

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  $\langle \text{Init} \rangle$  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  $\langle beb, Broadcast \mid [PREPARE, (ts, RANK(self))] \rangle$ 

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

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

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

Algorithm 2 Leaderless Repeatable Paxos - Accept and Decide Phases

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

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

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

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

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  $\langle \textit{Init} \rangle$  do
3:   ...                                 $\triangleright$  skipped rest of assignments for brevity
4:    $\textit{backoffDelay} := \textit{delay}$ 
5:   ...

6: upon event  $\langle \textit{pp2p}, \textit{Deliver} \mid p, [\text{NACK}, \textit{ballot}] \rangle$  do
7:   if  $(\textit{ts}, \text{RANK}(\textit{self})) = \textit{ballot}$  then
8:      $\textit{attemptDelay} := \text{RANDOM}(0, \textit{backoffDelay})$ 
9:      $\textit{startTimer}(\textit{attemptDelay}, \text{ATTEMPTPROPOSE})$ 
10:     $\textit{backoffDelay} := \textit{backoffDelay} * 2$ 

11: upon event  $\langle \textit{Timeout} \mid \text{ATTEMPTPROPOSE} \rangle$  do
12:    $\text{PROPOSE}()$ 

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:    $(ats, av) := (0, \perp);$  ▷ timestamp and value accepted
5:    $(pts, pv) := (0, \perp);$  ▷ proposer's timestamp and value
6:    $readlist := [\perp]^N;$ 
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:   $readlist := [\perp]^N;$ 
13:   $acks := 0;$ 
14:  trigger  $\langle beb, Broadcast \mid [PREPARE, pts, t] \rangle$ 

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

Algorithm 5 Abortable Paxos: Accept Phase

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

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

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

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