Distributed Computing

ffgt86

February 17, 2020

1 Perform a precise analysis of the time complexity of the Flooding algorithm.

1.1 The synchronous model

The *distance* between two nodes u, v in an undirected graph is the number of hops of a minimum path between u and v. The *radius* of a node u in a graph is the maximum distance between u and any other node. The radius of a graph is the minimum radius of any node in the graph. The *diameter* of a graph is the maximum distance betwen two arbitrary nodes.

In the synchronous model, process execution speeds and message delivery delays are upper-bounded by a fixed k.

If node v receives the message first from node u, then node v calls node u *parent*. This parent relation defines a spanning tree T.

In every execution of the broadcast algorithm in the synchronous model, every process at distance t from p_r in the spanning tree receives M in round t.

THe number of rounds needed to reach all nodes: the diameter of G.

By induction on the distance t of a process from p_r.

$t = 1$. Each child of p_r receives M from p_r in the first round. Assume that every process at distance $t - 1 \geq$ from p_r receives M in round $t - 1$.

Let p be any process at distance t from p_r. Let p' be the parent of p in the spanning tree. Since p' is at distance $t - 1$ from p_r, by the induction hypothesis, p' receives M in round $t - 1$. By the description of the algorithm, p receives M from p' in the next round.

Every processor receives M after at most D time and at most $|E|$ messages, where D is the diameter of the network, and E is the set of (directed) edges in the network. Proof is by induction.

Let $d(root, v) = k > 0$. Then v has a neighbour u such that $d(root, u) = k - 1$. By the induction hypothesis, u receives M for the first time no later than time $k - 1$. u sends M to all its neighbours, including v at k, so M arrives at v no later than time $(k - 1) + 1 = k$.

Each process only sends M to its neighbours once, so each edge carries at most one copy of M. The message complexity is therefore $|E|$.

1.2 The asynchronous model

In asychronous systems the flooding algorithm terminates after R time units, where R is the radius is the source. However, the constructed spanning tree may not be a breadth-first search spanning tree.

In every execution of the broadcast algorithm in the asynchronous model, every process at distance t from p_r in the spanning tree receives M in

2 Consider an anonymous ring where processors start with binary inputs.

2.1 Give an argument that there is no uniform synchronous algorithm for computing the AND of the input bits.

Proof is by contradiction. Assume that there is an uniform synchronous algorithm A that computes AND. Assume a ring where all inputs are 1. In any round i the states of all processors are identical, and these states do not depend on the size n of the ring, as the algorithm is uniform. As A is correct, there must exist a round t such that all processors terminate and output 1.

Now run A on a ring of size $2(t + 1)$, with 1 processor p with input 0 and $2t + 1$ processors with input 1. p disseminates 0. However, the processor 'opposite' p will not receive 0 by round t; it will instead terminate and output 1. This contradicts the assumption that A is correct (in which case 0 should be output at all processors).

2.2 Present an asynchronous (non-uniform) algorithm for computing the AND. The algorithm should send $O(n^2)$ messages in the worst-case.

Let n be the number of processors. Each processor sends a message to its right neighbour and waits for messages from its left neighbour. Each message m comprises a counter $hop = 0$ and a state x. hop is incremented every time m is sent. If $m.hop == n$, the message has arrived where it started.

When a processor p with input state x receives a message m, it updates $m.x = p.x \wedge m.x$. It then checks $m.hop$. If $m.hop < n$, it forwards the message to the right. If $m.hop == n$, the processor terminates.

As each processor sends a message, and each message makes n hops, the message complexity is $O(n^2)$.

2.3 Present a synchronous algorithm for computing the AND. The algorithm should send $O(n)$ messages in the worst case.

Let n be the number of processors. A processor with an initial state of 0 sends a message in both directions and halts (in state 0). A processor with an initial state of 1 waits for $\lfloor n/2 \rfloor$ cycles. If it receives a message during that period, then it forwards it and halts in state 0. If by cycle $\lfloor n/2 \rfloor$ a processor has not received any message, it halts in state 1. The total number of messages sent is $O(n)$.