

# 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  $D = \max\{dist(p_i, p_j) | p_i, p_j \in V_G\}$ , 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  $\langle parent \rangle$  or  $\langle already \rangle$ . 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 the maximum edge delay. Once every node has received a response from all its neighbours (excluding the parent), it terminates.

Overall, all nodes have responded and terminated in at most  $D + 1$  edge delays.

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

## Question 2

(a)

(b)

Code for each node. Initially,  $asleep = \text{true}$  for all nodes.  $in$  is the binary input,  $n$  is the total number of nodes

```
1 upon receiving no message :
2   if asleep then
3     asleep = false
4     send  $\langle in, 0 \rangle$  to left
5
6 upon receiving  $\langle bit, count \rangle$  from right :
7   if count == n: // If visited nodes == total number of nodes, end
8     result = bit
9     terminate
10  else:
11    send  $\langle bit \ \& \ in, count + 1 \rangle$  to left
```

### Plain English description

Each processor sends a message to the left with its input bit and a counter that starts at 0.

Upon receiving a message of this form, a processor compares the counter to  $n$ .

- If the counter is equal to  $n$ , then its original message has been passed around the ring, so it can set the result to the received bit and terminate

- If the counter is not equal to  $n$ , send a message to the left containing the received bit AND input bit, and the counter incremented by 1.

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)

Code for each node. Initially, *asleep* = **true** for all nodes. *in* is the binary input,  $n$  is the total number of nodes.

This algorithm has  $\lfloor n/2 \rfloor$  rounds numbered  $1, \dots, \lfloor n/2 \rfloor$ .

```

1 upon receiving no message :
2   if in == 0 then
3     result = 0
4     send <terminate> to left and right
5     terminate
6   else if asleep then
7     asleep = false
8
9 upon receiving <terminate> from left (resp., right):
10  result = 0
11  send <terminate> to right (resp., left)
12  terminate
13
14 if no messaged received by round  $\lfloor n/2 \rfloor$ :
15   result = 1
16   terminate

```

**Plain English description** If the input bit of a processor is 0, then the processor sets its result to be 0, sends a *<terminate>* message in both directions and terminates.

A processor with input bit 1, waits for  $\lfloor n/2 \rfloor$  rounds. If it receives a *<terminate>* in this time, then it sets its result to 0, forwards the *<terminate>* 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 *<terminate>* messages to the left and right. This means that in total  $2n$  messages are sent, so overall the message complexity is  $O(n)$ .