



ROYAL INSTITUTE
OF TECHNOLOGY
OF TECHNOLOGY
ROYAL INSTITUTE

Royal Institute of Technology

MSc. Software Engineering of Distributed Systems

ID2203 Distributed Systems Advanced Course

Homework 1

Andrei Shumanski
andreish@kth.se
0046707761992

Trigonakis Vasileios
vtri@kth.se
0046707694420

Stockholm 2010

TABLE OF CONTENTS

Exercise 1	4
experiments	4
observations.....	4
conclusions.....	6
Exercise 2	7
experiments	7
Observations	7
conclusions.....	9
Exercise 3	10
Observations	11
Conclusion.....	11
Exercise 4	12
experiments	12
Observations	13
conclusions.....	13
Question 1.....	14
Question 2.....	15
Question 3.....	17

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();
    }
};

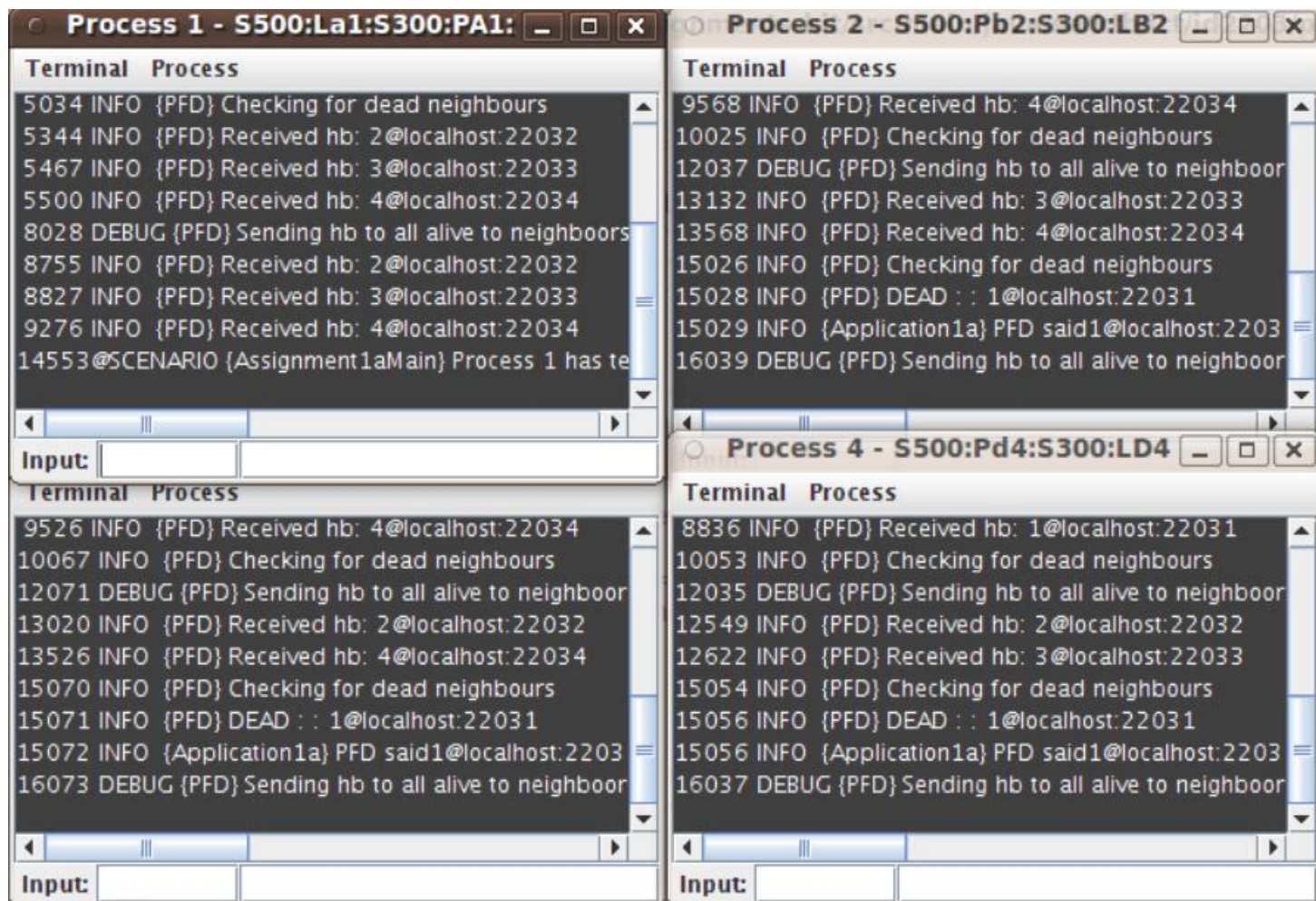
Topology topology2 = new Topology() {
    {
        node(1, "127.0.0.1", 22031);
        node(2, "127.0.0.1", 22032);
        node(3, "127.0.0.1", 22033);
        node(4, "127.0.0.1", 22034);
        defaultLinks(1000, 0);
    }
};
```

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:



```
Process 1 - S500:La1:S300:PA1:
Terminal Process
5034 INFO {PFD} Checking for dead neighbours
5344 INFO {PFD} Received hb: 2@localhost:22032
5467 INFO {PFD} Received hb: 3@localhost:22033
5500 INFO {PFD} Received hb: 4@localhost:22034
8028 DEBUG {PFD} Sending hb to all alive to neighbours
8755 INFO {PFD} Received hb: 2@localhost:22032
8827 INFO {PFD} Received hb: 3@localhost:22033
9276 INFO {PFD} Received hb: 4@localhost:22034
14553@SCENARIO {Assignment1aMain} Process 1 has te

Process 2 - S500:Pb2:S300:LB2
Terminal Process
29569 INFO {PFD} Received hb: 4@localhost:22034
30031 INFO {PFD} Checking for dead neighbours
32048 DEBUG {PFD} Sending hb to all alive to neighbor
33137 INFO {PFD} Received hb: 3@localhost:22033
35032 INFO {PFD} Checking for dead neighbours
35033 INFO {PFD} DEAD : : 4@localhost:22034
35034 INFO {Application1a} PFD said4@localhost:2203-
36050 DEBUG {PFD} Sending hb to all alive to neighbor
37139 INFO {PFD} Received hb: 3@localhost:22033

Process 3 - S500:La3:S300:PA3
Terminal Process
29541 INFO {PFD} Received hb: 4@localhost:22034
30081 INFO {PFD} Checking for dead neighbours
32083 DEBUG {PFD} Sending hb to all alive to neighbor
33017 INFO {PFD} Received hb: 2@localhost:22032
35081 INFO {PFD} Checking for dead neighbours
35082 INFO {PFD} DEAD : : 4@localhost:22034
35082 INFO {Application1a} PFD said4@localhost:2203-
36086 DEBUG {PFD} Sending hb to all alive to neighbor
37028 INFO {PFD} Received hb: 2@localhost:22032

Process 4 - S500:Pd4:S300:LD4
Terminal Process
20633 INFO {PFD} Received hb: 3@localhost:22033
24040 DEBUG {PFD} Sending hb to all alive to neighbor
24535 INFO {PFD} Received hb: 2@localhost:22032
24641 INFO {PFD} Received hb: 3@localhost:22033
25059 INFO {PFD} Checking for dead neighbours
28042 DEBUG {PFD} Sending hb to all alive to neighbor
28541 INFO {PFD} Received hb: 2@localhost:22032
28620 INFO {PFD} Received hb: 3@localhost:22033
Stream closed34984@SCENARIO {Assignment1aMain} Pr
```

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:

Process 1 - S5000

Terminal Process

1305@SCENARIO {Assignment1bMain} Process 1 has started
0 INFO {EPFD} Algorithm Start running
6 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
20 INFO {Application1b} Sleeping 5000 milliseconds...
4008 DEBUG {EPFD} Sending hb to all alive to neighbors
4029 INFO {EPFD} Checking for dead neighbours
7527 INFO {EPFD} Received hb 2@localhost:22032
8011 DEBUG {EPFD} Sending hb to all alive to neighbors
8032 INFO {EPFD} Checking for dead neighbours
11447 INFO {EPFD} Received hb 2@localhost:22032
12013 DEBUG {EPFD} Sending hb to all alive to neighbors
12035 INFO {EPFD} Checking for dead neighbours
15450 INFO {EPFD} Received hb 2@localhost:22032
16015 DEBUG {EPFD} Sending hb to all alive to neighbors
16037 INFO {EPFD} Checking for dead neighbours
21311@SCENARIO {Assignment1bMain} Process 1 has terminated

Input:

Process 2 - S5000

Terminal Process

1196@SCENARIO {Assignment1bMain} Process 2 has started
0 INFO {EPFD} Algorithm Start running
5 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
33 INFO {Application1b} Sleeping 500 milliseconds...
4008 DEBUG {EPFD} Sending hb to all alive to neighbors
4034 INFO {EPFD} Checking for dead neighbours
7243 INFO {EPFD} Received hb 1@localhost:22031
8011 DEBUG {EPFD} Sending hb to all alive to neighbors
8037 INFO {EPFD} Checking for dead neighbours
11019 INFO {EPFD} Received hb 1@localhost:22031
12013 DEBUG {EPFD} Sending hb to all alive to neighbors
12039 INFO {EPFD} Checking for dead neighbours
15021 INFO {EPFD} Received hb 1@localhost:22031
16015 DEBUG {EPFD} Sending hb to all alive to neighbors
16040 INFO {EPFD} Checking for dead neighbours
20017 DEBUG {EPFD} Sending hb to all alive to neighbors
20041 INFO {EPFD} Checking for dead neighbours
20042 INFO {EPFD} Suspected: 1@localhost:22031
20043 INFO {Application1b} PFD says 1@localhost:22031

Input:

Process 1 - S5000

Terminal Process

1581@SCENARIO {Assignment1bMain} Process 1 has started
0 INFO {EPFD} Algorithm Start running
5 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
24 INFO {Application1b} Sleeping 5000 milliseconds...
4007 DEBUG {EPFD} Sending hb to all alive to neighbors
4025 INFO {EPFD} Checking for dead neighbours
8009 DEBUG {EPFD} Sending hb to all alive to neighbors
8027 INFO {EPFD} Checking for dead neighbours
8030 INFO {EPFD} Suspected: 2@localhost:22032
8031 INFO {Application1b} PFD sais 2@localhost:22032 S
9838 INFO {EPFD} Received hb 2@localhost:22032
12011 DEBUG {EPFD} Sending hb to all alive to neighbors
12032 INFO {EPFD} Checking for dead neighbours
12033 INFO {EPFD} Increasing period, false alarm! New pe
12033 INFO {EPFD} Restored: 2@localhost:22032
13678 INFO {EPFD} Received hb 2@localhost:22032
16017 DEBUG {EPFD} Sending hb to all alive to neighbors
17035 INFO {EPFD} Checking for dead neighbours
17671 INFO {EPFD} Received hb 2@localhost:22032
20019 DEBUG {EPFD} Sending hb to all alive to neighbors

Input:

Process 2 - S500

Terminal Process

1522@SCENARIO {Assignment1bMain} Process 2 has started commands
0 INFO {EPFD} Algorithm Start running
5 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
10 INFO {Application1b} Sleeping 500 milliseconds...
4007 DEBUG {EPFD} Sending hb to all alive to neighbors
4011 INFO {EPFD} Checking for dead neighbours
8008 DEBUG {EPFD} Sending hb to all alive to neighbors
8027 INFO {EPFD} Checking for dead neighbours
8033 INFO {EPFD} Suspected: 1@localhost:22031
8034 INFO {Application1b} PFD sais 1@localhost:22031 SUSPECTED.
9769 INFO {EPFD} Received hb 1@localhost:22031
12009 DEBUG {EPFD} Sending hb to all alive to neighbors
12035 INFO {EPFD} Checking for dead neighbours
12035 INFO {EPFD} Increasing period, false alarm! New period: 5000
12035 INFO {EPFD} Restored: 1@localhost:22031
13476 INFO {EPFD} Received hb 1@localhost:22031
16011 DEBUG {EPFD} Sending hb to all alive to neighbors
17036 INFO {EPFD} Checking for dead neighbours
17477 INFO {EPFD} Received hb 1@localhost:22031
20013 DEBUG {EPFD} Sending hb to all alive to neighbors

Input:

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

```
Process 1 - S5000
Terminal Process
1418@SCENARIO {Assignment1bMain} Process 1 has started commands [S5000].
0 INFO {EPFDFL} Algorithm Start running
78 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
87 INFO {Application1b} Sleeping 5000 milliseconds...
1084 INFO {EPFDFL} Sending hb to all alive to neighbours
1090 INFO {EPFDFL} Checking for dead neighbours
2090 INFO {EPFDFL} Sending hb to all alive to neighbours
2094 INFO {EPFDFL} Checking for dead neighbours
2097 INFO {EPFDFL} Suspected: 2@Shum-PC:22034
2099 INFO {Application1b} PFD sais 2@Shum-PC:22034 SUSPECTED.
2846 INFO {EPFDFL} Received hb 2@Shum-PC:22034
3090 INFO {EPFDFL} Sending hb to all alive to neighbours
3099 INFO {EPFDFL} Checking for dead neighbours
3099 INFO {EPFDFL} Increasing period, false alarm! New period: 2000
3099 INFO {EPFDFL} Restored: 2@Shum-PC:22034
4097 INFO {EPFDFL} Sending hb to all alive to neighbours
5099 INFO {EPFDFL} Checking for dead neighbours
5099 INFO {EPFDFL} Suspected: 2@Shum-PC:22034
5099 INFO {Application1b} PFD sais 2@Shum-PC:22034 SUSPECTED.
5100 INFO {EPFDFL} Sending hb to all alive to neighbours
5754 INFO {EPFDFL} Received hb 2@Shum-PC:22034
6101 INFO {EPFDFL} Sending hb to all alive to neighbours
7100 INFO {EPFDFL} Checking for dead neighbours
7100 INFO {EPFDFL} Increasing period, false alarm! New period: 3000
7101 INFO {EPFDFL} Restored: 2@Shum-PC:22034
7101 INFO {EPFDFL} Sending hb to all alive to neighbours
8102 INFO {EPFDFL} Sending hb to all alive to neighbours
9103 INFO {EPFDFL} Sending hb to all alive to neighbours
10101 INFO {EPFDFL} Checking for dead neighbours
10102 INFO {EPFDFL} Suspected: 2@Shum-PC:22034
10103 INFO {Application1b} PFD sais 2@Shum-PC:22034 SUSPECTED.
10104 INFO {EPFDFL} Sending hb to all alive to neighbours
11105 INFO {EPFDFL} Sending hb to all alive to neighbours
12105 INFO {EPFDFL} Sending hb to all alive to neighbours
13103 INFO {EPFDFL} Checking for dead neighbours
13106 INFO {EPFDFL} Sending hb to all alive to neighbours
13764 INFO {EPFDFL} Received hb 2@Shum-PC:22034
14107 INFO {EPFDFL} Sending hb to all alive to neighbours
14761 INFO {EPFDFL} Received hb 2@Shum-PC:22034
15107 INFO {EPFDFL} Sending hb to all alive to neighbours
16104 INFO {EPFDFL} Checking for dead neighbours
16104 INFO {EPFDFL} Increasing period, false alarm! New period: 4000
16105 INFO {EPFDFL} Restored: 2@Shum-PC:22034
16116 INFO {EPFDFL} Sending hb to all alive to neighbours
17117 INFO {EPFDFL} Sending hb to all alive to neighbours
18119 INFO {EPFDFL} Sending hb to all alive to neighbours

Process 2 - S500
Terminal Process
1398@SCENARIO {Assignment1bMain} Process 2 has started commands [S500].
0 INFO {EPFDFL} Algorithm Start running
35 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
38 INFO {Application1b} Sleeping 500 milliseconds...
1036 INFO {EPFDFL} Sending hb to all alive to neighbours
1040 INFO {EPFDFL} Checking for dead neighbours
2037 INFO {EPFDFL} Sending hb to all alive to neighbours
2041 INFO {EPFDFL} Checking for dead neighbours
2042 INFO {EPFDFL} Suspected: 1@Shum-PC:22033
2043 INFO {Application1b} PFD sais 1@Shum-PC:22033 SUSPECTED.
2640 INFO {EPFDFL} Received hb 1@Shum-PC:22033
3040 INFO {EPFDFL} Sending hb to all alive to neighbours
3047 INFO {EPFDFL} Checking for dead neighbours
3047 INFO {EPFDFL} Increasing period, false alarm! New period: 2000
3047 INFO {EPFDFL} Restored: 1@Shum-PC:22033
4041 INFO {EPFDFL} Sending hb to all alive to neighbours
5041 INFO {EPFDFL} Sending hb to all alive to neighbours
5048 INFO {EPFDFL} Checking for dead neighbours
5048 INFO {EPFDFL} Suspected: 1@Shum-PC:22033
5048 INFO {Application1b} PFD sais 1@Shum-PC:22033 SUSPECTED.
5402 INFO {EPFDFL} Received hb 1@Shum-PC:22033
6042 INFO {EPFDFL} Sending hb to all alive to neighbours
7042 INFO {EPFDFL} Sending hb to all alive to neighbours
7049 INFO {EPFDFL} Checking for dead neighbours
7049 INFO {EPFDFL} Increasing period, false alarm! New period: 3000
7049 INFO {EPFDFL} Restored: 1@Shum-PC:22033
8042 INFO {EPFDFL} Sending hb to all alive to neighbours
9043 INFO {EPFDFL} Sending hb to all alive to neighbours
10048 INFO {EPFDFL} Sending hb to all alive to neighbours
10056 INFO {EPFDFL} Checking for dead neighbours
10056 INFO {EPFDFL} Suspected: 1@Shum-PC:22033
10057 INFO {Application1b} PFD sais 1@Shum-PC:22033 SUSPECTED.
11050 INFO {EPFDFL} Sending hb to all alive to neighbours
12050 INFO {EPFDFL} Sending hb to all alive to neighbours
13050 INFO {EPFDFL} Sending hb to all alive to neighbours
13057 INFO {EPFDFL} Checking for dead neighbours
13401 INFO {EPFDFL} Received hb 1@Shum-PC:22033
14050 INFO {EPFDFL} Sending hb to all alive to neighbours
14406 INFO {EPFDFL} Received hb 1@Shum-PC:22033
15050 INFO {EPFDFL} Sending hb to all alive to neighbours
16051 INFO {EPFDFL} Sending hb to all alive to neighbours
16057 INFO {EPFDFL} Checking for dead neighbours
16058 INFO {EPFDFL} Increasing period, false alarm! New period: 4000
16058 INFO {EPFDFL} Restored: 1@Shum-PC:22033
17051 INFO {EPFDFL} Sending hb to all alive to neighbours
18051 INFO {EPFDFL} Sending hb to all alive to neighbours
```

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:

The screenshot displays two terminal windows side-by-side, showing the output of a network simulation. The left window, titled "Process 1 - R:S500:Pmsg3:S500:Xnts-kit/src", shows logs for Process 1. It starts with a sleep of 500 milliseconds, then sends a lossy message to 2@localhost:22032. After a 6000 millisecond sleep, it checks for dead neighbors. At 19742, the scenario reports that Process 1 has terminated. At 19756, it reports that Process 1 has recovered. The logs then show the application starting and sending heartbeats to 2@localhost:22032. The right window, titled "Process 2 - S500", shows logs for Process 2. It starts with a sleep of 500 milliseconds, then sends a message to 1@localhost:22031. At 8047, it reports that Process 1 is suspected. At 12014, it reports that Process 1 has recovered. The logs then show the application starting and sending heartbeats to 1@localhost:22031. Below the terminal windows, there is a code editor showing the source code for the simulation. The code includes comments and function calls for sending messages, checking for dead neighbors, and sending heartbeats.

```
Process 1 - R:S500:Pmsg3:S500:Xnts-kit/src
Terminal Process
54 INFO {Application1b} Sleeping 500 milliseconds...
560 INFO {Application1b} Sending lossy message msg1 to 2@localhost:22032
561 INFO {Application1b} Sleeping 6000 milliseconds...
4051 DEBUG {EPFD} Sending hb to all alive to neighbours
4059 INFO {EPFD} Checking for dead neighbours
9742@SCENARIO {Assignment1bMain} Process 1 has terminated.
19756@SCENARIO {Assignment1bMain} Process 1 has recovered commands [R:S500:P
0 INFO {EPFD} Algorithm Start running
5 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
7 DEBUG {Application1b} I RECOVERED :-}
2014 INFO {EPFD} Received hb 2@localhost:22032
4026 DEBUG {EPFD} Sending hb to all alive to neighbours
4029 INFO {EPFD} Checking for dead neighbours
5768 INFO {EPFD} Received hb 2@localhost:22032
8035 DEBUG {EPFD} Sending hb to all alive to neighbours
8036 INFO {EPFD} Checking for dead neighbours
9763 INFO {EPFD} Received hb 2@localhost:22032
12037 INFO {EPFD} Checking for dead neighbours
12038 DEBUG {EPFD} Sending hb to all alive to neighbours
13772 INFO {EPFD} Received hb 2@localhost:22032
16039 INFO {EPFD} Checking for dead neighbours
16040 DEBUG {EPFD} Sending hb to all alive to neighbours
17772 INFO {EPFD} Received hb 2@localhost:22032

Input:

53 // command(4, "S500:Pd4:S300:LD4");
54 // }
55 // };
56
57 // scenario1.executeOn(topology2);
58 // scenario3.executeOn(topology1);
59 // scenario1.executeOn(topology2);

Process 2 - S500
Terminal Process
1527@SCENARIO {Assignment1bMain} Process 2 has started commands [S500].
0 INFO {EPFD} Algorithm Start running
6 DEBUG {Application1b} Application STARTED!!
Application STARTED!!
37 INFO {Application1b} Sleeping 500 milliseconds...
539 INFO {Application1b} DONE ALL OPERATIONS
4009 DEBUG {EPFD} Sending hb to all alive to neighbours
4039 INFO {EPFD} Checking for dead neighbours
8012 DEBUG {EPFD} Sending hb to all alive to neighbours
8041 INFO {EPFD} Checking for dead neighbours
8046 INFO {EPFD} Suspected: 1@localhost:22031
8047 INFO {Application1b} PFD sais 1@localhost:22031 SUSPECTED.
12014 DEBUG {EPFD} Sending hb to all alive to neighbours
12048 INFO {EPFD} Checking for dead neighbours
16016 DEBUG {EPFD} Sending hb to all alive to neighbours
16049 INFO {EPFD} Checking for dead neighbours
20018 DEBUG {EPFD} Sending hb to all alive to neighbours
20049 INFO {EPFD} Checking for dead neighbours
24020 DEBUG {EPFD} Sending hb to all alive to neighbours
24054 INFO {EPFD} Checking for dead neighbours
24756 INFO {EPFD} Received hb 1@localhost:22031
28028 DEBUG {EPFD} Sending hb to all alive to neighbours
28056 INFO {EPFD} Checking for dead neighbours
28056 INFO {EPFD} Increasing period, false alarm! New period: 5000
28056 INFO {EPFD} Restored: 1@localhost:22031
28737 INFO {EPFD} Received hb 1@localhost:22031
32030 DEBUG {EPFD} Sending hb to all alive to neighbours
32740 INFO {EPFD} Received hb 1@localhost:22031
33057 INFO {EPFD} Checking for dead neighbours
36031 DEBUG {EPFD} Sending hb to all alive to neighbours

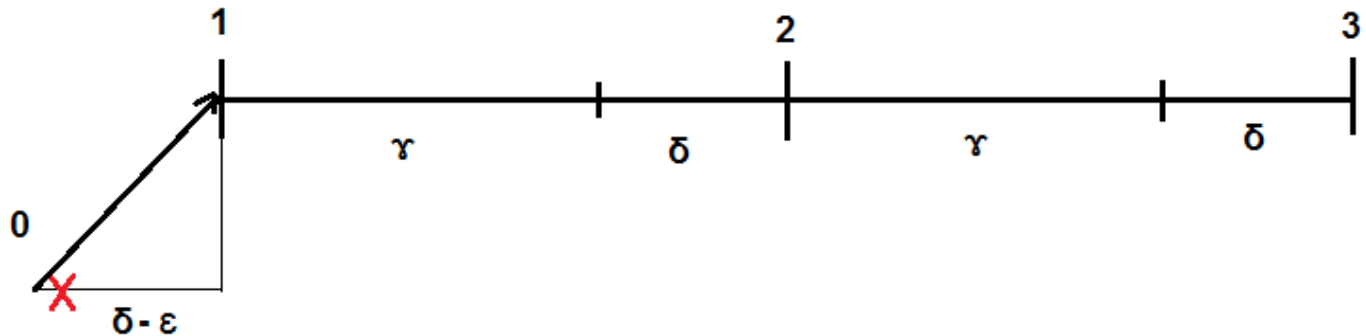
Input:
```

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.

QUESTION 1

The worst time is $2\gamma + 3\delta$. This situation illustrates following picture:



Let process p_1 sends heartbeat at 0 and immediately dies. The message is delivered in δ – maximum transmission delay. The time between 0 and 1 is $(\delta - \epsilon)$ where $\epsilon \rightarrow 0$ is very small. Process p_2 checks for heartbeats at 1, 2, 3. As heartbeat from p_1 was delivered at $1 + \epsilon$ process p_2 at 2 will not understand that p_1 is dead and understands it only at 3. It means after $(2\gamma + 3\delta - \epsilon)$, but as ϵ is very small we can consider that it is after $(2\gamma + 3\delta)$.

The best case is δ . If the node p_1 died less than δ before the check it means that the previous its heartbeat was less than $(\delta + \gamma)$ before and was received during that check round so the node will not be detected as dead during this check.

QUESTION 2

Can you improve the algorithm for PFD, such that it improves the worst case failure detection time?

Yes. Solutions:

More timers (more computations)

Keep a specific timer for every neighbor.

```
upon event < pp2pDeliver | Pi, [HEARTBEAT]> do
    cancel timer for Pi;
    startTimer (Pi);

upon event < Timeout | Pi > do
    detected := detected  $\vee$  {Pi};
    trigger < crash | Pi >;
```

By this way, we have a worst case of:

Pi sends a heartbeat and dies after $\epsilon \rightarrow 0$, that arrives to Pj after δ

Pj cancels the timer and starts a new one, that will trigger and detect Pi after $\gamma + \delta$

So, worst case of $\gamma + 2\delta$

More messages

A process can broadcast a heartbeat request to the other processes every γ and then wait for 2δ ($2 * \text{max network delay}$) for the response. If one process does not response in this time, it has crashed.

By this way, we have a worst case of:

Pi sends a heartbeat request to Pj (and all other nodes), request that arrives after $\epsilon \rightarrow 0$

Pj responses after $\zeta \rightarrow 0$

Pi checks for crashes after **2δ** , Pj looks ok

Pi sends a heartbeat request to Pj (and all other nodes) after γ

Pi detects the crash of Pj after **2δ**

So, worst case of **$2\delta + \gamma + 2\delta = \gamma + 4\delta$**

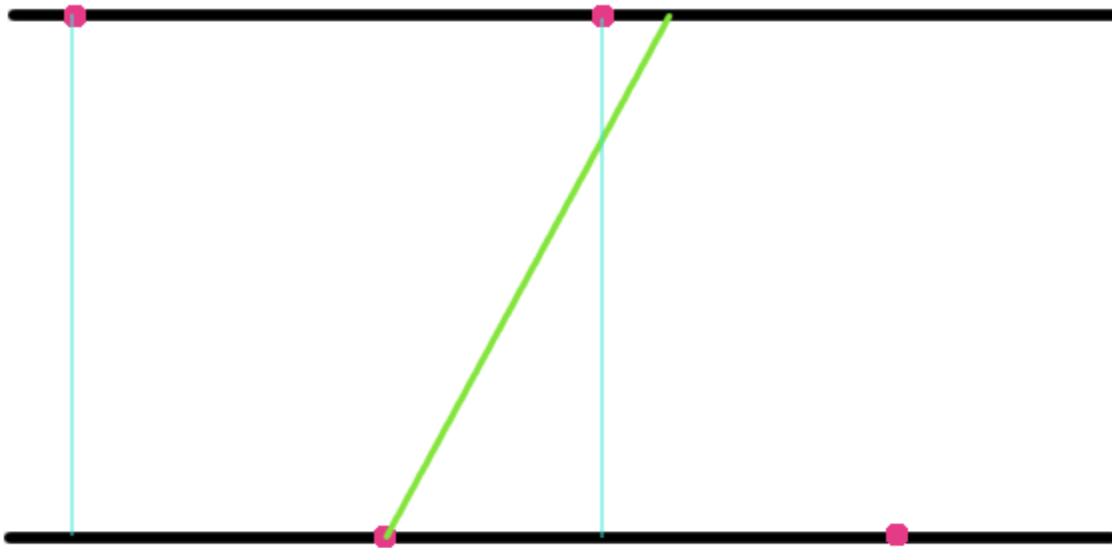
QUESTION 3

Can you improve the algorithm for PFD, such that it requires a single timer?

If we use a heuristic solution, we can answer yes. In the algorithm that we used for this homework, we can use **only one timer** with period δ and select a value for γ , such that $\gamma = v * \delta$, where $v \in I$. Then we can simply use a counter and two ifs to select when each part of the algorithm will execute.

Else, if we want to use an algorithm that is making use of only one timer, we would have to use the book's algorithm, which we will prove that is wrong in a specific case.

Assume that the TimeDelay that is used in the book's algorithm can be written as:



$\text{TimeDelay} = \gamma + \delta$, where δ is the maximum network delay. If we do not assume that the cycles in every node are synchronized, then there is the chance that the following execution could appear:

Which would obviously cause a false alarm.