Distributed Systems Advanced Course - ID2203 Homework 01

Students:

Rahim Delaviz, rahimda@kth.se

Tryggvi Larusson, larusson@kth.se

Exercise 1

To check the impact of changing latency on the Flooding Message , We have done two runs on two topologies with same structure but different latency. Our topologies are as below:

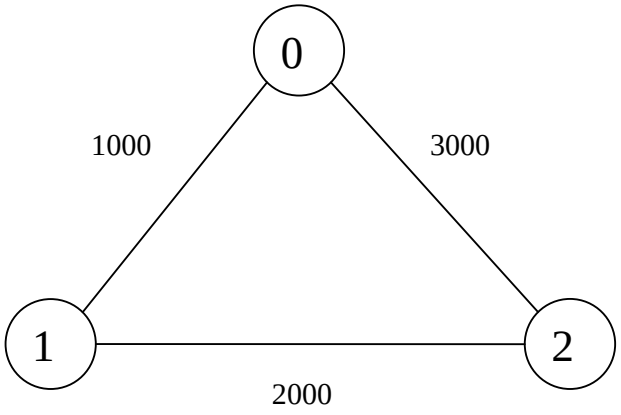

Figure1: Topology of run 1

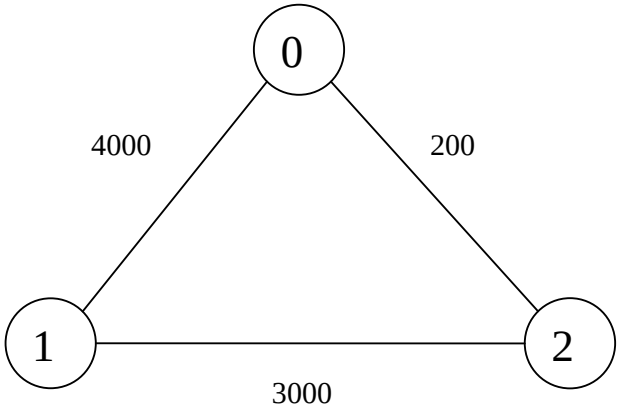

Figure2: Topology of run number 2

the numbers show the latency of each edge. The result of running algorithm with above topology has been shown below:

	Node	Ending Time	Difference	Source Node	Destination Node	Latency	DirectionType
Run 1	0	1202152059265		0	1	1000	Bidirectional
	1	1202152060437	1172	1	2	2000	Bidirectional
	2	1202152062359	1922	2	0	3000	Bidirectional
Run 2	0	1202152309218		0	1	1000	Bidirectional
	1	1202152310390	1172	1	2	2000	Bidirectional
	2	1202152312296	1906	2	0	3000	Bidirectional
Result	As it is obvious the node number 2 has sent FloodDone message after other nodes , the reason is that the sum of latencies associated with this node (2000+3000) is greater that than other nodes. Also the sum of associated latencies of node 1 (1000+2000) is less than node number 0 (1000+3000) but node 0 has finished earlier than node 1. The reason for this unusual output can be processing time and the difference between latencies is not so high that covers processing time. Because node 0 sends the first message so it is acceptable that with low latency difference rate, it finishes before others.						
Run 1	2	1202152494609		0	1	4000	Bidirectional
	0	1202152494843	234	1	2	3000	Bidirectional
	1	1202152497703	2860	2	0	200	Bidirectional
Run 2	0	1202152742734		0	1	4000	Bidirectional
	2	1202152743093	359	1	2	3000	Bidirectional
	1	1202152746203	3110	2	0	200	Bidirectional
Result	In both runs node 1 has sent FloodDone message after all other nodes. The reason is obvious because the sum up of incoming edges latencies(3000+ 4000) is greater than others. In the run 1 node 2 has finished before node 0 and in the run 2 vice versa. The reason is same as previous latency rates run. Node 2 finished before node 1 because sum of its edges latency (3000+200) is less than others. Node 0 has finished before other nodes because it has send the first message and may be has received all messages before others. And it is acceptable that in different runs one time node 0 finishes before node 2 and other time node 2 but deterministically node 1 will be finished after other nodes.						

Exercise 2

We set up another different topology, this time with link loss on all the links, which changed the results considerably.
We ran this twice and we see that in the second time there is a case where the FloodDoneEvent is never received at nodes 1 and 2. This is because the loss makes some messages not be delivered at all which of course makes the FloodComponent not detect all incoming FloodMessages so in some cases FloodDoneEvent may never be raised.

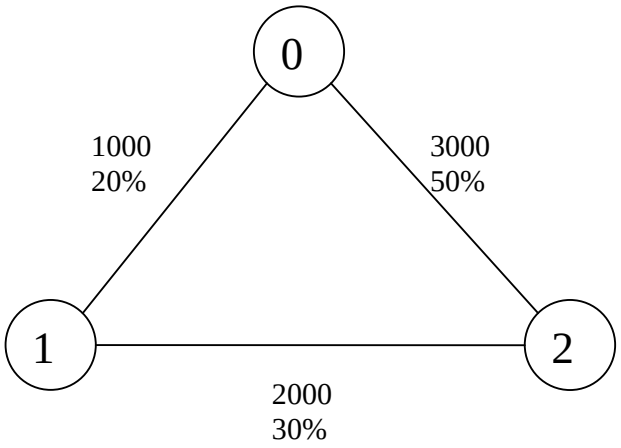

Figure3: A topology with loss rates greater than zero.

Run Num	Node	Ending Time	Difference	Source Node	Destination Node	Latency	Loss Rate	Type
Run 1	0	1202149225032		0	1	1000	20%	Bidirectional
	1	1202149226111	1079	1	2	2000	30%	Bidirectional
	2	1202149227077	2045	2	0	3000	50%	Bidirectional
Run 2	0	1202150010271		0	1	1000	20%	Bidirectional
	1	Never		1	2	2000	30%	Bidirectional
	2	Never		2	0	3000	50%	Bidirectional
Result	In some cases FloodDoneEvent may never be raised be cause of dropped messages							

Exercise 3

If a topology is set up that is unidirectional for some links between nodes the result will be that some nodes will not be able to receive messages from their neighbors. The flood algorithm could not work in these cases because the flood algorithm requires that all nodes can receive messages from all other nodes.

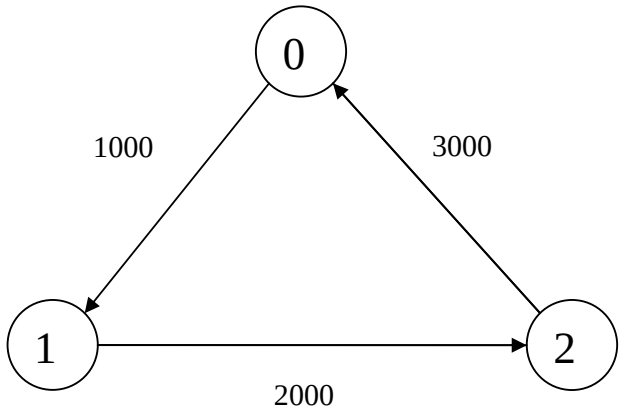

Figure3: A topology with loss rates greater than zero.

Run Num	Node	Ending Time	Difference	Source Node	Destination Node	Latency	Loss Rate	Type
Run 1	**0**	**never**		**0**	**1**	**1000**	0	unidirectional
	1	**never**		**1**	**2**	**2000**	0	unidirectional
	2	**never**		**2**	**0**	**3000**	0	unidirectional
Result	**With the topology of Figure3 , none of the nodes finished. That is reasonable be for example for node number 1. When node 1 parses the topology it finds out the it should send message to node 2 and also receive message from node 2 but in reality there is no way for node 1 to receive message from node 2 so it waits for a message from node 2 and never ends. There is similar case for other nodes.**							

In the topology below we relax some of the constrains on edges and prepare conditions for some of the nodes to finish, that lucky nodes are node 0 and 1.

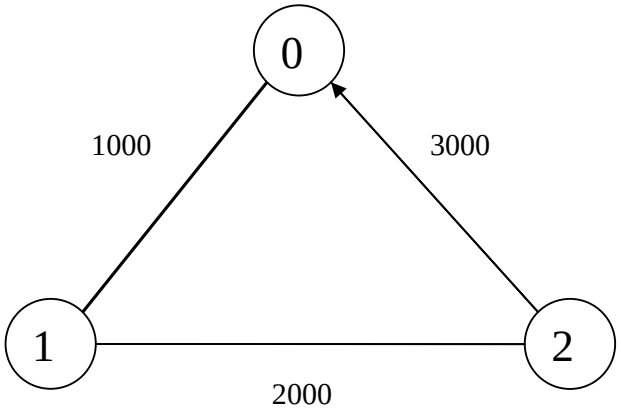

Figure4: Topology with relaxed edges

Lets see what happens?

	Node	Ending Time	Difference	Source Node	Destination Node	Latency	Loss Rate	Type
Run 1	0	1202247888593		0	1	1000	0	unidirectional
	1	1202247891671	3078	1	2	2000	0	unidirectional
	2	never		2	0	3000	0	bidirectional
Result	As it was predictable from the topology , node 2 never ends. Because it expects that it receive a message from node 0 , but there is not an edge from node 0 to it. Node 0 finishes before node 1 because it need to receive just one message from node 1 and the edge latency is just 1000. To send a FloodDone message , node 1 needs to receive two message from 0 and 1 with total Latency of 3000 (with average 1500) .							