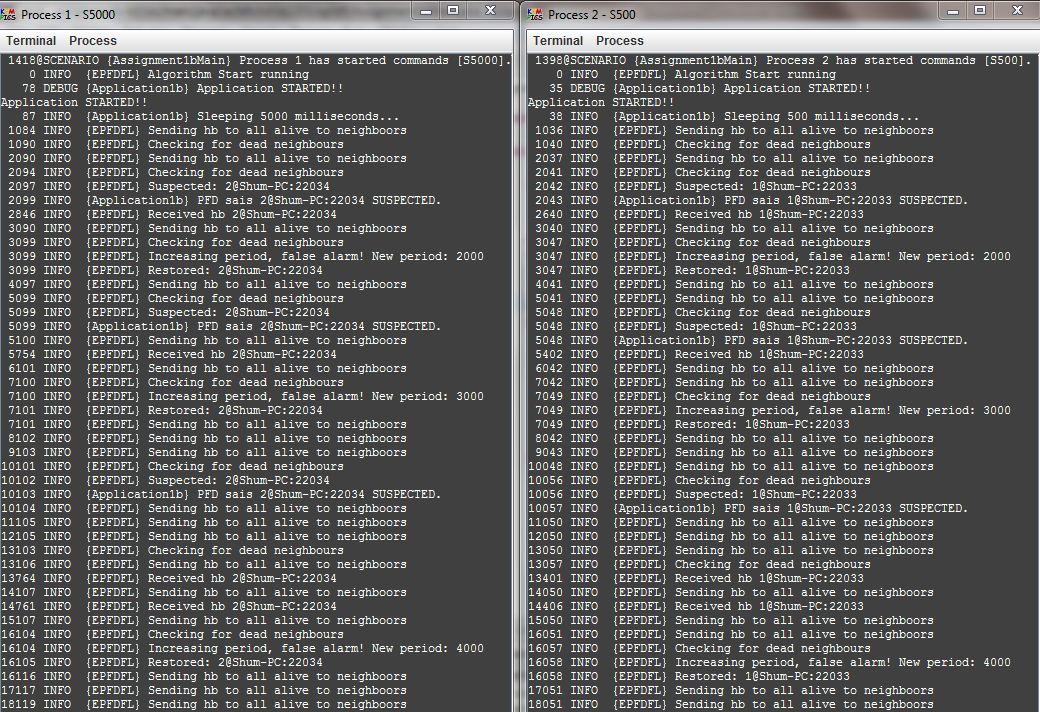
LossRate = 0.7;

TimeDelay = 1000 Ms;

Delta = 1000 Ms.

We used big loss rate to make adjustment of TimeDelay more often and visible.

## Observations



## Conclusion.

We see that in this scenario TimeDelay increases to accommodate large delays when messages are lost. We observed that the larger is probability of loosing messages the faster grows TimeDelay. In the run shown above after 137 second TimeDelay increased to 8000 Ms.

# Exercise 4

## experiments

We implemented the doRecover method, that is being run when a process recovers from a failure. It just prints an informational message to the logger. The topology and scenario that we used:

Topology topology1 = **new** Topology() {

{

node(1, "127.0.0.1", 22031);

node(2, "127.0.0.1", 22032);

link(1, 2, 3213, 0).bidirectional();

}

};

Scenario scenario3 = **new** Scenario(Assignment1bMain.**class**) {

{

command(1, "S500:Lmsg1:S6000:X").recover("R:S500:Pmsg3:S500:X", 10000);

command(2, "S500");

}

};

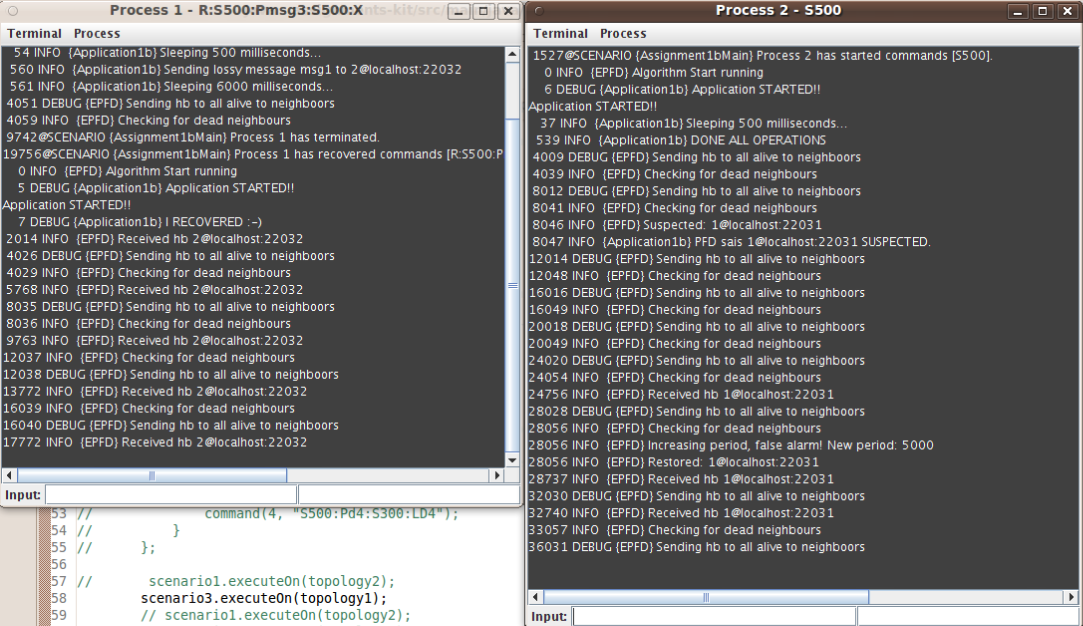
With the following values of TimeDelay and Δ:

**private** **static** **long** *TIMEDELAY* = 4000;

**private** **static** **long** *DELTA* = 1000;

## Observations

The results can be seen in the following screenshot:



## conclusions

As we expected, the EPFD correctly suspected the process that failed. This is absolutely normal, since in exercise 2 we tested that the EPFD detected a crashed node. Moreover, we can see that after the node recovers, it starts sending heartbeats again, and the other node calls a “false alarm” about the suspected event. Finally, we can conclude that the nodes that suspect the crashed node, when this node recovers, they are not able to distinguish between a node failure and recovery, an omission failure or just if the period of checking was not set to the correct value.