ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2015

Prof. R. Wattenhofer

Distributed Systems Part II

Solution to Exercise Sheet 2

1 Consensus with Edge Failures

a) $f = n - 1$. Remove all edges from a node – this node will not receive any messages and cannot decide.

Proof why $n-2$ is too small to prevent consensus:

Note that if $f < n - 1$ the remaining network is always connected. To prove that, assume by contradiction, that you can split the network into two partitions by only removing $n - 2$ many edges.

In that case there are two groups of nodes K and L which do not have a remaining edge between them: one containing k many nodes, and one containing l many nodes. Note that $k \geq 1, l \geq 1$ and $k + l = n$.

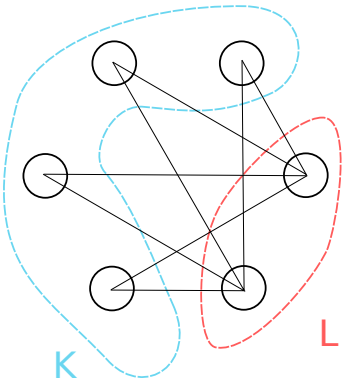

Figure 1: $n = 6$ nodes partitioned into two groups K and L. Since they should be disconnected, all edges between the two sets have to fail. (Edges within the groups are not drawn to avoid cluttering up the drawing.)

Since the network is initially fully connected, there are initially $k \cdot l$ many edges between the two groups. As we assumed that the two groups are disconnected, all of these edges have failed, and for that $k \cdot l \leq n - 2$.

From $k + l = n$ follows $k = n - l$. Plugging that into the inequality we know that

$$k \cdot l = (n - l) \cdot l = n \cdot l - l^2 \leq n - 2$$

or

$$-l^2 + nl - n + 2 \leq 0$$

Since n is fixed, we need to see if there is an l which satisfies this inequality, subject to $l \in [1, n-1]$. The first derivative of the inequality $(-2l + n)$ shows that there is only one extreme point ($l = n/2$), and it is a local maximum (compare, e.g., the second derivative). Since the function is continuous, it follows that it is minimal on the border cases for l, i.e., $l = 1$ or $l = n - 1$. Plugging in these values for l we see:

$$-1 + n - n + 2 \leq 0$$

$$-(n-1)^2 + n \cdot (n-1) - n + 2 = -n^2 + 2n - 1 + n^2 - n - n + 2 = 1 \leq 0$$

Which both do not hold. Hence the inequality is unsatisfiable, and the split into two partitions is not possible[1]. □

We showed that with only $f = n - 2$ many failures, the network always stays connected. Thus, all nodes can – by routing messages via other nodes – always learn all initial values, and then decide.

b) $f = \frac{n(n-1)}{2} - (n-1) = \frac{n^2-3n+2}{2}$. Remove all edges ($\frac{n(n-1)}{2}$) except a path with connects all nodes $(n-1)$.

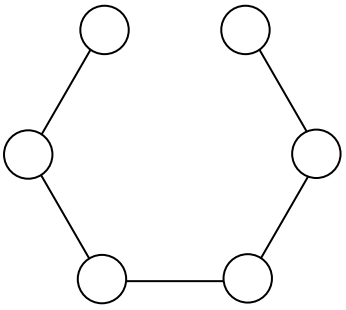

Figure 2: Smallest remaining connected network with $n = 6$.

c) Up to $n - 1$ time units.

Observe that the time to terminate is equal to the longest shortest path between two nodes in the network. This path contains at most n nodes, since there are only n nodes in the network, and a shortest path can never contain a loop. In **b)** we showed that such a scenario can indeed happen, where we have two nodes in distance $n-1$. Hence, the algorithm requires up to $n - 1$ time units.

2 Deterministic Random Conensus?!

We partition the nodes into two groups A and B. All nodes in A start with initial value 0, all nodes in B with 1. Let A contain $\lceil n/2 \rceil + 1$ many nodes, and all other nodes are in B.

We show that – only with a bad scheduling, we do not even need any crash failures – it is possible that the system remains in the identical distribution as before; and since there is no randomness anymore, we can therefore deduce that the system will not reach consensus.

1) In the Propose phase only one node $u \in A$ receives messages only from A. All other nodes receive messages from both A and B. Hence, u proposes 0, and all other nodes propose $\perp$.

[1] What did we do in this proof? We showed that there is no cut of size $n - 2$ in the fully connected graph with n nodes.

2) In the Follow phase all nodes from A receive the proposal of u, and all nodes from b do not receive the proposal of u. Thus, all nodes in A set $v = 0$, and all nodes in B set $v = 1$. This is exactly the same configuration as initially assumed!

Since our goal is to solve consensus in the worst-case scenario (allowing for a worst-case scheduling), this scheduling might happen in every round and the algorithm does not terminate.

3 Consensus with Bandwidth Limitations

a) Note that we assumed that no nodes or edges crashed. Hence, we use a designated leader decides for a value, and the only question is how fast can the leader distribute its value to all nodes.

For readability we call one time unit a *round*.

It is obvious that the leader can distribute its value one by one to every other node, requiring $n - 1$ rounds. But can we do better?

Yes: By having a carefully selected forwarding mechanism. Let u_1 be the leader. In the first round, u_1 sends its value to u_2. In the second round u_1 sends the value to u_3, and u_2 to u_4. This pattern is easily generalized: Every node which receives the value in round i will send it to nodes in all rounds $i + 1, i + 2, \ldots$. Note that with this approach the number of nodes which know the value doubles in every round.

Since the number of nodes which know the value of the leader doubles in every round, it follows that the algorithm requires only $\log(n)$ many rounds.

b) It requires $\log(n)$ time units.

c) The task is that n nodes learn $n - 1$ values. Since we assumed that each message can only contain one value, it follows that every node needs to receive at least $n - 1$ many messages. Thus, there are in total at least $n \cdot (n - 1)$ many received messages. Of course the number of received messages is equal to the number of sent messages (assuming no message loss). Observe that in any round every node can send at most one message, thus, the number of sent messages in a round is at most n. Therefore, any algorithm which requires $n \cdot (n - 1)$ many sent messages requires at least $n - 1$ many rounds.