Software Engineering of Distributed Systems, KTH

Distributed Systems Advanced Homework 1

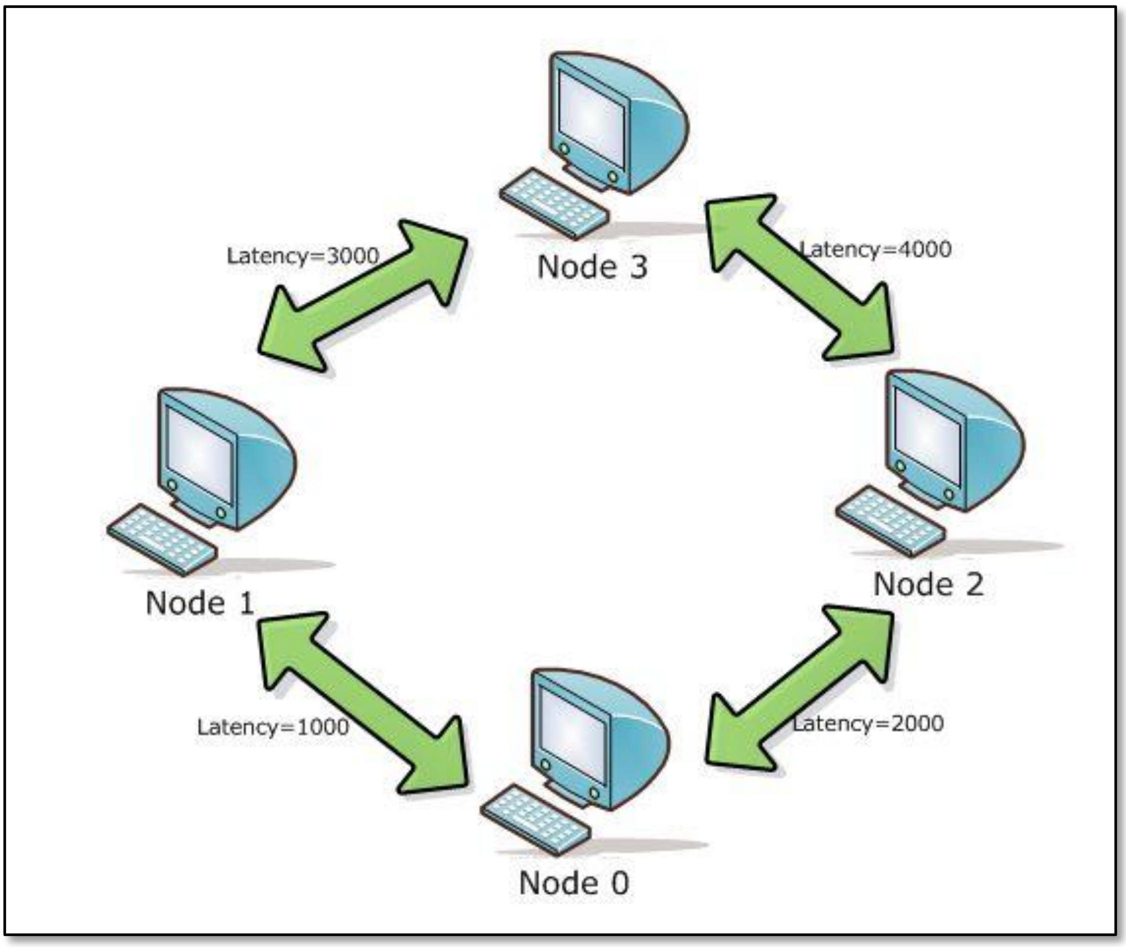
TBN KickStart

Sike Huang and Shanbo Li

2/3/2008

Question 1: Bidirectional links and no loss

The first topology we build upon our implementation is illustrated in the figure below.



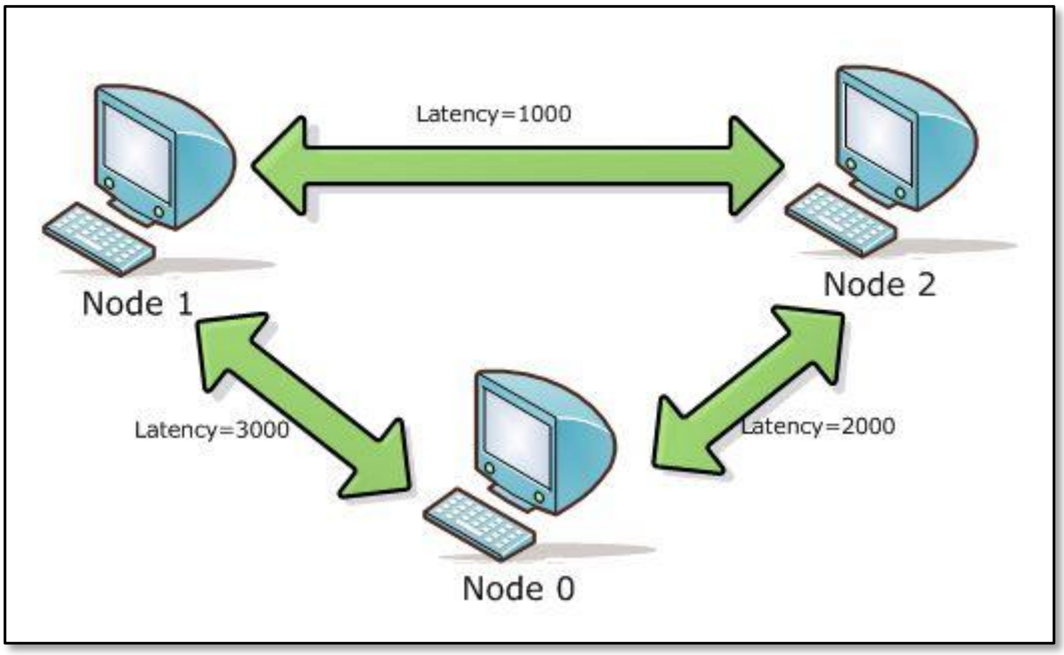
Four nodes are connected in the way to form a ring network. The links are with no loss rate and has various latencies in milliseconds.
We obtain the following results by printing the current time when handling the FloodDoneEvent in the application component on each node:

```
Node 0 -> Start flooding message: hello at 1202070369000
Node 0 -> Done flooding message: "hello" at 1202070373053
Node 1 -> Done flooding message: "hello" at 1202070376138
Node 2 -> Done flooding message: "hello" at 1202070377130
Node 3 -> Done flooding message: "hello" at 1202070375053
Node 0 takes 4053ms
Node 1 takes 7138ms
Node 2 takes 8130ms
Node 3 takes 6053ms
```

One can see that *node 0* finishes first, followed by *node 3*, *node 1* and *node 2* are among the last

respectively, the order in which the food is done at each process is within our expectation.
Node 0 initializes the flooding by sending the flood message to *node 1* and *node 2*, when *node 1* sees the message it floods forward to *node 0* and *node 3*, so does *node 2*. As soon as *node 0* receives message back from *node 1* and *node 2*, it finishes the job. The time spent in the transmission for *node 0* to complete is 2000*2 = 4000ms.
Node 3 sees the message from node 1 first (and flood it to *node 1* and *node 2*), followed by the same message from *node 2* after two seconds, then it is done, the time spent in the transmission for *node 3* to complete is 2000+4000 = 6000ms.
Then *node 1* gets the message from *node 3* one second before *node 2* does, and they are done. The time spent in the transmission for *node 1* to complete is 1000+3000*2 = 7000ms, whilst for *node 2* is 2000+(3000-(2000-1000))+4000 = 8000ms.
Notice that time calculated above counts from the start of the initial flooding by *node 0*, and the sequence of the time matches the order of termination.
If we ignore processing time on each node, than the values we obtained and calculated match very well.

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



The result is as follows:

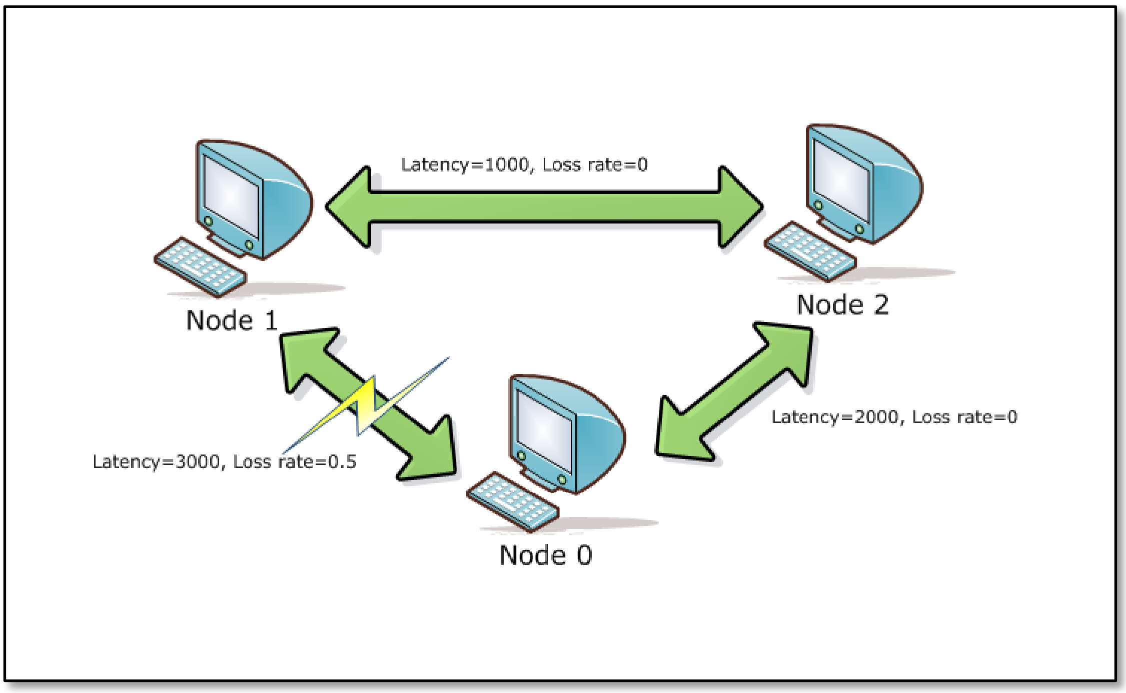
```
Node 0 -> Start flooding message: hello at 1202072206899
Node 0 -> Done flooding message: "hello" at 1202072212950
Node 1 -> Done flooding message: "hello" at 1202072210000
Node 2 -> Done flooding message: "hello" at 1202072210959
Node 0 takes 6051ms
Node 1 takes 3101ms
Node 2 takes 4060ms
```

The order can be sorted as *node 1*, *node 2*, *node 0*, from top to down, earliest to latest.

The initial message flooded by *node 0* arrives at *node 2* 1000ms earlier than at *node 1*, as soon as *node 2* sees the message it floods the message to *node 0* and *node 1*.
Meanwhile, *node 1* gets the message and floods it to *node 2* and *node 0*.
Because the latency of link between *node 1* and *node 2* is much smaller than the links between *node 2* and *node 0* as well as *node 1* and *node 0*, *node 1* and *node 2* will end their job much earlier than *node 0*.
The time spent in the transmission for *node 1* to complete is 3000ms approximately.
The time spent in the transmission for *node 2* to complete is 2000 + 1000*2 = 4000ms.
The time spent in the transmission for *node 0* to complete is 3000*2 = 6000ms

Question 2: Bidirectional links and loss rate = 50%

The topology with loss rate is shown as following, three nodes are connected together, and each node has two links to the other two nodes respectively, and all links have loss rate which is 50%.


Latency=1000, Loss rate=0
Node 1
Node 2
Latency=3000, Loss rate=0.5
Latency=2000, Loss rate=0
Node 0

The results are as follows:

Scenario A:

Node 0 -> Start flooding message: "A" at 1202071558437

Node 0 -> Done flooding message: "A" at 1202071564765

Node 1 -> No flooding Done

Node 2 -> Done flooding message: "A" at 1202071562749

Package Lost:

Node 0 -> 3313 INFO [DropComponentFactory:DropComponent] {DropComponent} Message FloodMessage to 1 dropped

Scenario B:
Node 0 -> Start flooding message: "B" at 1202071978624
Node 0 -> No flooding Done
Node 1 -> Done flooding message: "B" at 1202071981843
Node 2 -> Done flooding message: "B" at 1202071982734
Package Lost: **Node 1 -> 9860 INFO [DropComponentFactory:DropComponent] {DropComponent} Message FloodMessage to 0 dropped**

Scenario C:
Node 0 -> Start flooding message: "C" at 1202072847890
Node 0 -> Done flooding message: "C" at 1202072854015
Node 1 -> Done flooding message: "C" at 1202072851109
Node 2 -> Done flooding message: "C" at 1202072852031
Package Lost: **No Package Lost!**

In this question we focus on the loss rate. From the topology picture, we know that there is a loss rate 50% between *node 0* and *node 1*. In other words, message from *node 0* to *node 1* may drop as well as the opposite direction.

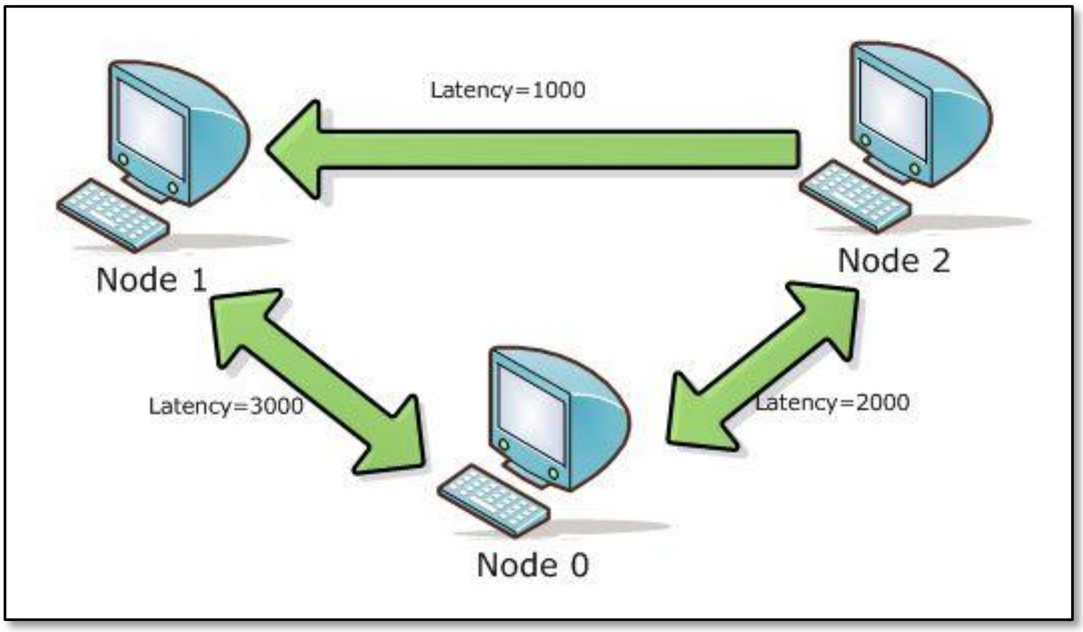
In Scenario A, *node 0* loses a package to *node 1*, since each message will be send only once by each node, *node 1* will never get message “A” from *node 1*. So *node 1* will never finish its flood. Although *node 0* failed to send message to *node 1*, *node 1* can still get the same message from *node 2*. And *node 1* will send the message to both *node 2* and *node 0*. Finally *node 0* gets a message from *node 1* and finishes flooding.

In Scenario B, when *node 1* sends the message to *node 0*, the message drops. And *node 0* will never get the message from *node 1*. So *node 0* cannot finish flooding.

In Scenario C, no package lost. So all node finish flooding.

Question 3: Unidirectional links

The new topology is modified from second one from question 1. The only change is the link between *node 1* and *node 2*, now it is unidirectional, starting from *node 2*, ending at *node 1*.



We run the test, and finds that *node 0* and *node 1* raises the flood done event, but not *node 2*.

Node 0 has been though the steps:

```
Start flooding message: question3 at 1202073238119
Raising FloodInitEvent
Node 0: send message "question3" to Node 1
Node 0: send message "question3" to Node 2
Node 0: get message "question3" from Node 2
Node 0: get message "question3" from Node 1
Node 0: Raising FloodDoneEvent
Done flooding message: "question3" at 1202073244178
```

Node 1 has been though the steps:

```
Node 1: get message "question3" from Node 0
Start flooding message: question3 at 1202073241161
Node 1: send message "question3" to Node 0
[Missing -> send message to Node 2], because there is no link going from node 1 to node 2
Node 1: get message "question3" from Node 2
Node 1: Raising FloodDoneEvent
Done flooding message: "question3" at 1202073241162
```

Node 2 has been though the steps:

```
Node 2: get message "question3" from Node 0
Start flooding message: question3 at 1202073240202
Node 2: send message "question3" to Node 1
Node 2: send message "question3" to Node 0
[Waiting for message from Node 1…]
```

It is obviously to conclude that *node 2* will never raise flood done event, due to the lack of message from *node 1*.