Assignment 1

Advanced distributed system

2013

Mahboobeh Abdal Mahmood Abadi & Mehran Nasseri

KTH

2/6/2013

Contents

[1 Exercise 1 3](#_Toc347904229)

[1.1 Perfect failure detector 3](#_Toc347904230)

[1.2 Eventually perfect failure detector 5](#_Toc347904231)

[2 Exercise 2 7](#_Toc347904232)

[3 Exercise 3 8](#_Toc347904233)

[4 Exercise 4 9](#_Toc347904234)

# Exercise 1

## Perfect failure detector

The perfect failure detector (PFD) was examined with the following 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(10000, 0);

In this configuration, all nodes are fully connected to each other with ᵟ=4000ms, ϒ=1000ms, and link delay=10000ms, which set to be higher than the check interval (Gamma + Delta). As can be seen in figure 1 the implemented PFD reported failure.

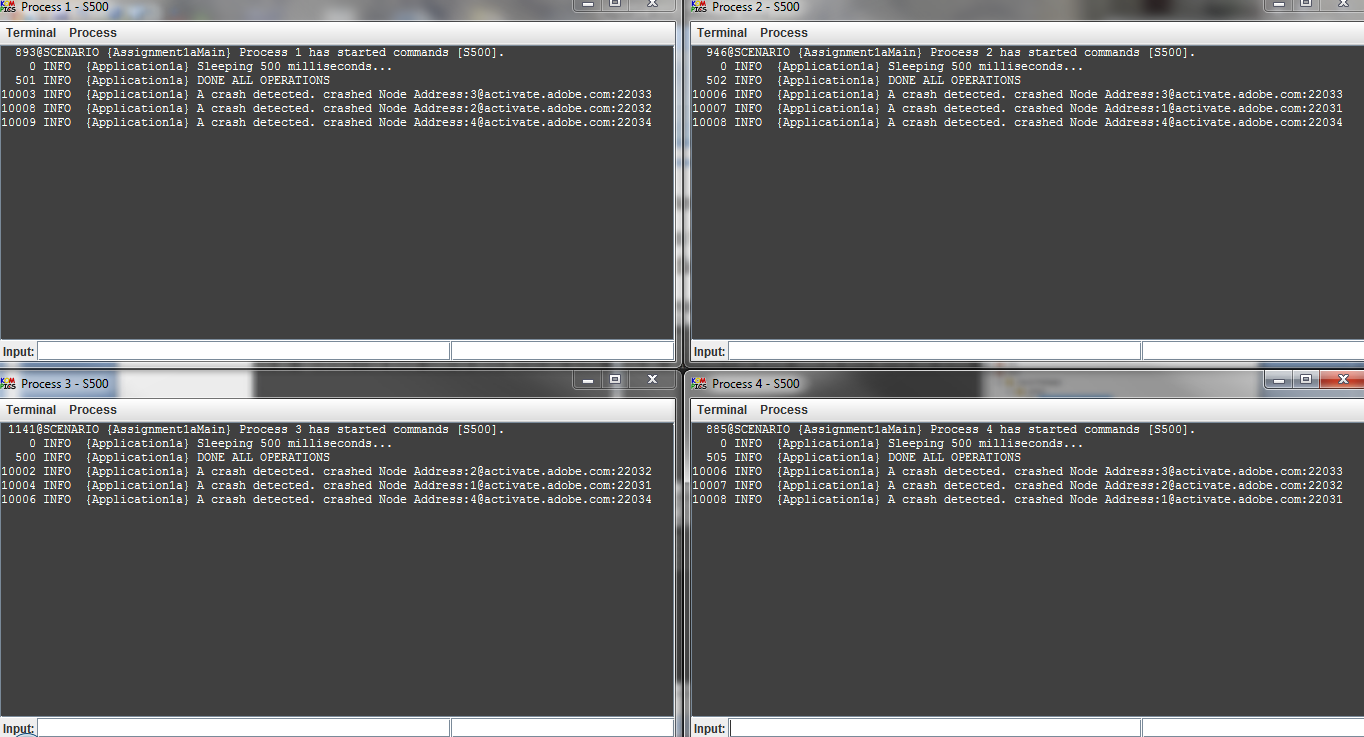


Figure 1

As it is shown in figure 1 all PFD reported all other nodes as crashed, while they were correct.

It must be considered that the PFD assumes a synchronous system and there is a bound on the time it takes for crashes to be detected (a priori a time period such that, whenever a process crashes, all correct processes suspect this process to have crashed after this period). In our case each HB message is guaranteed to arrive within a time period equal to the worst-case processing delay plus the worst-case message transmission delay (10000ms) plus ϒ=1000ms. Therefore, the check interval must be set to a value higher than the calculated upper bound. In other word in a synchronous system, process can measure the worst-case delay required to obtain a HB message and detect a crash when no such message arrives within the timeout period.

In the second experiment the perfect failure detector (PFD) was examined with the following 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(500, 0);

In this configuration, all nodes are fully connected to each other with ᵟ=4000ms, ϒ=1000ms. Then we killed one of the processes to by issuing ‘X’ command to check the completeness of the PFD as can be seen in figure 2 all other correct processes detect the crashed one.

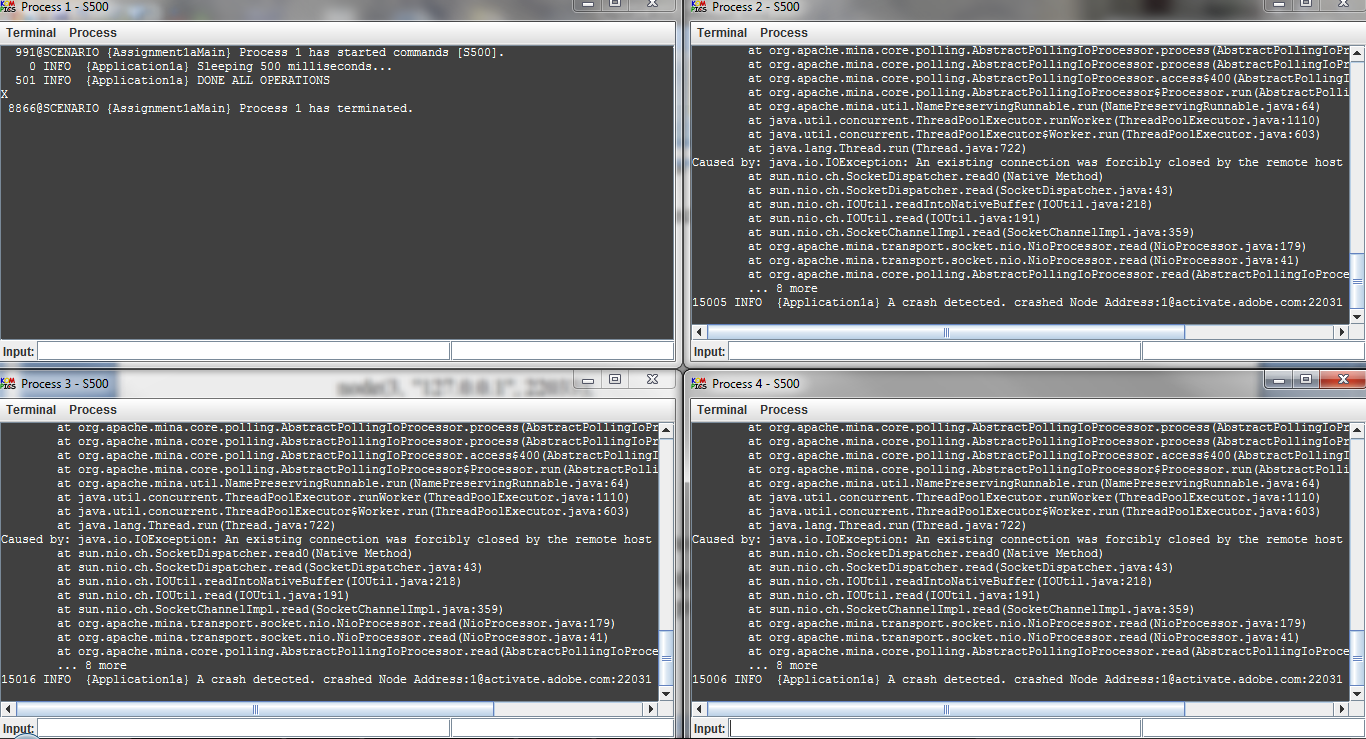


Figure 2

## Eventually perfect failure detector

The EPFD was examined by following 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(2000, 0.0);

In this configuration, all nodes are fully connected to each other with time delay set as 1000ms, and delta set as 500ms. As can be seen in figure 3 the implemented EPDF reported suspect/restore as it is intended.

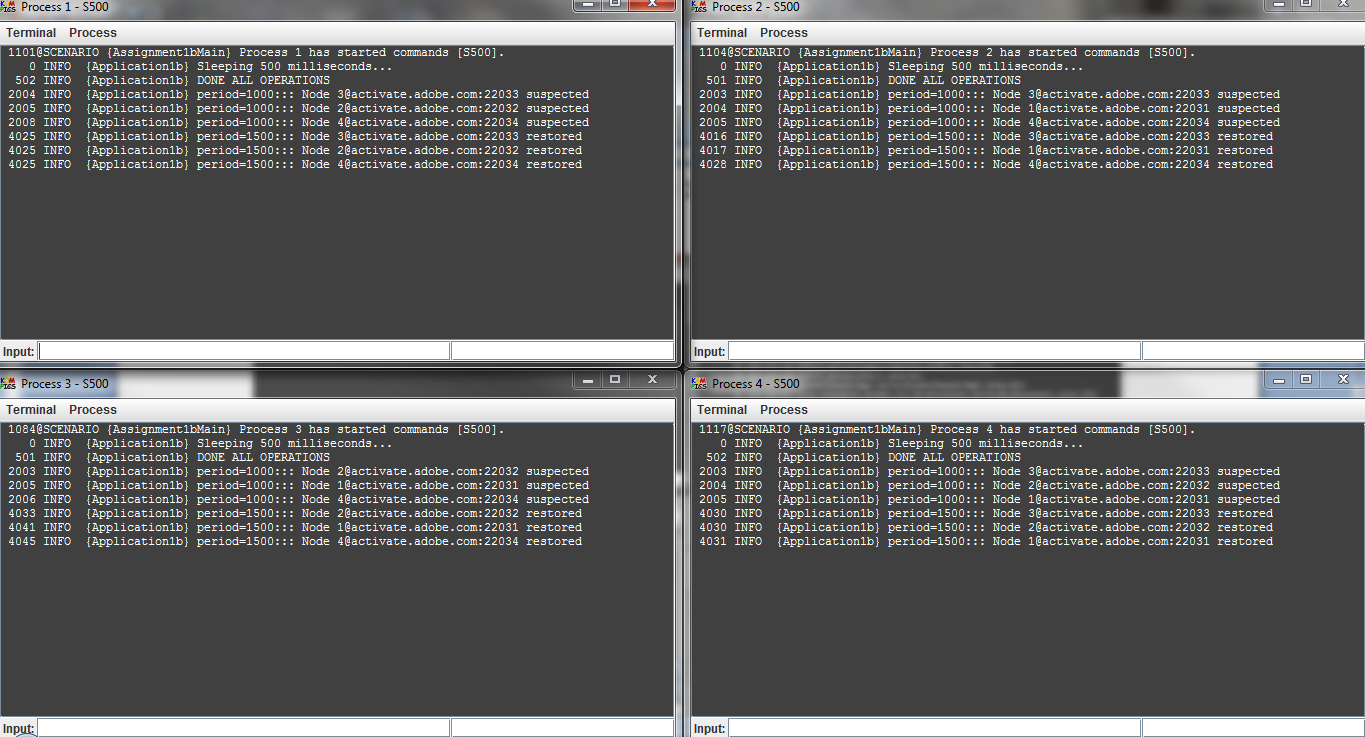


Figure 3

In figure 4 we show that eventually the killed process was permanently suspected by all correct processes.

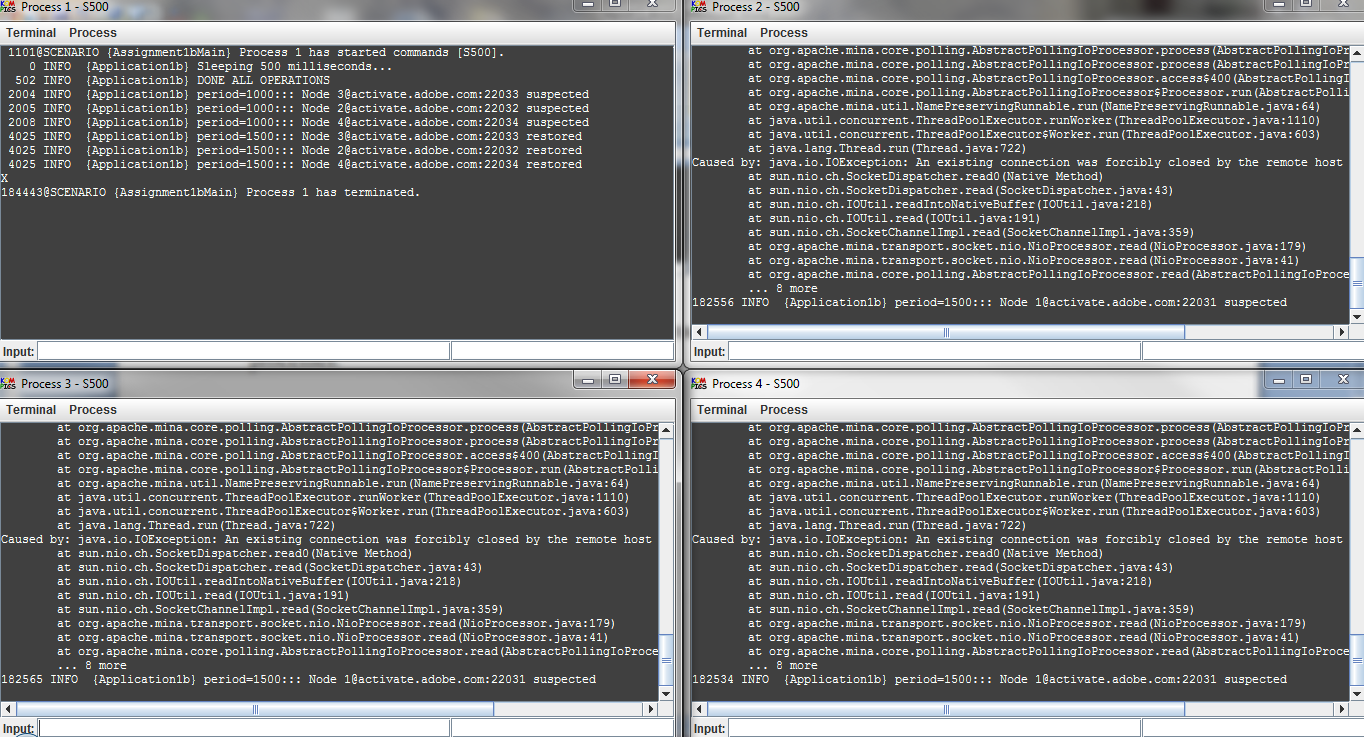


Figure 4

# Exercise 2

For EPFD, TimeDelay =1000 was initialized to a value smaller than link delay=4000 and following topology used to observe how EPFD is adjusted to accommodate larger transmission delays.

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(4000, 0.0);

As it is illustrated in figure 5 three of the EPDFs stabilized after increasing the checking period from 1000 to 2000 and the other one stabilized after increasing the checking period from 1000 to 1500ms.

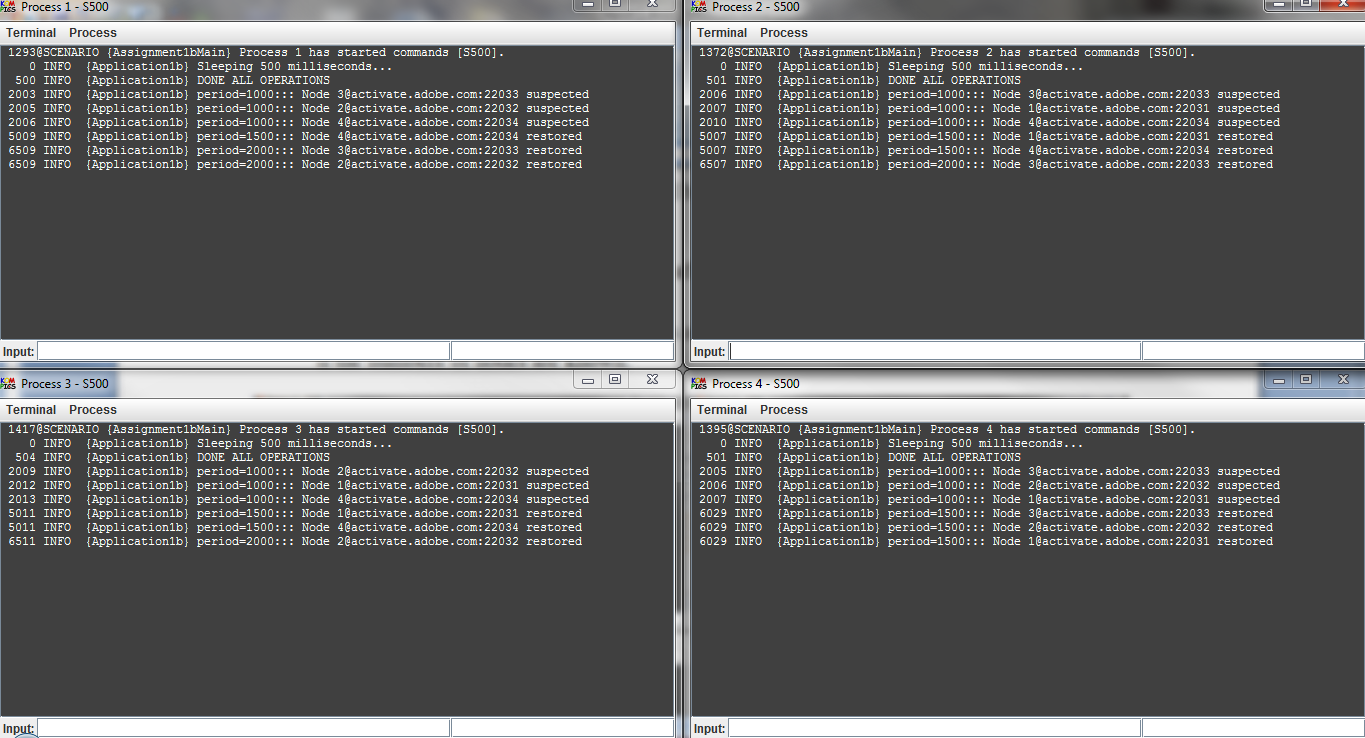


Figure 5

# Exercise 3

# 

For this exercise we used following 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(5000, 0.5);

For EPFD the time delay is set to 1000ms, delta is 500ms. In addition, the topology has 5000ms delay and 0.5 loss rate to simulate the how EPFD accommodates larger transmission delays when the messages are lost.

The observation, as can be seen in figures 6, proves that as the larger loose rate and/or delay link the faster period increases and EPFD can never be stabilized.

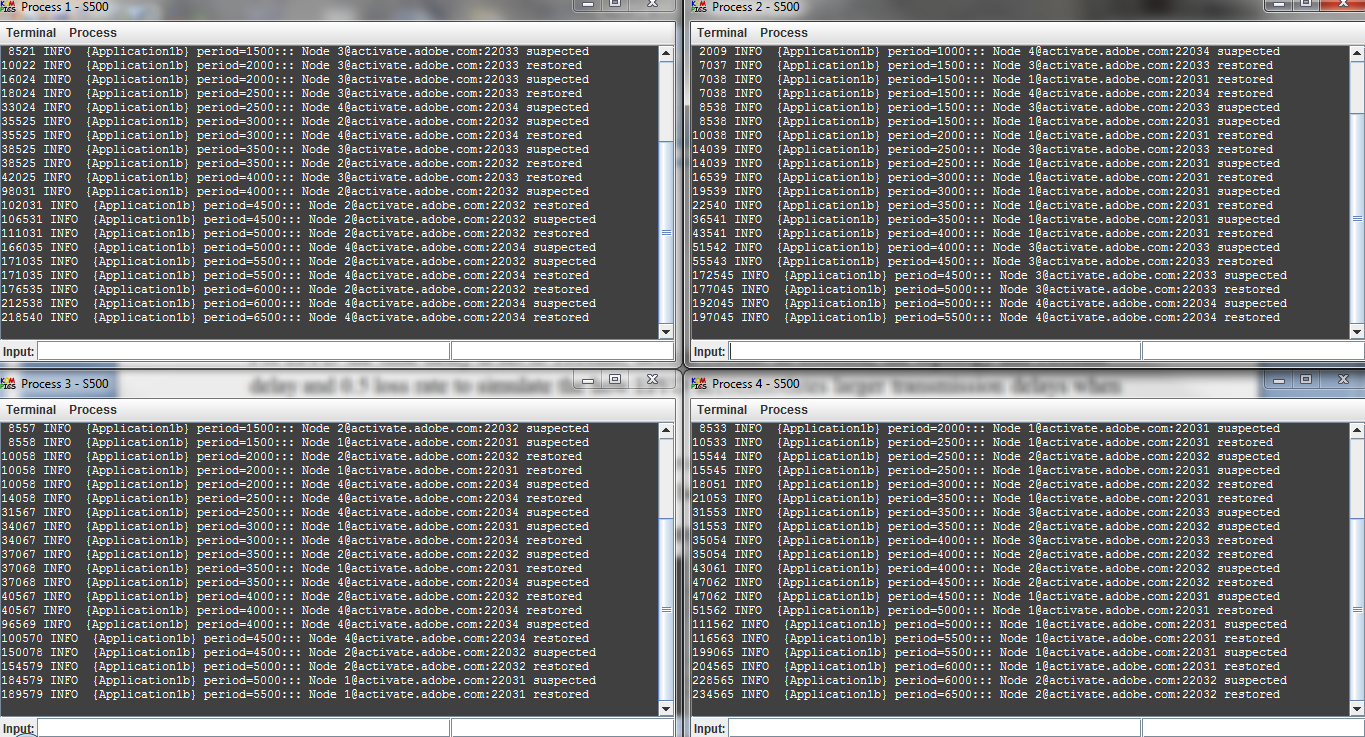


Figure 6: EPFD test with 0.5 loss rate

With fair-loss link the eventual strong accuracy property is not met, because correct process is suspected by other correct processes before it crashes. i.e. process was correct but the link dropped the message due to, for example buffer overflow or network congestion. This violates the safety property. If a process is detected at time before, it has crashed then the property is violated at time t.

# Exercise 4

# For this exercise we implement myRecovery () method, which is being called when a process recovers from a failure. To call this method we add command ‘R’ in the application component. However, this method just informs about the recovery.

The topology and scenario is executed as follow for this experiment:

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(500, 0.0);

command(1, "S2500:X").recover("R", 3500);

command(2, "S500");

command(3, "S500");

command(4, "S500");

For EPFD the time delay is set to 1000ms, delta is 500ms.

As can be seen in figure 7, when a process crashes the other nodes suspect that process and when the failed node recovers itself the other nodes recognize this and EPFDs report to their corresponding application that the node is alive again.

However, the problem is that the EPFD cannot distinguish between a correct node who suspected due to incorrect checking time interval (so it must adjust the checking period only in such case) and a crashed node who recovered again. Therefore, the algorithm cannot be used in a crash recovery model. The problem is that after each cycle of crash/restore the period increases by 500ms. Therefore, after a while, a faulty process can cause the checking period to increase continuously. Therefore, it will violate strong completeness property i.e. it cannot eventually suspect crashed node permanently.

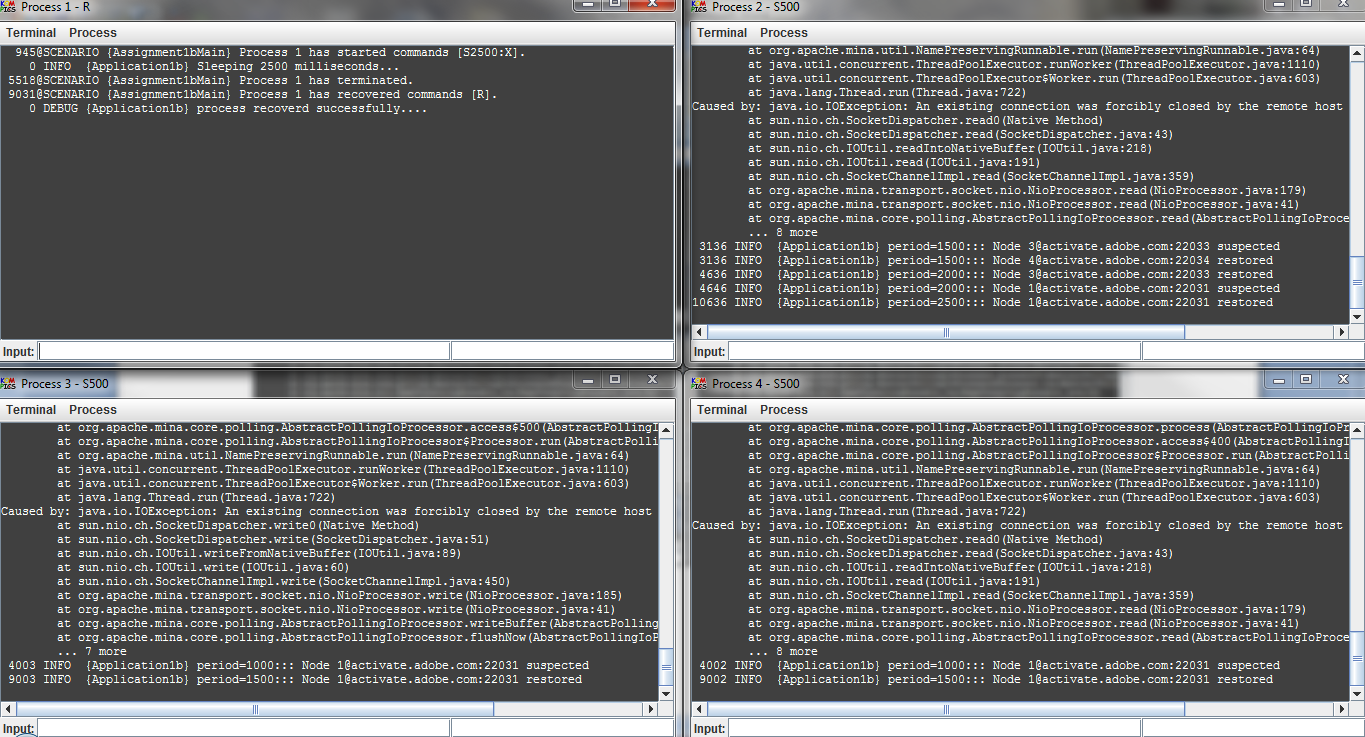


Figure 7