ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2017 Prof. R. Wattenhofer

Distributed Systems Part II

Exercise Sheet 2

Quiz

1 Consensus with Edge Failures

In the lecture we only discussed node failures, but we always assumed that edges (links) never fail. Let us now study the opposite case: Assume that all nodes work correctly, but up to f edges may fail.

Analogously to node failures, edges may fail at any point during the execution. We say that a failed edge does not forward any message anymore, and remains failed until the algorithm terminates. Assume that an edge always simultaneously fails completely, i.e., no message can be exchanged over that edge anymore in either direction.

We assume that the network is initially fully connected, i.e., there is an edge between every pair of nodes. Our goal is to solve consensus in such a way, that *all* nodes know the decision.

a) What is the smallest f such that consensus might become impossible? (Which edges fail in the worst-case)

b) What is the largest f such that consensus might still be possible? (Which edges fail in the best-case)

c) Assume that you have a setup which guarantees you that the nodes always remain connected, but possibly many edges might fail. A very simple algorithm for consensus is the following: Every node learns the initial value of all nodes, and then decides locally. How much time might this algorithm require?

Assume that a message takes at most 1 time unit from one node to a direct neighbor.

Basic

2 Deterministic Random Consensus?!

Algorithm 2.15 from the lecture notes solves consensus in the asynchronous time model. It seems that this algorithm would be faster, if nodes picked a value deterministically instead of randomly in Line 23. However, a remark in the lecture notes claims that such a deterministic selection of a value will not work. We did it anyway! (See algorithm below, the only change is on Line 23).

Show that this algorithm does not solve consensus! Start by choosing initial values for all nodes and show that the algorithm below does not terminate.

Algorithm 1 Randomized Consensus (Ben-Or)

```
1:  v_i ∈ {0,1}                          ◁ input bit
2:  round = 1
3:  decided = false
4:  Broadcast myValue(v_i, round)
5:  while true do
        Propose
6:      Wait until a majority of myValue messages of current round arrived
7:      if all messages contain the same value v then
8:          Broadcast propose(v, round)
9:      else
10:         Broadcast propose(⊥, round)
11:     end if
12:     if decided then
13:         Broadcast myValue(v_i, round+1)
14:         Decide for v_i and terminate
15:     end if
        Adapt
16:     Wait until a majority of propose messages of current round arrived
17:     if all messages propose the same value v then
18:         v_i = v
19:         decided = true
20:     else if there is at least one proposal for v then
21:         v_i = v
22:     else
23:         Choose v_i = 1
24:     end if
25:     round = round + 1
26:     Broadcast myValue(v_i, round)
27: end while
```

Advanced

3 Consensus with Bandwidth Limitations

Consensus with no failures, a fully connected network and unlimited bandwidth is trivial: First, every node sends its value to all other nodes. Second, every node waits for all values, and then decides.

So far we only studied failures. However, in practice bandwidth limitations are often of great importance as well. To simplify the problem, we assume no node crashes and no edge crashes in this exercise. Additionally, you can assume that all nodes have unique ids from 1 to n.

We assume that all messages are transmitted reliably, and arrive exactly after one time unit. The bandwidth limitation is as follows: Assume that every node can only send *one* message (containing one value) to *one* neighbor per time unit. E.g., at time 0, u_1 can send a message to u_2, at time 1 a message to u_3, and so on. However, u_1 cannot send a message to both u_2 and u_3 at the same time! Also, a node cannot send multiple values in the same message.

a) Develop an algorithm that solves consensus in this scenario. Optimize your algorithm for runtime!

b) What is the runtime of your algorithm?

c) Assume that you not only need to solve consensus, but the more challenging task that every node must learn the input values of all nodes. Show that this problem requires at least $n - 1$ time units!