Distributed Computing Summative Coursework

wtxd25

1 Question 1

(a) The time complexity of a synchronous model is the maximum number of rounds of any execution of the algorithm. In each round any given processor can send messages, receive incoming messages, and process the data obtained from the incoming messages. The Flooding algorithm works by broadcasting a message to all neighbours and waiting for a response from them to know whether they are a child or not. In this way the algorithm is able to construct a spanning tree T of the input graph G.

The diameter, d, of the graph G is the maximum distance between any two vertices of the graph, where distance between two vertices is the number of edges traversed to get from one to the other. In the worst case, the root is one of the vertices with which the diameter is defined, u. The message must travel from the root to the other vertex in the diameter definition, v. By definition the path $u \to v$ is the longest path in the graph. The message takes at most d rounds to reach the vertex v from u. During this time, the vertices on this path through which messages have been passed simultaneously (in the same round) send a message to the next vertex in the path and the previous vertex to inform it whether it is a parent or other. It takes one additional round for v to send a message to its parent vertex and for this vertex to add it to the set of children and terminate. The vertex v has one more round where it realises that it has no vertices to send to and has received no messages as all other vertices have terminated. In this round the processor terminates itself. This means that the total number of rounds in the worst case is $d + 2$, which is $O(d)$.

(b) In the asynchronous model there is usually a maximum message delay, for convenience this message delay is equal to 1 unit of time. The time complexity of the asynchronous model is the number of units of time taken to solve a problem in the worst case. In the previous answer we gave a brief overview of the Flooding algorithm in natural language and in the same way we make use of the diameter of the graph G, d.

In the worst case the root of the graph, p_r is one of the vertices in the diameter, u. We know that the maximum distance leaf from $p_r = u$ is the other vertex in the diameter definition, v, and that the length of this path is d. Messages are passed in both directions along this longest path meaning that the length of the walk is $2d$. Assume that every edge has maximum message delay, and so we use 1 unit of time for each edge traversal. Therefore, the total number of units of time used in the worst case is $2d \Rightarrow$ the time complexity of the flooding algorithm in asynchronous model is $O(d)$.

2 Question 2

(a) An anonymous ring is where each processor in the ring is indistinguishable; doesn't have a unique identifier. We want to give an argument that no uniform, synchronous algorithm can perform the AND of input bits to individual processors on this anonymous ring. A uniform algorithm means that the number of processors is unknown to the algorithm. A synchronous algorithm is an algorithm which operates in rounds where each processor can send/receive messages and process data received in a round. Let's assume such an algorithm exists. This would mean that each processor p_i would have a bit input and the outcome of the algorithm would be $p_1 \wedge p_2 \wedge ... \wedge p_n$ for some n which is unknown to the processors. This is not possible: to coordinate the AND operation we could use a leader processor, however, the issue her is that no anonymous leader election algorithm in rings exists. This means that the AND operation would need to be performed distributed over the processors. If the processors each had knowledge of the number of processors in the ring (non-uniform) then they could perform communication for a set number of rounds, as in later answers. However, if the number is unknown, and the ring is anonymous, there is no way of knowing when all of the bits have been communicated. The processors would not be able to tell if they had received all the bits; could not keep track of the IDs of each process or count the number of bits received. As such there is no possible way that a uniform, synchronous algorithm on anonymous rings exists to compute the AND of input bits to individual processors.

(b) The algorithm in natural language for each processor p_i is as follows:

1. Each processor takes some bit as input, b_i and stores this value.
2. count $\leftarrow 1$
3. While count is less than number of processors n do the following:
4. Send b_i value to both neighbours in the ring.
5. The values that are received from the neighbours in the ring should be AND-ed with the value b_i and the result stored in b_i.
6. Add 2 to the count value

We add 2 to the count value each iteration as in each iteration the value of b_i has the value of two other processors AND-ed with it. Duplicate ANDs are inconsequential because $a \wedge b \wedge a \equiv a \wedge b$. In the end the result on each processor will be the AND of all bit inputs. The number of messages sent are 2 from each of the n processors each round. The max number of rounds of the ring is less than $\frac{n}{2}$. Therefore, $2n \cdot \frac{n}{2} = n^2 \Rightarrow$ algorithm sends $O(n^2)$ messages.

(c) The algorithm in natural language for each processor p_i is as follows:

1. Each processor takes some bit as input, b_i and stores this value
2. If the value of b_i is 0 then send a <terminate> message to the left, return 0 and terminate execution.
3. If the value of b_i is 1 then sleep for n turns or until receive a message. If received no message then return 1. If sleep process interrupted by receiving a message then send the <terminate> message to the left, return 0, and terminate.

In the algorithm above only the processors which have a 0 send any messages, this means that in the worst case, the message complexity will be O(n) if all processors have a value 0. The algorithm works because a 0 will instantly nullify a conjunction operation; obviously return 0. The return value will only be 1 if all values of b_i are 1 which is true if no messages reach the current node telling it to terminate in n turns.