Software Engineering of Distributed Systems, KTH

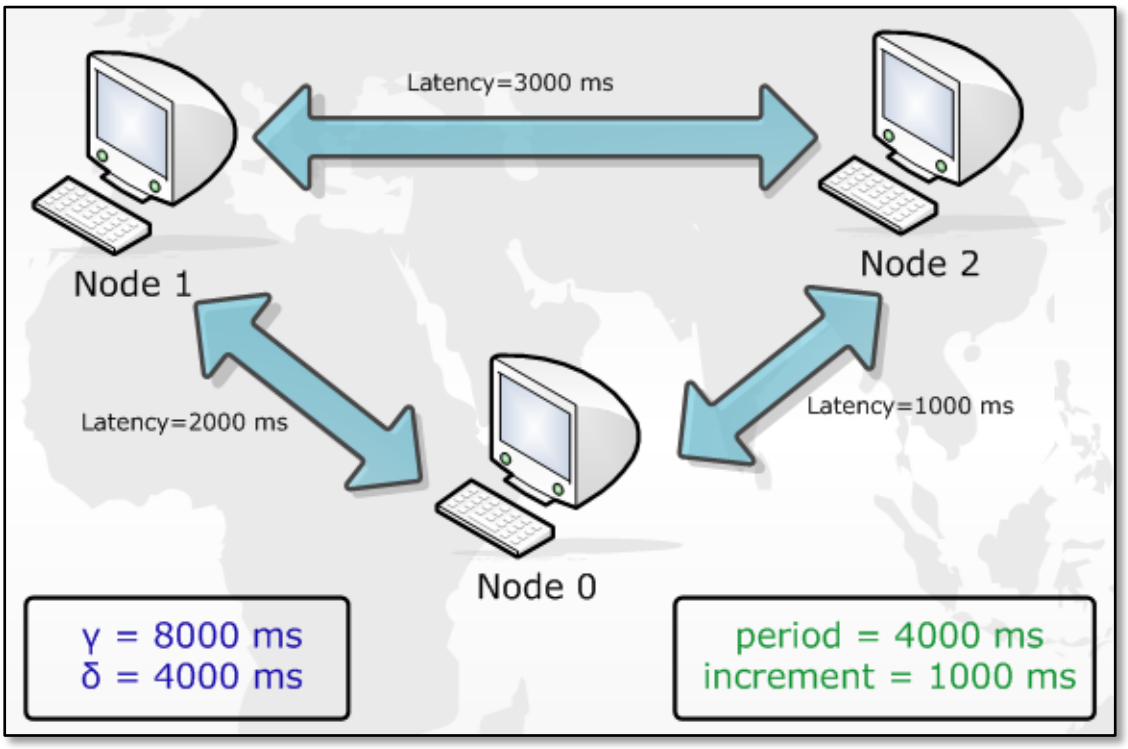
Distributed Systems Advanced Homework 2

Implementation of Perfect Failure Detector Component and Eventually Perfect Failure Detector Component

Shanbo Li and Sike Huang
2/10/2008

Exercise 1: Verify the completeness of the failure detectors

The topology is shown as following, three nodes are connected together, and each node has two links to the other two nodes respectively.



1.1 Experiment with Perfect Failure Detector Component

After system starts, every node works well. Each of them can receive heartbeat from others.
We manually crash node 2 and the follows are the output of node 1 and node 0:

After Manually Crash Node 2

Node 1:

Failure detected! Node 2 crash!
At 1202571867827
Duration since last heartbeat is 14859 ms
Gamma = 8000 ms
Delta = 4000 ms

After Manually Crash Node 2

Node 0:

Failure detected! Node 2 crash!
At 1202571869686
Duration since last heartbeat is 18703 ms
Gamma = 8000 ms
Delta = 4000 ms

From the log above, we can observe that each node is detecting others by receiving heartbeat. Node 1 detects the duration since he received the node 2's last heartbeat is 14859 ms, which is longer than γ + δ (12000 ms). So he detects that node 2 crashed. And it is similar for node 0, he also detects that node 2 crashed. The duration since node 0 gets node 2's last heartbeat is 18703 ms which is also longer than 12000 ms.

Shortly after node 2 crash, we manually crash node 1, and the follow log are from node 0's output

After Manually Crash Node 1

Node 0:

Failure detected! Node 1 crash!
At 1202571893686
Duration since last heartbeat is 15859 ms
Gamma = 8000 ms
Delta = 4000 ms

It shows that node 1 failed to send out heartbeat every γ time, and *node 0* has not received heartbeat from node 1 for 15859 ms. So it detects node 1 crashed.

1.2 Experiment with Eventually Perfect Failure Detector Component

After system starts, every node works well. Each of them can receive heartbeat from others. We manually crash *node 2* and the follows are the output of *node 1* and *node 0*:

After Manually Crash Node 2

Node 1:
Suspect "Node 2"!
At 1202575987905
Period = 5000 ms
Duration since last heartbeat is 8516 ms

After Manually Crash Node 2

Node 0:
Suspect "Node 2"!
At 1202575985514
Period = 5000 ms
Duration since last heartbeat is 8125 ms

From the log above, we can observe that each node is detecting others by receiving heartbeat. Node 1 suspect *node 2* because he has not received heartbeat from *node 2* for 8516 ms, which is longer than period which is 5000 ms. On node 0, it has not received heartbeat from *node 2* for 8125 ms which is also longer than period, so it will also suspect *node 2*.

Then we manually crash *node 1*, see what happens on *node 0*.

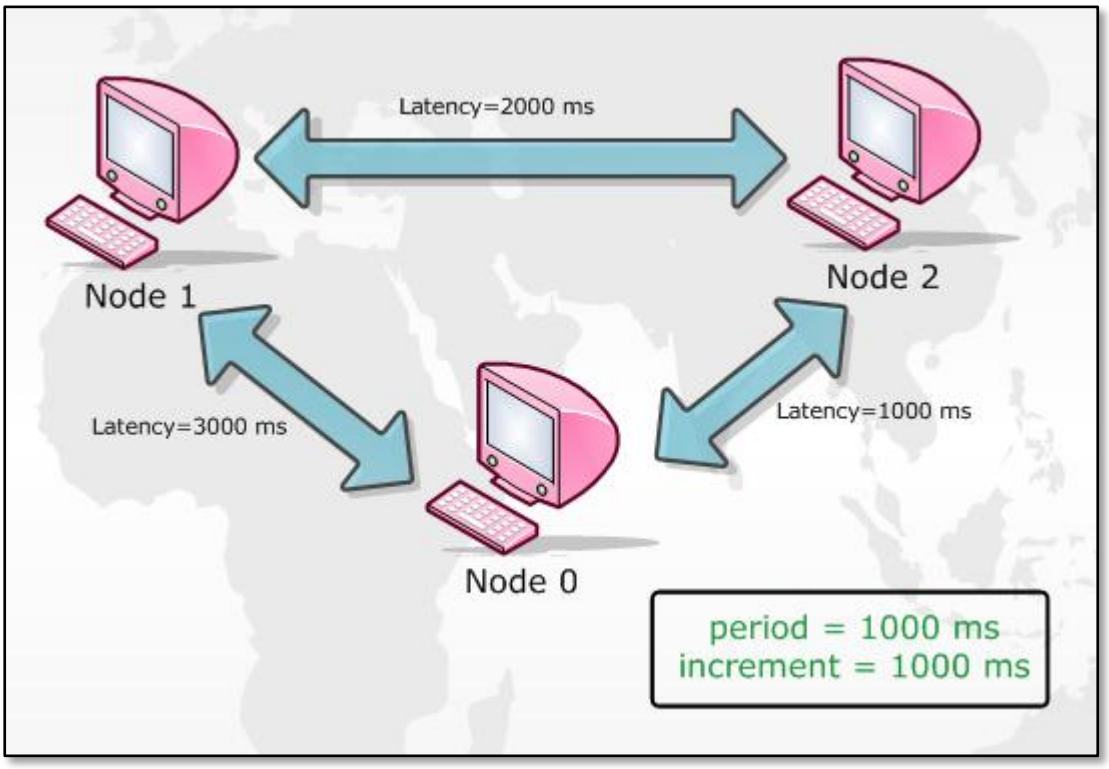
After Manually Crash Node 2

Node 0:
Suspect "Node 1"!
At 1202575995514
Period = 5000 ms
Duration since last heartbeat is 5609 ms

Node 0 suspect *node 1* for it has not received heartbeat from node 0 for 5609 ms. So *node 0* add *node 1* to its suspect list.

Exercise 2. Observation of TimeDelay Adjustment in EPFD

The topology used in our observation is illustrated in below, there are three nodes connected to each other with significant delay on the links, the initial period, or so called time delay of the eventually perfect failure detector is set to be 1000ms, smaller than link delays, and it will increase 1000ms each time.



First let's look at the events happened in each node.

Node 0

Node 0: startup at 1202660295963

Suspect "Node 1"!, at 1202660297986
Period = 1000 ms

Suspect "Node 2"!, at 1202660297986
Period = 1000 ms

Restore "Node 1"!, at 1202660302002
Period = 2000 ms

Restore "Node 2"!, at 1202660302002

Period = 2000 ms

Suspect "Node 1"!, at 1202660308001
Period = 2000 ms

Restore "Node 1"!, at 1202660310001
Period = 3000 ms

Node 1

Node 1: startup at 1202660297873

Suspect "Node 0"!, at 1202660299879
Period = 1000 ms

Suspect "Node 2"!, at 1202660299880
Period = 1000 ms

Restore "Node 0"!, at 1202660300879
Period = 2000 ms

Restore "Node 2"!, at 1202660302881
Period = 3000 ms

Node 2

Node 2: startup at 1202660299090

Suspect "Node 0"!, at 1202660304099
Period = 1000 ms

Suspect "Node 1"!, at 1202660304100
Period = 1000 ms

Restore "Node 0"!, at 1202660305100
Period = 2000 ms

Restore "Node 1"!, at 1202660305100
Period = 2000 ms

Suspect "Node 1"!, at 1202660307101
Period = 2000 ms

Restore "Node 1"!, at 1202660309100
Period = 3000 ms

The result is in expectation, we can observe that each node adjusts its time delay, and finally they all reach the consensus that period should be 3000ms in order to accommodate larger transmission delay (which is 3000ms).

If we regard the time node 0 starts as 0ms, then node 1 starts at 1910ms, and node 2 starts at 3127ms.
After node 0 starts, it waits 1000ms to broadcast heartbeat for the first time, assuming all nodes are alive without any suspicion:
Node 0: send heartbeat to Node 1 at 1202660296985 [at 1022ms, will reach node 1 at 4022ms]
Node 0: send heartbeat to Node 2 at 1202660296985 [at 1022ms, will reach node 2 at 2022ms]
At 2000ms, node 0 still haven't received any heartbeat, since node 1 just starts up, and node 2 haven't started yet, so node 0 puts node 1 and node 2 into suspicion, and broadcast again.
Node 0: send heartbeat to Node 1 at 1202660297985 [at 2022ms, will reach node 1 at 5022ms]
Node 0: send heartbeat to Node 2 at 1202660297985 [at 2022ms, will reach node 2 at 3022ms]
Suspect "Node 1"!, at 1202660297986 [at 2023ms]
Suspect "Node 2"!, at 1202660297986 [at 2023ms]
At 3000ms, node 0 broadcasts again:
Node 0: send heartbeat to Node 1 at 1202660298999 [at 3036ms]
Node 0: send heartbeat to Node 2 at 1202660298999 [at 3036ms]

Meanwhile at 2000ms (1910ms, to be accurate), node 1 starts up, 1000ms later node 1 broadcasts heartbeat:
Node 1: send heartbeat to Node 0 at 1202660298879 [at 2916ms, will reach node 0 at 5916ms]
Node 1: send heartbeat to Node 2 at 1202660298879 [at 2916ms, will reach node 2 at 3916ms]
At 3916ms, node 1 broadcasts again, and suspect node 0 and node 2:
Node 1: send heartbeat to Node 0 at 1202660299879 [at 3916ms]
Node 1: send heartbeat to Node 2 at 1202660299879 [at 3816ms]
Suspect "Node 0"!, at 1202660299879 [at 3916ms]
Suspect "Node 2"!, at 1202660299880 [at 3917ms]

At the same time, while at 3127ms, node 2 starts up, it first receives a heartbeat from node 0:
Node 2: Get a Heartbeat from "Node 0" at 1202660300006 [at 4043ms], the message was send at 1202660298999 [at 3036ms]
And node 2 broadcasts at 4127ms:
Node 2: send heartbeat to Node 0 at 1202660300093 [at 4130ms]
Node 2: send heartbeat to Node 1 at 1202660300093 [at 4130ms]
Later on node 2 receives a heartbeat from node 2:
Node 2: Get a Heartbeat from "Node 1" at 1202660300892 [at 4929ms], the message was send at 1202660298879 [at 2916ms]

Come back to node 0, it receives a heartbeat from node 2 and a heartbeat from node 1 as well:
Node 0: Get a Heartbeat from "Node 2" at 1202660301126 [at 5163ms], the message was send at 1202660300093 [at 4130ms]
Node 0: Get a Heartbeat from "Node 1" at 1202660301882 [at 5919ms], the message was send

at 1202660298879 [at 2916ms]

When node 0 times out again at 6000ms, it restores node 1 and node 2, and changes its period to 2000ms:

Restore "Node 1"!, at 1202660302002 [at 6039ms]

Restore "Node 2"!, at 1202660302002 [at 6039ms]

Period = 2000 ms

Come back to node 1, it receives a heartbeat from node 0 at 4059ms:

Node 1: Get a Heartbeat from "Node 0" at 1202660300022 [at 4059ms], the message was send at 1202660296985 [at 1022ms]

When node 1 times out at 4127ms, it restores node 0 and increases its period to 2000ms:

Restore "Node 0"!, at 1202660300879 [at 4916ms]

Period = 2000 ms

Later on, node 1 receives a heartbeat from node 2 at 6136ms:

Node 1: Get a Heartbeat from "Node 2" at 1202660302099 [at 6136ms], the message was send at 1202660300093 [at 4130ms]

When node 1 times out at 6127ms, it restores node 2 and increases its period to 3000ms:

Restore "Node 2"!, at 1202660302881 [at 6918ms]

Period = 3000 ms

So far node 1 has reached the consensus.

Next, let's see how node 2 reaches the consensus.

Node 2 suspects node 0 and node 1:

Suspect "Node 0"!, at 1202660304099 [at 8136ms]

Suspect "Node 1"!, at 1202660304100 [at 8137ms]

Since the last message received from node 1 is:

Node 2: Get a Heartbeat from "Node 1" at 1202660302883 [at 6920ms], the message was send at 1202660300879 [at 4916ms]

8137ms – 6920ms > 1000ms

The last message received from node 0 is:

Node 2: Get a Heartbeat from "Node 0" at 1202660303005 [at 7042ms], the message was send at 1202660302001 [6038ms]

8136ms – 7042ms > 1000ms

Later on, node 2 receives heartbeat again from node 0 and node 1:

Node 2: Get a Heartbeat from "Node 1" at 1202660304882 [at 8920ms], the message was send at 1202660302880 [at 6917ms]

Node 2: Get a Heartbeat from "Node 0" at 1202660305004 [at 9041ms], the message was send at 1202660304001 [at 8038ms]

When node 2 times out again at 9127ms, it restores node 0 and node 1 and increases period to 2000ms:

Restore "Node 0"!, at 1202660305100 [at 9137ms]

Restore "Node 1"!, at 1202660305100 [at 9137ms]

Period = 2000 ms

After the restoration, within two seconds, there isn't any heartbeat from node 1:

Suspect "Node 1"!, at 1202660307101 [at 11138ms]

Period = 2000 ms

The heartbeat from node 1 comes at:

Node 2: Get a Heartbeat from "Node 1" at 1202660307885 [at 11922ms], the message was send at 1202660305880 [at 9917ms]

When node 2 times out at 13127ms:

Restore "Node 1"!, at 1202660309100 [at 13137ms]

Period = 3000 ms

Till now node 2 has reached the consensus.

Finally back to node 0, after it restores node 1 and node 2, it suspects node 1 again:

Suspect "Node 1"!, at 1202660308001 [at 12038ms]

Period = 2000 ms

Because the last heartbeat received:

Node 0: Get a Heartbeat from "Node 1" at 1202660305882 [at 9919ms], the message was send at 1202660302879 [at 6916ms]

12038 ms - 9919ms > 2000ms

Later on, node 0 receives a heartbeat from node 1 again, so restore it at 14000ms eventually:

Node 0: Get a Heartbeat from "Node 1" at 1202660308883 [at 12920ms], the message was send at 1202660305880 [at 9917ms]

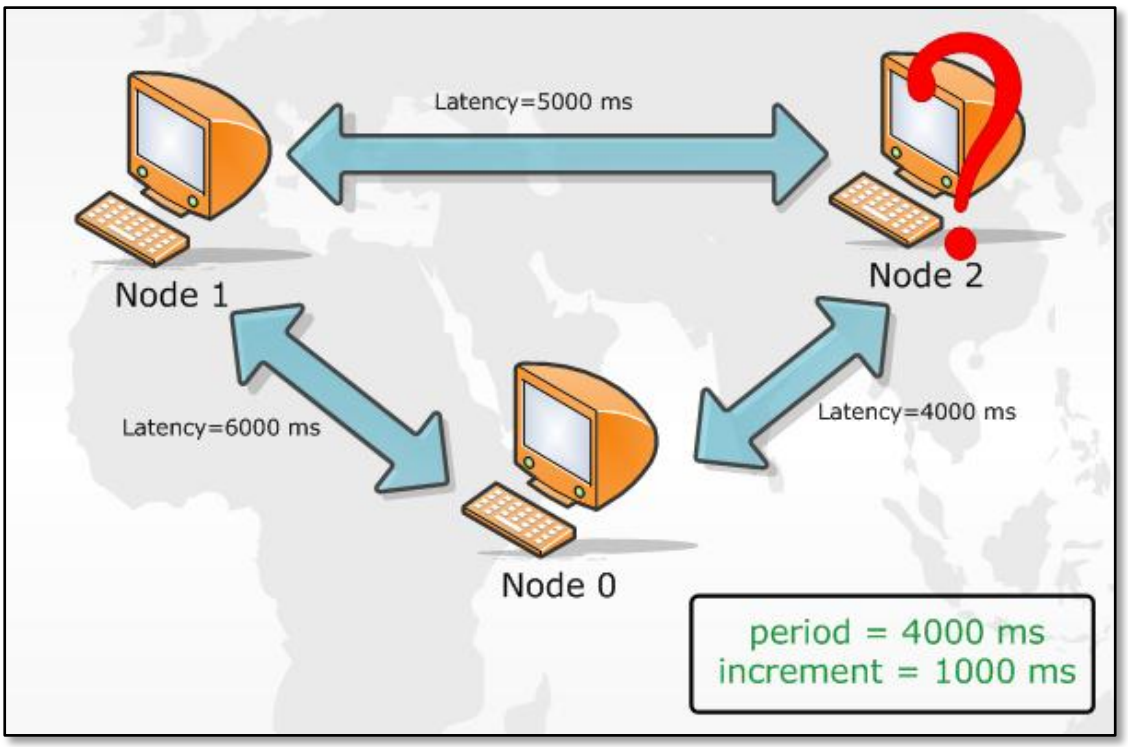
Restore "Node 1"!, at 1202660310001 [at 14038ms]

Period = 3000 ms

Lastly node 0 has reached the consensus.

Exercise 3. EPFD in Crash-Recovery Model

In exercise, the topology is similar to previous ones, whereas the question mark indicates that node 2 will be killed and restarted to observe what will happen.



After three nodes reach the consensus that the period is 5000ms, we stop node 2, and the following shows what happens in the other two nodes.

Node 0

Suspect "Node 2"!, at 1202679962534

Period = 5000 ms

Since the last heartbeat from node 2 is:

Node 0: Get a Heartbeat from "Node 2" at 1202679956151, the message was send at 1202679952149

1202679962534 – 1202679956151 = 6383ms > 5000ms

Node 1

Suspect "Node 2"!, at 1202679963523

Period = 5000 ms

Since the last heartbeat from node 2 is:

Node 1: Get a Heartbeat from "Node 2" at 1202679957151, the message was send at 1202679952150

1202679963523 – 1202679956151 = 7372ms > 5000ms

Later on, we restart node 2 again, and the other nodes receive the heartbeat again from node 2, so they restore node 2.

Node 0

Node 0: Get a Heartbeat from "Node 2" at 1202679984652, the message was send at 1202679980644

Restore "Node 2"!, at 1202679987534

Period = 6000 ms

Node 1

Node 1: Get a Heartbeat from "Node 2" at 1202679985646, the message was send at 1202679980644

Restore "Node 2"!, at 1202679988524

Period = 6000 ms

As long as node 0 and node 1 receive heartbeat from node 2 once more, they simply sweep away the suspicion made before, and consider node 2 is alive as if node 2 or the network was heavily loaded, rather than the fact that it was crashed.

We can solve this problem by assign a unique number to each node during it starts up. When a node broadcasts, it piggybacks this number in the heartbeat message, if this node crashes and recoveries, the number will be reassigned to a new one, so other nodes will see a different number in the heartbeat message, therefore they can tell this node is no longer the one before.