Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Prof. Dr. Peter M. Fischer
Y. Maus

April 23, 2015

Distributed Systems, Summer Term 2015
Problem Set 1

The assignment is due on Tuesday, May 5, 2015, 14.15h. You can either hand it in electronically (*yannic.maus@cs.uni-freiburg.de*) or hand it in at the tutorial session itself.

Exercise 1: Schedules

Given are three nodes v_1, v_2 and v_3 which are connected via FIFO channels, that is, (two) messages, which are sent from some node i to the some node j, will arrive at node j in the order in which node i released the messages.
Devise **one** possible schedule S which is consistent with the following local restrictions to the three nodes.

- $S|1 = s_{1,3}\ s_{1,3}\ r_{1,2}\ r_{1,3}\ s_{1,2}\ r_{1,2}\ s_{1,3}$,
- $S|2 = s_{2,3}\ s_{2,1}\ r_{2,1}\ s_{2,1}$,
- $S|3 = r_{3,2}\ r_{3,1}\ s_{3,1}\ r_{3,1}\ r_{3,1}$.

$s_{i,j}$ denotes the send event from node i to node j and $r_{j,i}$ denotes the event that node j receives a message from node i.

Solution

A solution can be obtained by drawing the diagram as in the lecture. One possible schedule is the following

$$s_{2,3}\ s_{1,3}\ r_{3,2}\ r_{3,1}\ s_{2,1}\ s_{1,3}\ r_{1,2}\ s_{3,1}\ r_{1,3}\ s_{1,2}\ r_{2,1}\ s_{2,1}\ r_{1,2}\ s_{1,3}\ r_{3,1}.$$

Exercise 2: (Variations) of Two Generals

In the lecture we considered the (deterministically unsolvable) **Two Generals** consensus problem:

- two deterministic nodes, synchronuous communication, unreliable messages,
- **input**: 0 or 1 for each node,
- **output**: each node needs to decide either 0 or 1,
- **agreement**: both nodes must output the same decision (0 or 1),
- **validity**: if both nodes have the same input $x \in \{0, 1\}$ and no messages are lost, both nodes output x,
- **termination**: both nodes terminate in a bounded number of rounds.

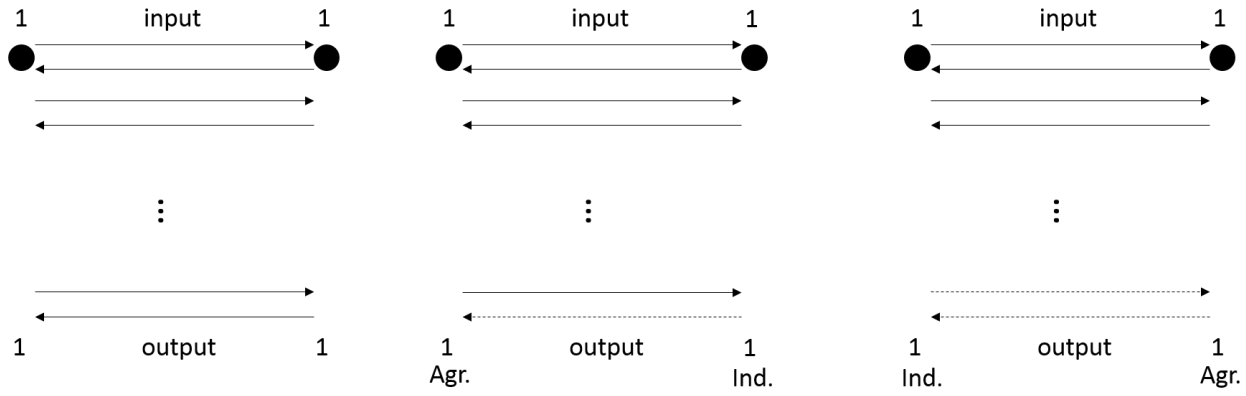


Figure 1: We begin with the execution E and show that it is indistinguishable from E' via a long chain of indistinguishable executions (see last page).

In this exercise we consider three modifications of the model. For each of them, either give a (deterministic) algorithm or state a proof which shows that the variation cannot be solved deterministically.

a) There is the guarantee that within the first 7 rounds at least *one* message in *each* direction succeeds.

b) There is the guarantee that within the first 7 rounds at least *one* message succeeds.

c) Let $k \in \mathbb{N}$ be a natural number. The input for each node is a number $x_i \in \{0, \ldots, k\}$.

Goal: If no message gets lost *and* both have the same input $x \in \{0, \ldots, k\}$, both have to output x. In all other cases the nodes should output numbers which do not differ by more than one. The algorithm still has to terminate in a finite number of rounds.

Hint: This problem is solvable. Ideas from the lecture might help.

Solution

a) As there is the guarantee that at least one message in *each* direction succeeds every node sends its value repeatedly for seven rounds. At least one time it will reach the other node. So both nodes know both values and can decide on a output by a previously fixed algorithm, e.g., output $val_1 \cdot val_2$.

b) The problem is not solvable for two deterministic generals. Assume that it is solvable in T rounds. By a sequence of executions we show that both nodes need to have the same output in the following two executions.

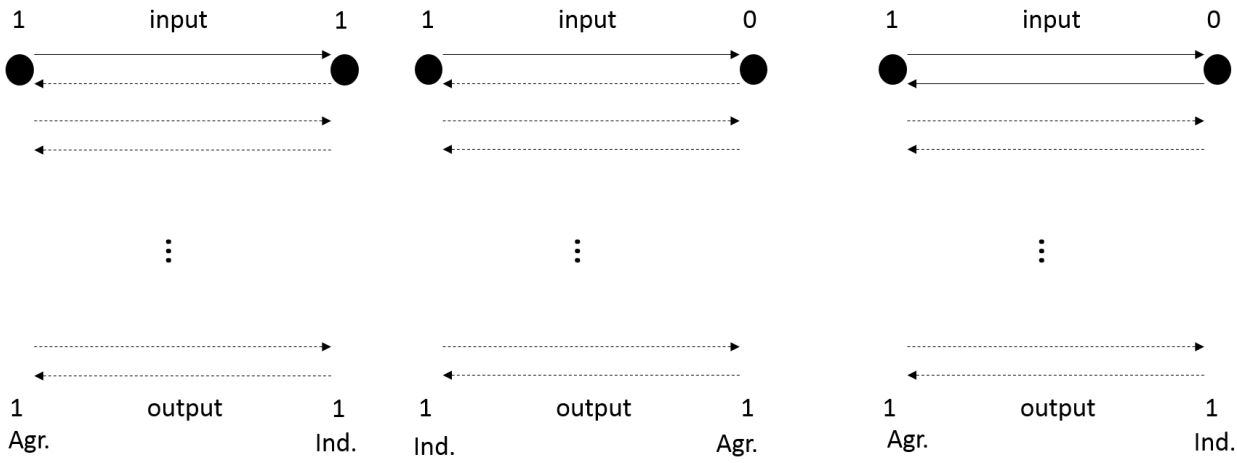
- E: both inputs are 1, all messages are delivered
- E': both inputs are 0, all messages are delivered

Because of validity both nodes need to output 1 in the case of E and 0 in the case of E', a contradiction.

The proof is similar to the proof in the lecture - one only has to be careful in the last step to always keep at least one successfully delivered message.

Important is that we drop at most one message at a time to keep the two following executions similar (see definition of similar in the lecture). The reasoning is always that two similar executions are indistinguishable for one node and thus he has to output the same value in both execution. Then, to fulfill agreement all nodes need to output the same in both executions.

c) The problem is solvable with the help of the level algorithm from the lecture. Both nodes execute the level algorithm for k rounds.


1
input
1
1
output
1
Agr.
Ind.
1
input
0
1
output
1
Ind.
Agr.
1
input
0
1
output
1
Agr.
Ind.
1
input
0
1
output
1
Ind.
Agr.
0
input
0
1
output
1
Agr.
Ind.
0
input
0
1
output
1
Ind.
Agr.
0
input
0
1
output
1

1) Both nodes initialize their level with 0.

2) In each round the nodes send their own level (and its own value) to the other node

3) The level is updated as follows: $l_u := max\{l_u, l_{u'} + 1\}$, where $l_{u'} = -1$ if no level is received.

Let l_u denote the level of node u after executing the level algorithm for k rounds and define $k_u := k - l_u$. Then node u outputs

$$\max\{0, \max\{x, y\} - (k_u)\} \tag{1}$$

Proof that his solution is correct:

From the lecture we know that the levels differ by at most two and they are equal to the number of rounds if all messages are delivered.

- Case 1: All messages get delivered. ($l_{u_1} = l_{u_2} = k$)
  At first if all messages get delivered $k_{u_1} = k_{u_2} = 0$ and both nodes know each others value. Then their common output will be $\max\{0, \max\{x, y\} - (k_u)\} = \max\{0, \max\{x, y\} - 0\} = \max\{x, y\} = x = y$.
- Case 2: $l_{u_1} = 0$ or $l_{u_2} = 0$
  W.l.o.g. let $l_{u_1} = 0$, that is $l_{u_2} \leq 1$. Then $k_{u_1} = T$ and $\max\{0, \max\{x, y\} - (k_{u_1})\} = 0$. $k_{u_2} \geq T - 1$ and $\max\{0, \max\{x, y\} - (k_{u_1})\} \leq 1$.
- Case 3: else
  Not all messages have been delivered because $l_{u_1} \neq k$ or $l_{u_2} \neq k$, so validity cannot be violated. As both nodes have a level greater than 0, they know each others values. Their levels differ by no more than one and hence their output, i.e., $\max\{0, \max\{x, y\} - (k_{u_1})\}$ and $\max\{0, \max\{x, y\} - (k_{u_2})\}$, will differ by at most one.

Exercise 3: Asynchronous Message Passing, BFS Algorithm

In the lecture we introduced the distributed Bellman-Ford Algorithm for constructing a BFS tree in an asynchronous message passing system. The time complexity of the algorithm is $\mathcal{O}(D)$ and the message complexity is $\mathcal{O}(mn)$, where D is the diameter of the graph, n the number of nodes and m the number of edges.
The number of edges in any graph is in $\mathcal{O}(n^2)$, which implies an upper bound of $\mathcal{O}(n^3)$ on the number of messages. Find a graph and an execution of the algorithm in which $\Theta(n^3)$ messages are sent and explain your solution.

Solution

We describe an asynchronous execution of the BFS algorithm on the fully connected graph with n nodes (n clique). The first node v_1 sends messages to all its neighbors ($n-1$ messages). The message to v_2 arrives first. It sends a message to all its neighbors except for v_1 ($n-2$ messages). The message from v_2 to v_3 arrives next (even before all other messages initiated by v_1. So now v_3 sends a message to all its neighbors and so on. After v_n has received the message from v_{n-1} the message from v_1 to v_3 arrives. No v_3 has a shorter path to the initiating node v_1 (1 hop instead of two). It broadcasts this to all its neighbors except for v_1. The process repeats. In the end we obtain $\Theta(n^3)$ messages in total.