

# Computational Methodologies: Distributed Computing

bcng57

## Question 1

Let  $G = (V_G, E_G)$  be a connected topology graph with specified root  $p_r \in V_G$ . Let  $D = \max\{\text{dist}(p_i, p_j) \mid p_i, p_j \in V_G\}$ , be the diameter of the graph.

(a)

### Theorem

The time complexity of the synchronous model of the Flooding algorithm on  $G$  is  $O(D)$ .

### Proof

In the worst case the distance from  $p_r$  to some  $p_i \in V_G$  is  $D$  as this is the furthest any message can travel from  $p_r$  without reaching an already visited node.

- In this case it takes at most  $D$  rounds for  $\langle M \rangle$  to reach  $p_i$ .
- This is because  $\langle M \rangle$  must be passed along each of the  $D$  nodes on the path from  $p_r$  to  $p_i$  and the message can only be transferred once per round.
- As  $p_i$  is the furthest node from  $p_r$ , after  $D$  rounds every node will have received  $\langle M \rangle$ .

Once a node has received  $\langle M \rangle$ , each node responds in one round. Therefore, after  $D + 1$  rounds every node will have responded with either  $\langle \text{parent} \rangle$  or  $\langle \text{already} \rangle$ . As all nodes have responded, each node will also have terminated.

Therefore, the overall time complexity is  $D + 1 = O(D)$

(b)

**Theorem**

The time complexity of the asynchronous model of the Flooding algorithm on  $G$  is  $O(D)$

**Proof**

Assume that the distance from  $p_r$  to some  $p_i \in V_G$  is  $D$ .

Let  $P$  be a path of length  $D$  between  $p_r$  and  $p_i$ .

Suppose there is a maximum delay on every edge in  $G$ .

Then the total max time for  $\langle M \rangle$  to get from  $p_r$  to  $p_i$  is  $D$ .

As  $p_i$  is the furthest any node can be from  $p_r$ , every node in  $V_G$  will have received  $\langle M \rangle$  after  $D$  maximum edge delays.

Once a node has received  $\langle M \rangle$ , it responds immediately. This response takes at most one maximum edge delay. Once every node has received a response from all its neighbours (excluding its parent), it terminates.

All nodes respond and, therefore, terminate in at most  $D + 1$  edge delays.

Therefore the overall time complexity is  $O(D)$ .

## Question 2

(a)

We shall prove this by contradiction. First, assume that there is some uniform synchronous algorithm  $A$ , that correctly computes the AND of the input bits.

Run  $A$  on a ring where all input bits are 1. In any round  $i$ , the states of all the processors are the same. Therefore, there must be some round  $r$  where all the processors terminate. Now run  $A$  on a ring with  $2(r+1)$  processors, where processor  $p_0$  has input bit 0 and the remaining processors have input bit 1. For  $A$  to be correct some message  $\langle M \rangle$  originating from  $p_0$  must reach all the nodes in the ring, however, it takes  $r+1$  rounds for  $\langle M \rangle$  to reach processor  $p_{r+1}$ . As  $p_{r+1}$  has initial bit 1, it will terminate in  $r$  rounds with its result as 1. This is incorrect, so  $A$  cannot be correct which is a contradiction. Therefore, a uniform synchronous algorithm for this problem does not exist.

(b)

Initially, each processor  $p_i$  sends a message  $\langle b_i, 0 \rangle$  to the right where  $b_i$  is its input bit.

Upon receiving a message  $\langle b_{j-1}, c_{j-1} \rangle$ , a processor  $p_j$  compares  $c_{j-1}$  to  $n$ .

- If  $c_{j-1}$  is not equal to  $n$ , send a message  $\langle b_{j-1} \text{ AND } b_j, c_{j-1} + 1 \rangle$  to the right.
- If  $c_{j-1}$  is equal to  $n$ , then its original message has been passed around the ring, so it can set the result to  $b_{j-1}$  and terminate

Each processor sends a message that must be forwarded to all  $n$  of the processors in the ring. This means that in total  $n^2$  messages are sent, so the message complexity is  $O(n^2)$ .

(c)

If the input bit of a processor is 0, then the processor sets its result to be 0, sends a  $\langle terminate \rangle$  message in both directions and terminates.

A processor with input bit 1, waits for  $\lfloor n/2 \rfloor$  rounds.

- If it receives a  $\langle terminate \rangle$  in this time, then it sets its result to 0, forwards the  $\langle terminate \rangle$  message and terminates.
- If no message is received within  $\lfloor n/2 \rfloor$  rounds, then it sets its result to 1 and terminates.

In the worst case, where all the processors have 0 as an input bit, they all send 2  $\langle terminate \rangle$  messages to the left and right. This means that in total  $2n$  messages are sent, so overall the message complexity is  $O(n)$ .