Algorithms – Consensus in the Fail-Noisy Model

Algorithm 1 Leaderless Repeatable Paxos - Prepare Phase

Implements:

Consensus, **instance** c.

Uses:

BestEffortBroadcast, **instance** *beb*.

PerfectPointToPointLinks, **instance** *pp2p*.

1: **upon event** $\langle$ *Init* $\rangle$ **do**
2: $decided := false$
3: $promises := \emptyset$
4: $ts := 0$ ▷ logical clock for Paxos rounds
5: $numOfAccepts := 0$
6: $pv := av := \perp$ ▷ Propose and Accept Values
7: $promBallot := accBallot := (0, 0)$

8: **function** PROPOSE
9: **if** $\neg decided$ **then**
10: $ts := ts + 1$
11: $numOfAccepts := 0$
12: $promises := \emptyset$
13: **trigger** $\langle$ *beb*, *Broadcast* | [PREPARE, $(ts,$ RANK$(self))]$ $\rangle$

14: **upon event** $\langle$ *c*, *Propose* | *v* $\rangle$ **do**
15: $pv := v$
16: PROPOSE()

17: **upon event** $\langle$ *pp2p*, *Deliver* | *p*, [PROMISE, *b*, *a*, *v*] $\rangle$ **do**
18: **if** $(ts,$ RANK$(self)) = b$ **then**
19: $promises := promises \cup (a, v)$
20: **if** $\#promises = \frac{(N+1)}{2}$ **then**
21: $(maxBallot, value) :=$ HIGHESTBYBALLOT$(promises)$
22: $pv := value$ *if* $value \neq \perp$ *else* pv
23: **trigger** $\langle$ *beb*, *Broadcast* | [ACCEPT, $(ts,$ RANK$(self)), pv]$ $\rangle$

24: **upon event** $\langle$ *pp2p*, *Deliver* | *p*, [ACCEPTED, *ballot*] $\rangle$ **do**
25: **if** $(ts,$ RANK$(self)) = ballot$ **then**
26: $numOfAccepts := numOfAccepts + 1$
27: **if** $numOfAccepts = \frac{(N+1)}{2}$ **then**
28: **trigger** $\langle$ *beb*, *Broadcast* | [DECIDED, *pv*] $\rangle$

Algorithm 2 Leaderless Repeatable Paxos - Accept and Decide Phases

29: **upon event** $\langle$ *beb*, *Deliver* | *p*, [PREPARE, *ballot*] $\rangle$ **do**
30: **if** *promBallot* $<$ *ballot* **then**
31: *promBallot* := *ballot*
32: **trigger** $\langle$ *pp2p*, *Send* | p, [PROMISE, *promBallot*, *accBallot*, *av*] $\rangle$
33: **else**
34: **trigger** $\langle$ *pp2p*, *Send* | p, [NACK, *ballot*] $\rangle$

35: **upon event** $\langle$ *beb*, *Deliver* | *p*, [ACCEPT, *ballot*, *v*] $\rangle$ **do**
36: **if** *promBallot* $\leq$ *ballot* **then**
37: *promBallot* := *accBallot* := *ballot*
38: *av* := *v*
39: **trigger** $\langle$ *pp2p*, *Send* | p, [ACCEPTED, *ballot*] $\rangle$
40: **else**
41: **trigger** $\langle$ *pp2p*, *Send* | p, [NACK, *ballot*] $\rangle$

42: **upon event** $\langle$ *pp2p*, *Deliver* | *p*, [NACK, *ballot*] $\rangle$ **do**
43: **if** $(ts, \text{RANK}(self)) = ballot$ **then**
44: PROPOSE()

45: **upon event** $\langle$ *beb*, *Deliver* | *p*, [DECIDED, *v*] $\rangle$ **do**
46: **if** $\neg$*decided* **then**
47: **trigger** $\langle$ *c*, *Decide* | *v* $\rangle$
48: *decided* := *true*

Algorithm 1 can eventually terminate when a paxos round is complete, after several failed attempts. To minimise congestion and reach decision in less rounds, one can enforce a backoff strategy so that competing processes wait for increased random time before they attempt to propose again. Algorithm 3 shows an example of such a strategy. Mind that we only show the changes to Algorithm 1, for brevity.

Algorithm 3 Leaderless Repeatable Paxos (with Backoff)

Implements:

Consensus, **instance** c.

Uses:

BestEffortBroadcast, **instance** *beb*.

PerfectPointToPointLinks, **instance** *pp2p*.

```
1: ...

2: upon event ⟨ Init ⟩ do
3:     ...                                  ▷ skipped rest of assignments for brevity
4:     backoffDelay := delay
5:     ...

6: upon event ⟨ pp2p, Deliver | p, [Nack, ballot] ⟩ do
7:     if (ts, Rank(self)) = ballot then
8:         attemptDelay := Random(0, backoffDelay)
9:         startTimer(attemptDelay, AttemptPropose)
10:        backoffDelay := backoffDelay * 2

11: upon event ⟨ Timeout | AttemptPropose ⟩ do
12:    Propose()

13: ...
```

Algorithm 4 Abortable Paxos - Prepare Phase

Implements:

AbortableConsensus, **instance** *ac*.

Uses:

BestEffortBroadcast, **instance** *beb*;
PerfectPointToPointLinks, **instance** *pp2p*.

1: **upon event** $\langle\ ac, Init\ \rangle$ **do**
2: $t := 0$; ▷ logical clock
3: $prepts := 0$; ▷ prepared timestamp
4: $(ats, av) := (0, \bot)$; ▷ timestamp and value accepted
5: $(pts, pv) := (0, \bot)$; ▷ proposer's timestamp and value
6: $readlist := [\bot]^N$;
7: $acks := 0$;

8: **upon event** $\langle\ ac, Propose \mid v\ \rangle$ **do**
9: $t := t + 1$;
10: $pts := t \times N + rank(self)$;
11: $pv := v$;
12: $readlist := [\bot]^N$;
13: $acks := 0$;
14: **trigger** $\langle\ beb, Broadcast \mid [\text{PREPARE}, pts, t]\ \rangle$

15: **upon event** $\langle\ beb, Deliver \mid q, [\text{PREPARE}, ts, t']\ \rangle$ **do**
16: $t := max(t, t') + 1$;
17: **if** $ts < prepts$ **then**
18: **trigger** $\langle\ pp2p, Send \mid q, [\text{NACK}, ts, t]\ \rangle$
19: **else**
20: $prepts := ts$;
21: **trigger** $\langle\ pp2p, Send \mid q, [\text{PREPAREACK}, ats, av, ts, t]\ \rangle$

Algorithm 5 Abortable Paxos: Accept Phase

22: **upon event** $\langle\ pp2p, Deliver \mid q, [\textsc{Nack}, pts', t']\ \rangle$ **do**
23: $\quad t := max(t, t') + 1;$
24: $\quad$ **if** $pts' = pts$ **then**
25: $\quad\quad pts := 0;$
26: $\quad\quad$ **trigger** $\langle\ ac, Abort\ \rangle$

27: **upon event** $\langle\ pp2p, Deliver \mid q, [\textsc{PrepareAck}, ts, v, pts', t']\ \rangle$ **do**
28: $\quad t := max(t, t') + 1;$
29: $\quad$ **if** $pts' = pts$ **then**
30: $\quad\quad readlist[q] := (ts, v);$
31: $\quad\quad$ **if** $\#(readlist) > N/2$ **then**
32: $\quad\quad\quad (ts, v) := highest(readlist);$ $\quad \triangleright$ pair with greatest timestamp
33: $\quad\quad\quad$ **if** $ts \neq 0$ **then**
34: $\quad\quad\quad\quad pv := v;$
35: $\quad\quad\quad readlist := [\perp]^N;$
36: $\quad\quad\quad$ **trigger** $\langle\ beb, Broadcast \mid [\textsc{Accept}, pts, pv, t]\ \rangle$

37: **upon event** $\langle\ beb, Deliver \mid q, [\textsc{Accept}, ts, v, t']\ \rangle$ **do**
38: $\quad t := max(t, t') + 1;$
39: $\quad$ **if** $ts < prepts$ **then**
40: $\quad\quad$ **trigger** $\langle\ pp2p, Send \mid q, [\textsc{Nack}, ts, t]\ \rangle$
41: $\quad$ **else**
42: $\quad\quad ats := prepts := ts;$
43: $\quad\quad av := v;$
44: $\quad\quad$ **trigger** $\langle\ pp2p, Send \mid q, [\textsc{AcceptAck}, ts, t]\ \rangle$

45: **upon event** $\langle\ pp2p, Deliver \mid q, [\textsc{AcceptAck}, pts', t']\ \rangle$ **do**
46: $\quad t := max(t, t') + 1;$
47: $\quad$ **if** $pts' = pts$ **then**
48: $\quad\quad acks := acks + 1;$
49: $\quad\quad$ **if** $acks > N/2$ **then**
50: $\quad\quad\quad pts := 0;$
51: $\quad\quad\quad$ **trigger** $\langle\ ac, Return \mid pv\ \rangle$