Paxos – Sample Input 2

Suppose we want to simulate a scenario where there are two proposers, one of which fails and later recovers. We define the simulation as follows:

$$(n_P = 2, n_A = 3, t_{\max} = 50, E)$$

Where E contains four events:

- $(t = 0, F = \emptyset, R = \emptyset, \pi_c = p_1, \pi_v = 42)$
- $(t = 8, F = \{p_1\}, R = \emptyset, \pi_c = \varnothing, \pi_v = \varnothing)$
- $(t = 11, F = \emptyset, R = \emptyset, \pi_c = p_2, \pi_v = 37)$
- $(t = 26, F = \emptyset, R = \{p_1\}, \pi_c = \varnothing, \pi_v = \varnothing)$

In other words, we have a system with two Proposers and three Acceptors. p_1 proposes value 42 at tick 0, but fails at tick 8. Then, at tick 11, p_2 proposes a different value (37). Finally, p_1 recovers at tick 26.

Ticks 0–7

Same as in Example 1.

Contents of N at the end of tick 7

m.type	m.src	m.dst	Other attributes
ACCEPT	p_1	a_2	proposal_id=1, value=42
ACCEPT	p_1	a_3	proposal_id=1, value=42
ACCEPTED	a_1	p_1	proposal_id=1, value=42

Tick 8

In this tick, there is an event which specifies that p_1 will go into a failed state (remember that this happens *before* checking the contents of the network). So, since all the messages in N have p_1 as either the source or the destination, they cannot be delivered.

Contents of N at the end of tick 8

m.type	m.src	m.dst	Other attributes
ACCEPT	p_1	a_2	proposal_id=1, value=42
ACCEPT	p_1	a_3	proposal_id=1, value=42
ACCEPTED	a_1	p_1	proposal_id=1, value=42

Ticks 9 and 10

During these ticks, no messages are delivered. However, the simulation carries on because $t_{\max}$ hasn't been reached, and there are still messages pending delivery.

Ticks 11-14

In tick 11, there is an event specifying that p_2 must propose the a new value (37). Ticks 11-14 are similar to ticks 0-3, except p_2 will use a larger proposal number (2). Additionally, a_1 had previously accepted a value (value 42 in proposal 1). So, when sending a `PROMISE` message to p_2, it specifies the prior value it accepted (and its proposal number).

Contents of N at the end of tick 14

m.type	m.src	m.dst	Other attributes
`ACCEPT`	p_1	a_2	proposal_id=1, value=42
`ACCEPT`	p_1	a_3	proposal_id=1, value=42
`ACCEPTED`	a_1	p_1	proposal_id=1, value=42
`PROMISE`	a_1	p_2	proposal_id=2, prior_proposal=$(v = 42, p = 1)$
`PROMISE`	a_2	p_2	proposal_id=2, prior_proposal=$\varnothing$
`PROMISE`	a_3	p_2	proposal_id=2, prior_proposal=$\varnothing$

Ticks 15-16

Similar to ticks 4-5, except p_2 cannot use the value specified in the `PROPOSE` message. Since one of the Acceptors had previously accepted value 42, then p_2 is bound to use that value.

Contents of N at the end of tick 16

m.type	m.src	m.dst	Other attributes
`ACCEPT`	p_1	a_2	proposal_id=1, value=42
`ACCEPT`	p_1	a_3	proposal_id=1, value=42
`ACCEPTED`	a_1	p_1	proposal_id=1, value=42
`ACCEPT`	p_2	a_1	proposal_id=1, value=42
`ACCEPT`	p_2	a_2	proposal_id=1, value=42
`ACCEPT`	p_2	a_3	proposal_id=1, value=42

Tick 17-23

Similar to ticks 6-12 in Example 1. At the end of these ticks, p_2 has determined there is consensus around value 42.

Contents of N at the end of tick 23

m.type	m.src	m.dst	Other attributes
`ACCEPT`	p_1	a_2	proposal_id=1, value=42
`ACCEPT`	p_1	a_3	proposal_id=1, value=42
`ACCEPTED`	a_1	p_1	proposal_id=1, value=42

Ticks 24-25

During these ticks, no messages are delivered. However, the simulation carries on because $t_{\max}$ hasn't been reached, and there are still messages pending delivery in N and future events in E.

Tick 26

In this tick, there is an event specifying that p_1 should recover. This means we can deliver the `ACCEPT` message from p_1 to a_2. However, at this point, a_2 has already promised to not participate in any proposals with a number less than 2. So, it sends a `REJECTED` message back to p_1.

Contents of N at the end of tick 26

m.type	m.src	m.dst	Other attributes
`ACCEPT`	p_1	a_3	proposal_id=1, value=42
`ACCEPTED`	a_1	p_1	proposal_id=1, value=42
`REJECTED`	a_1	p_1	proposal_id=1

Tick 27

Same as tick 27, but with Acceptor a_3.

Contents of N at the end of tick 27

m.type	m.src	m.dst	Other attributes
`ACCEPTED`	a_1	p_1	proposal_id=1, value=42
`REJECTED`	a_2	p_1	proposal_id=1
`REJECTED`	a_3	p_1	proposal_id=1

Tick 28-30

In these ticks, the `ACCEPTED` message from a_1 (which had been delayed since tick 7) and the `REJECTED` messages from a_2 and a_3 are delivered. Confronted with a majority of `REJECTED` messages from the acceptors, p_1 realizes its proposal has failed, and decides to initiate a new proposal for value 42, but using proposal number 3.

Tick 31-42

Similar to ticks 1-12 in Example 1, except that the `PROMISE` messages from the Acceptors will all include the previously accepted proposal (2) and value (42). The algorithm still runs through the `ACCEPT`/`ACCEPTED` phase, but is guaranteed to produce consensus around the same value that p_2 reached consensus on.