Assignment 5.2 v1

Project

An implementation of the **asynchronous Ben-Or agreement** algorithm, working on the complete graph.

The project shall consist of a single source file, containing definitions for **Node** and for **Arcs**, an adapted version of our previous Arcs (not part of the algorithm).

We use **automaker**, with known convention regarding **stdin** and **stdout**.

Config

Arcs reads the config file from **stdin**, e.g.:

```
4 1 0 20
1 0 1
2 0 -1
3 1 -1
4 1 -1
1 0 2 3
2 1 10
2 3 10 20
2 4 20 10
0 0 1 2
```

Breakdown

The first line indicates, in order: the number of nodes, N=4; the maximum fault tolerance, F=1; the V0 value; the maximum number of rounds that will be simulated (even if the nodes do not decide), i.e. a loop limit, L=20.

Each node has its own random number generator, initialised by a **seed** = 7 × node number (use this value for consistency).

Here, our rounds are counted sequentially across steps, i.e.:

1 : step 1, round I (aka Report)

2 : step 1, round II (aka Proposal)

3 : step 2, round I (aka Report)

4 : step 2, round II (aka Proposal)

Each of the following N lines indicates: the node number (1..N), its initial choice (0/1), the number of sending rounds completed before crashing (1 for #1), where -1 indicates no bounds (no failure).

Then, each of the other following lines indicate arcs delays, e.g.:

```
2 3 10 20
```

Format: from (2), to (3), then a sequence of positive delay values, one for each round (10 for round 1, 20 for round 2 and the following rounds). Thus, the last delay is repeated if required. There are no round #0 delays – these are init messages sent by Arcs.

Receiver 0 indicates all receivers not explicitly mentioned (excluding self-addressed messages).

Sender 0 indicates all senders not explicitly mentioned (excluding self-addressed messages).

Unless specified, the overall default delay is also 1, and this includes self-addressed messages.

Messages (suggested format)

(time, round, from, to, value)

Time is an "universal logical time", determined by Arcs (according to delays).

As used here, round combines the textbook step and round numbers.

Possible values are: 0, 1, and 2 (for undefined / null).

Output

A line is printed by the nodes themselves, just before sending out new round messages. Crashing nodes print such a line before stopping, and so do nodes that reach the loop limit L.

Each output line contains: time, round, current node, *received*, x, z; where *received* is a string containing the values received from the other nodes for this round, with missing values indicated by an underscore.

Here x is not just the initial value, but the value that will be send out at the end of the current round.

Initially, z will be undefined, and after being defined will be followed by bang (!).

Sample output (excerpt):

```
0 0 1 ____ 0 2
0 0 2 ____ 0 2
0 0 3 ____ 1 2
0 0 4 ____ 1 2
1 1 1 0_11 2 2
1 1 2 _011 2 2
1 1 3 __11 2 2
1 1 4 __11 2 2
2 2 2 _2__ 2 2
2 1 3 0_11 2 2
2 1 4 0_11 2 2
3 2 3 __2_ 2 2
3 2 4 ___2 2 2
4 2 2 _222 0 2
4 2 3 __22 2 2
4 2 4 __22 2 2
5 3 2 _0__ 2 2
…
89 20 2 _111 1 1 !
89 20 4 _111 1 1 !
105 20 3 _111 1 1 !
```

Pseudocode

We base our project on the latest version on BenOr's algorithm, given in:

Marcos K. Aguilera, Sam Toueg

The correctness proof of Ben-Or's randomized consensus algorithm

Distrib. Comput. (**2012**) 25:371–381, DOI 10.1007/s00446-012-0162-z

For consistency, we give here an adapted version of the loop, as used by the model solution:

```
X = from Arcs (0 or 1)
Z = undef (2)
Round = 0

loop
        Round += 1  // odd round = report
        SendAll (Round, X)
        X = 2 (undef)
        Wait for messages of the form (Round, v) from at least N-F processes
        If received > N/2.0 with the same v (0, or 1)
        Then X = v

        Round += 1  // even round = propose
        SendAll (Round, x)
        X = 2 (undef)
        Wait for messages of the form (Round, v) from at least N-F processes
        If received ≥ F+1 with the same v (0, or 1)  // not 2
        Then Z = v (decide)
        If received ≥ 1 with the same v (0, or 1)  // not 2
        Then X = v Else X = Random (0, or 1)
```

Notes: The quotas required by the Waits may eventuate in several steps. Also, we may receive future round messages, that need to be stashed until their round.

Skeleton (optional).