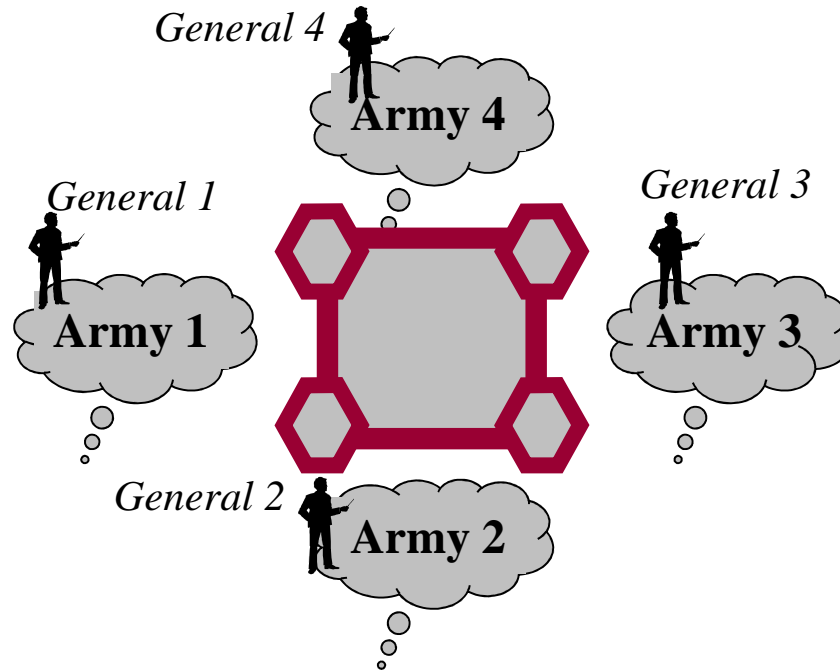


Consensus

Consensus Problem

- ④ A group of processes have to agree on a value that has been proposed by one of them (es. commit/abort of a transaction). It is an abstraction of a class of problems where processes starts with their opinion and then converge on one of them
- ④ It is a fundamental problem
- ④ We study algorithms working on weak models

Consensus Example



Generals have to reach consensus between attack and get back

Impossibility Result

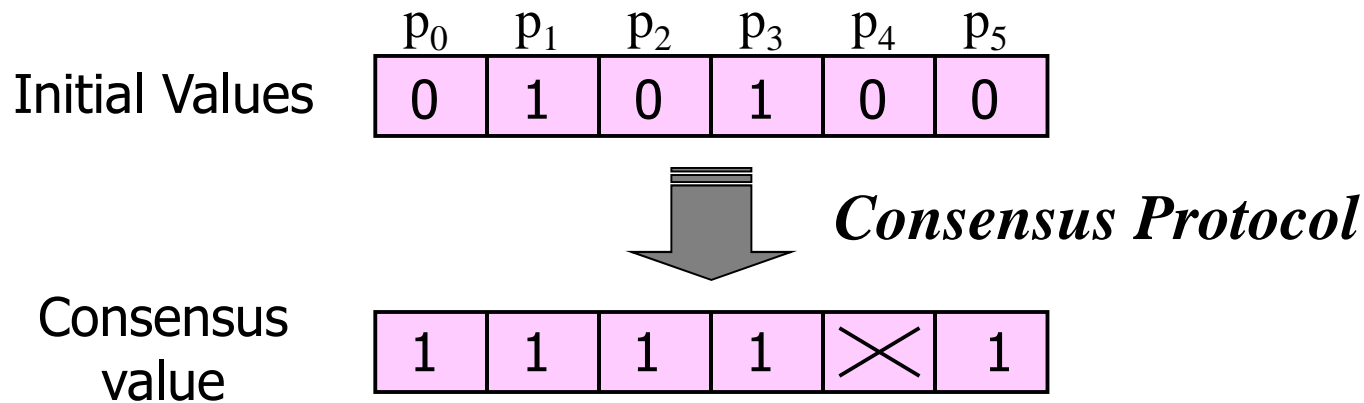


**Impossibility of Consensus
in asynchronous systems
in the presence also of a single crash**

Fisher, Lynch e Patterson (**FLP result**).
Ref: *Journal of the ACM*, Vol. 32, No. 2, April 1985.

Consensus\definition

- ② Set of initial values $\in \{0,1\}$.
- ② Every process has to decide the same value $\in \{0,1\}$ based on their initial proposals.



Module:

Name: (regular) Consensus (c).

Events:

Request: $\langle cPropose, v \rangle$: Used to propose a value for consensus.

Indication: $\langle cDecide, v \rangle$: Used to indicate the decided value for consensus.

Properties:

C1: Termination: Every correct process eventually decides some value.

C2: Validity: If a process decides v , then v was proposed by some process.

C3: Integrity: No process decides twice.

C4: Agreement: No two correct processes decide differently.

A flooding consensus algorithm

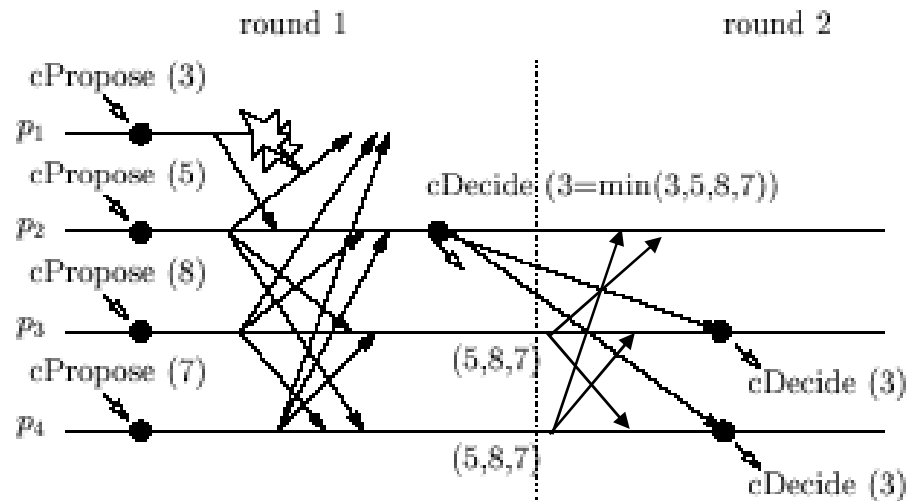


Figure 5.1. Sample execution of the flooding consensus algorithm.

Algorithm 5.1 Flooding Consensus

Implements:

 Consensus (c).

Uses:

 BestEffortBroadcast (beb);

 PerfectFailureDetector (\mathcal{P}).

upon event $\langle Init \rangle$ **do**

 correct := correct-this-round[0] := \perp ;

 decided := \perp ; round := 1;

for $i = 1$ **to** N **do**

 correct-this-round[i] := proposal-set[i] := \emptyset ;

upon event $\langle crash \mid p_i \rangle$ **do**

 correct := correct $\setminus \{p_i\}$;

upon event $\langle cPropose \mid v \rangle$ **do**

 proposal-set[1] := proposal-set[1] $\cup \{v\}$;

 trigger $\langle bebBroadcast \mid [MYSET, 1, proposal-set[1]] \rangle$;

upon event $\langle bebDeliver \mid p_i, [MYSET, r, set] \rangle$ **do**

 correct-this-round[r] := correct-this-round[r] $\cup \{p_i\}$;

 proposal-set[r] := proposal-set[r] $\cup set$;

upon correct \subseteq correct-this-round[round] \wedge (decided = \perp) **do**

 if (correct-this-round[round] = correct-this-round[round-1]) **then**

 decided := \min (proposal-set[round]);

 trigger $\langle cDecide \mid decided \rangle$;

 trigger $\langle bebBroadcast \mid [DECIDED, decided] \rangle$;

else

round := round + 1;

 trigger $\langle bebBroadcast \mid [MYSET, round, proposal-set[round-1]] \rangle$;

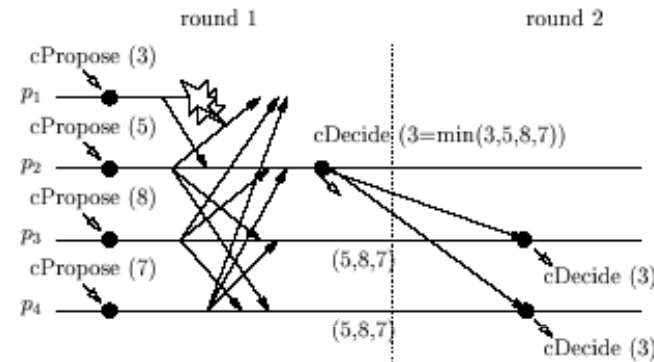
upon event $\langle bebDeliver \mid p_i, [DECIDED, v] \rangle \wedge p_i \in \text{correct} \wedge (\text{decided} = \perp)$ **do**

 decided := v ;

 trigger $\langle cDecide \mid v \rangle$;

 trigger $\langle bebBroadcast \mid [DECIDED, decided] \rangle$;

1. Every message is tagged with the round number in which the message was broadcast. A round terminates at a given process p_i when p_i has received a message from every process that has not been detected to have crashed by p_i in that round. That is, a process does not leave a round unless it receives messages, tagged with that round, from all processes that have not been detected to have crashed.


Figure 5.1. Sample execution of the flooding consensus algorithm.

Correctness and Performance

- Correctness
 - Validity and integrity follow from the properties on the communication channels
 - Termination. At most after N rounds processes decide
 - Agreement. The same deterministic function is applied to the same values by correct processes
- Performance
 - Best Case (No failures). One communication round ($2N^2$)
 - Worst case ($n-1$ failures). N^2 messages exchanged for each communication step

Module:

Name: UniformConsensus (uc).

Events:

$\langle ucPropose, v \rangle$, $\langle ucDecide, v \rangle$: with the same meaning and interface of the consensus interface.

Properties:

C1-C3: from consensus.

C4': *Uniform Agreement*: no two processes decide differently..

Module 5.2 Interface and properties of uniform consensus.

A flooding uniform consensus algorithm

- why the previous algorithm does not impose uniform consensus?
- decide in N steps

Algorithm 5.3 Flooding Uniform Consensus

Implements:

UniformConsensus (uc).

Uses:

BestEffortBroadcast (beb);

PerfectFailureDetector (\mathcal{P}).

upon event $\langle \text{Init} \rangle$ **do**

correct := Π ; round := 1; decided := \perp ; proposal-set := \emptyset ;

for $i = 1$ to N do delivered[i] := \emptyset ;

upon event $\langle \text{crash} \mid p_i \rangle$ **do**

correct := correct \setminus $\{p_i\}$;

upon event $\langle \text{ucPropose} \mid v \rangle$ **do**

proposal-set := proposal-set \cup $\{v\}$;

trigger $\langle \text{bebBroadcast} \mid [\text{MYSET}, 1, \text{proposal-set}] \rangle$;

upon event $\langle \text{bebDeliver} \mid p_i, [\text{MYSET}, r, \text{newSet}] \rangle$ **do**

proposal-set := proposal-set \cup newSet;

delivered[r] := delivered[r] \cup $\{p_i\}$;

upon (correct \subseteq delivered[round]) \wedge (decided = \perp) **do**

if round = N then

decided := min (proposal-set);

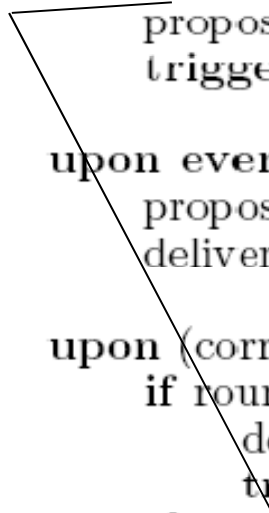
trigger $\langle \text{ucDecide} \mid \text{decided} \rangle$;

else

round := round + 1;

trigger $\langle \text{bebBroadcast} \mid [\text{MYSET}, \text{round}, \text{proposal-set}] \rangle$;

Trigger init



Correctness and Performance

- Correctness
 - Validity and integrity follow from the properties on the communication channels
 - Termination. At the round N non-crashed processes (at most N) decide
 - Agreement. The same deterministic function is applied to the same values by processes that reach round N
- Performance
 - $N \cdot (N-1)^2$ messages for all correct processes to decide

Rotating Coordinator Consensus Algorithm

- ④ Each process has access to a FD $\diamond P$
- ④ Majority of correct processes $n > 2t$
- ④ *rotating coordinator paradigm*, each process knows that during round r the coordinator is $c = (r \bmod n) + 1$. The coordinator p_c **attempts to decide the value. If p_c is correct and it is not suspected by any process, the value will be sent to any process by reliable broadcast and the value will be decided.**
- ④ Each round is split in four steps

Algoritmo con FD \diamond P \implementazione

Step 1: each process sends its current estimation of the decided value (labelled with the current round value, the timestamp) to the coordinator c

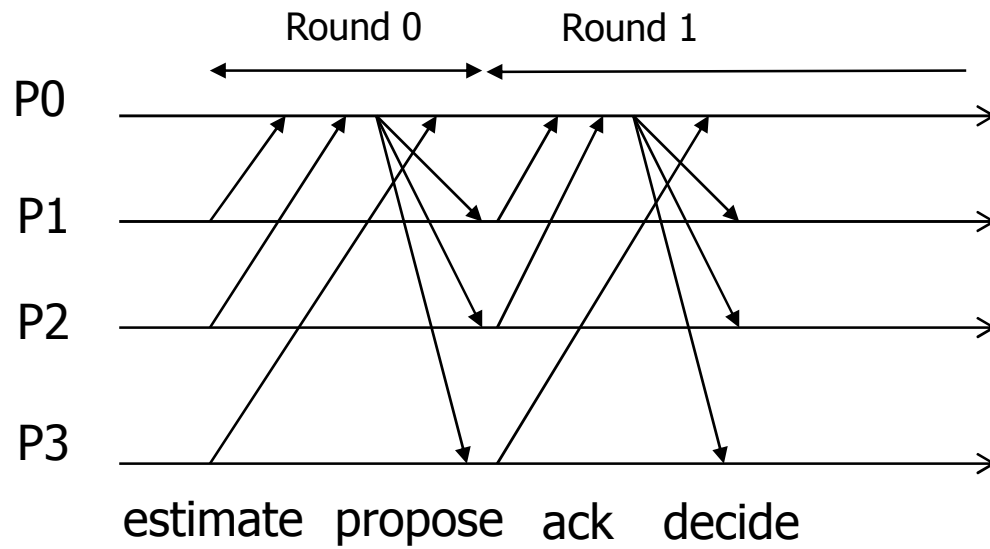
Step 2: the coordinator c gathers a majority of such estimated values, selects the one with the highest timestamp and sends this value to every process as new estimation.

Step 3: for each correct process p two options are valid:

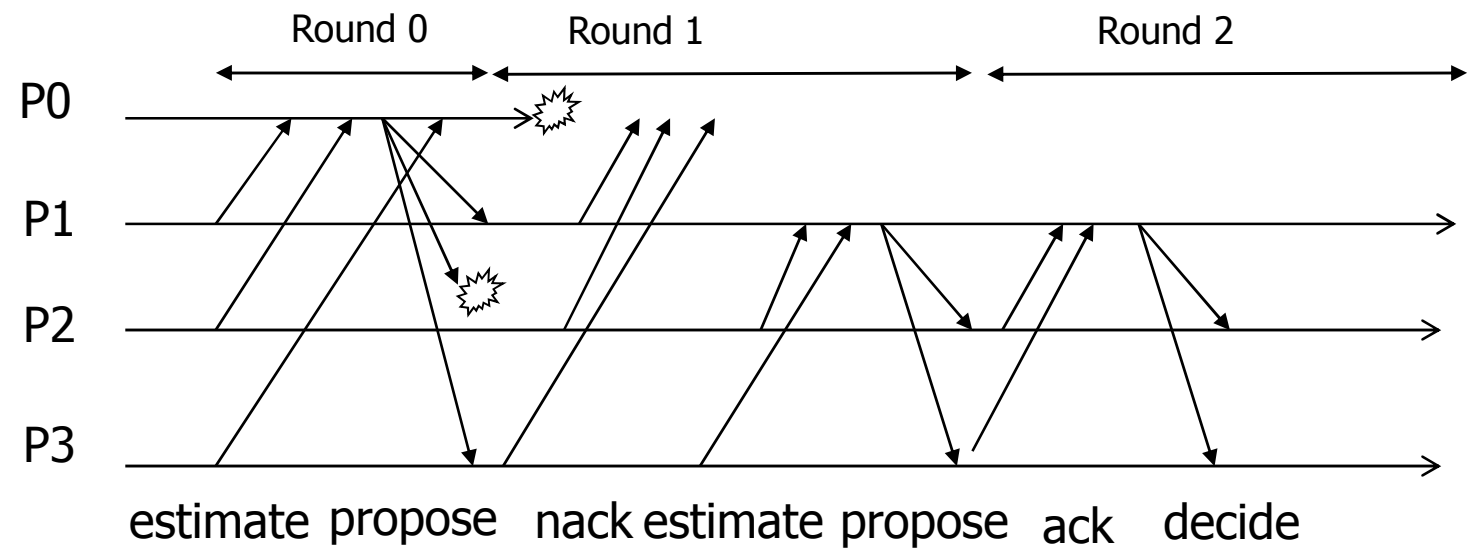
- *p receives the estimated value and sends back an ack message to c to show that p has received and adopted the new estimation; or*
- *p suspects c , then p sends a nack message to c and p goes to the next round*

Step 4: c gathers ack and nack messages.

- *If c receives a majority of acks, c decides the current estimated value as the decided value and c sends this information to all the processes through a reliable broadcast.*
- *If c receives one nack, c goes to the next round*



Run without failures



Run with the failure
Of the coordinator

Algorithm 5.11 Rotating coordinator: leader role.

Uses:

PerfectPointToPointLinks (pp2p);
ReliableBroadcast (rb);
BestEffortBroadcast (beb);
EventuallyPerfectFailureDetector ($\diamond\mathcal{P}$);

function leader (r) **returns** processid is
 return p_i : ($\text{rank}(p_i) = (r \bmod N + 1)$);

upon event $\langle \text{Init} \rangle$ **do**
 $\text{round} := 1$; $\text{proposal} := \text{decided} := \perp$;
 $\text{suspected} := \text{estimate-set}[] := \text{ack-set}[] := \text{nack-set}[] := \emptyset$;
 forall r **do** $\text{estimate}[r] := \text{ack}[r] := \text{false}$; $\text{proposed}[r] := \perp$

upon event $\langle \text{ucPropose} \mid v \rangle$ **do**
 $\text{proposal} := (v, 0)$;

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ESTIMATE}, r, e] \rangle$ **do**
 $\text{estimate-set}[r] := \text{estimate-set}[r] \cup \{e\}$;

upon ($\text{leader}(\text{round}) = \text{self}$) \wedge ($|\text{estimate-set}[\text{round}]| > N/2$) **do**
 $\text{proposal} := \text{highest}(\text{estimate-set}[\text{round}])$;
 trigger $\langle \text{bebBroadcast} \mid [\text{PROPOSE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ACK}, r] \rangle$ **do**
 $\text{ack-set}[r] := \text{ack-set}[r] \cup \{p_i\}$;

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{NACK}, r] \rangle$ **do**
 $\text{nack-set}[r] := \text{nack-set}[r] \cup \{p_i\}$;

upon ($\text{leader}(\text{round}) = \text{self}$) \wedge $\text{nack-set}[\text{round}] \neq \emptyset$ **do**
 $\text{round} := \text{round} + 1$;

upon ($\text{leader}(\text{round}) = \text{self}$) \wedge ($|\text{ack-set}[\text{round}]| > N/2$) **do**
 trigger $\langle \text{rbBroadcast} \mid [\text{DECIDE}, \text{proposal}] \rangle$;

Algorithm 5.12 Rotating coordinator: witness role.

upon event ($\text{proposal} \neq \perp$) \wedge ($\text{estimate}[\text{round}] = \text{false}$) **do**
 $\text{estimate}[\text{round}] := \text{true}$;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ESTIMATE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{bebDeliver} \mid p_i, [\text{PROPOSE}, r, v] \rangle$ **do**
 $\text{proposed}[r] := v$;

upon event ($\text{proposed}[\text{round}] \neq \perp$) \wedge ($\text{ack}[\text{round}] = \text{false}$) **do**
 $\text{proposal} := (\text{proposed}[\text{round}], \text{round})$;
 $\text{ack}[\text{round}] := \text{true}$;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ACK}, \text{round}] \rangle$;
 $\text{round} := \text{round} + 1$;

upon event ($\text{leader}(\text{round}) \in \text{suspected}$) \wedge ($\text{ack}[\text{round}] = \text{false}$) **do**
 $\text{ack}[\text{round}] := \text{true}$;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{NACK}, \text{round}] \rangle$;
 $\text{round} := \text{round} + 1$;

upon event $\langle \text{rbDeliver} \mid p_i, [\text{DECIDED}, v] \rangle$ \wedge ($\text{decided} = \perp$) **do**
 $\text{decided} := v$;
 trigger $\langle \text{ucDecided} \mid v \rangle$;

upon event $\langle \text{suspect} \mid p_i \rangle$ **do**
 $\text{suspected} := \text{suspected} \cup \{p_i\}$;

upon event $\langle \text{restore} \mid p_i \rangle$ **do**
 $\text{suspected} := \text{suspected} \setminus \{p_i\}$;

Algorithm 5.11 Rotating coordinator: leader role.

Uses:

PerfectPointToPointLinks (pp2p);
ReliableBroadcast (rb);
BestEffortBroadcast (beb);
EventuallyPerfectFailureDetector ($\diamond\mathcal{P}$);

function leader (r) **returns** processid is
 return p_i : ($\text{rank}(p_i) = (r \bmod N + 1)$);

upon event $\langle \text{Init} \rangle$ **do**
 round := 1; proposal := decided := \perp ;
 suspected := estimate-set[] := ack-set[] := nack-set[] := \emptyset ;
 forall r **do** estimate[r] := ack[r] := false; proposed[r] := \perp

upon event $\langle \text{ucPropose} \mid v \rangle$ **do**
 proposal := (v,0);

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ESTIMATE}, r, e] \rangle$ **do**
 estimate-set[r] := estimate-set[r] \cup {e};

upon (leader(round)=self) \wedge ($|\text{estimate-set}[\text{round}]| > N/2$) **do**
 proposal := highest(estimate-set[round]);
 trigger $\langle \text{bebBroadcast} \mid [\text{PROPOSE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ACK}, r] \rangle$ **do**
 ack-set[r] := ack-set[r] \cup { p_i };

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{NACK}, r] \rangle$ **do**
 nack-set[r] := nack-set[r] \cup { p_i };

upon (leader(round)=self) \wedge nack-set[round] $\neq \emptyset$ **do**
 round := round + 1;

upon (leader(round)=self) \wedge ($|\text{ack-set}[\text{round}]| > N/2$) **do**
 trigger $\langle \text{rbBroadcast} \mid [\text{DECIDE}, \text{proposal}] \rangle$;

Algorithm 5.12 Rotating coordinator: witness role.

upon event (proposal $\neq \perp$) \wedge (estimate[round] = false) **do**
 estimate[round] := true;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ESTIMATE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{bebDeliver} \mid p_i, [\text{PROPOSE}, r, v] \rangle$ **do**
 proposed[r] := v;

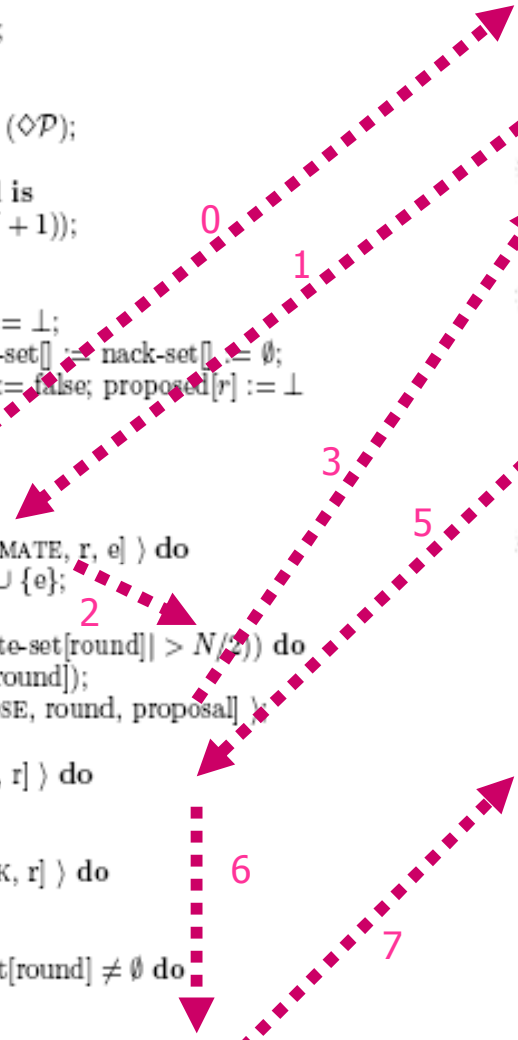
upon event (proposed[round] $\neq \perp$) \wedge (ack[round] = false) **do**
 proposal := (proposed[round], round);
 ack[round] := true;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ACK}, \text{round}] \rangle$;
 round := round + 1;

upon event (leader(round) \in suspected) \wedge (ack[round] = false) **do**
 ack[round] := true;
 trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{NACK}, \text{round}] \rangle$;
 round := round + 1;

upon event $\langle \text{rbDeliver} \mid p_i, [\text{DECIDED}, v] \rangle$ \wedge (decided = \perp) **do**
 decided := v;
 trigger $\langle \text{ucDecided} \mid v \rangle$;

upon event $\langle \text{suspect} \mid p_i \rangle$ **do**
 suspected := suspected \cup { p_i };

upon event $\langle \text{restore} \mid p_i \rangle$ **do**
 suspected := suspected \setminus { p_i };



Algorithm 5.11 Rotating coordinator: leader role.

Uses:

PerfectPointToPointLinks (pp2p);
ReliableBroadcast (rb);
BestEffortBroadcast (beb);
EventuallyPerfectFailureDetector ($\diamond\mathcal{P}$);

function leader (r) returns processid is
return p_i : ($\text{rank}(p_i) = (r \bmod N + 1)$);

upon event $\langle \text{Init} \rangle$ do
round := 1; proposal := decided := \perp ;
suspected := estimate-set[] := ack-set[] := nack-set[] := \emptyset ;
forall r do estimate[r] := ack[r] := false; proposed[r] := \perp

upon event $\langle \text{ucPropose} \mid v \rangle$ do
proposal := (v,0);

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ESTIMATE}, r, e] \rangle$ do
estimate-set[r] := estimate-set[r] \cup {e};

upon (leader(round)=self) \wedge ($|\text{estimate-set}[\text{round}]| > N/2$) do
proposal := highest(estimate-set[round]);
trigger $\langle \text{bebBroadcast} \mid [\text{PROPOSE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{ACK}, r] \rangle$ do
ack-set[r] := ack-set[r] \cup { p_i };

upon event $\langle \text{pp2pDeliver} \mid p_i, [\text{NACK}, r] \rangle$ do
nack-set[r] := nack-set[r] \cup { p_i };

upon (leader(round)=self) \wedge $|\text{nack-set}[\text{round}]| \neq \emptyset$ do
round := round + 1;

upon (leader(round)=self) \wedge ($|\text{ack-set}[\text{round}]| > N/2$) do
trigger $\langle \text{rbBroadcast} \mid [\text{DECIDE}, \text{proposal}] \rangle$;

Algorithm 5.12 Rotating coordinator: witness role.

upon event (proposal $\neq \perp$) \wedge (estimate[round] = false) do
estimate[round] := true;
trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ESTIMATE}, \text{round}, \text{proposal}] \rangle$;

upon event $\langle \text{bebDeliver} \mid p_i, [\text{PROPOSE}, r, v] \rangle$ do
proposed[r] := v;

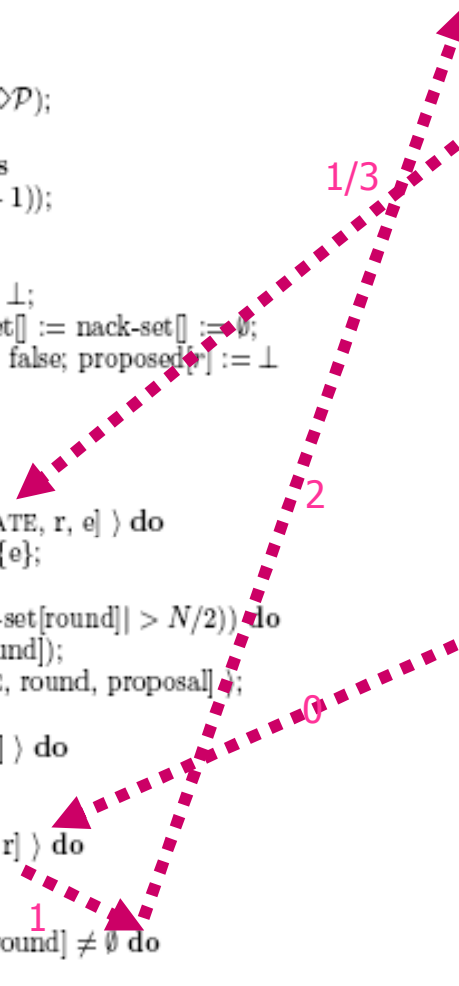
upon event (proposed[round] $\neq \perp$) \wedge (ack[round] = false) do
proposal := (proposed[round], round);
ack[round] := true;
trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{ACK}, \text{round}] \rangle$;
round := round + 1;

upon event (leader(round) \in suspected) \wedge (ack[round] = false) do
ack[round] := true;
trigger $\langle \text{pp2pSend} \mid \text{leader}(\text{round}), [\text{NACK}, \text{round}] \rangle$;
round := round + 1;

upon event $\langle \text{rbDeliver} \mid p_i, [\text{DECIDED}, v] \rangle \wedge$ (decided = \perp) do
decided := v;
trigger $\langle \text{ucDecided} \mid v \rangle$;

upon event $\langle \text{suspect} \mid p_i \rangle$ do
suspected := suspected \cup { p_i };

upon event $\langle \text{restore} \mid p_i \rangle$ do
suspected := suspected \setminus { p_i };



Correctness and Performance

- Correctness
 - Validity and integrity follow from the properties on the communication channels
 - Termination. After time t (when eventual synchrony applies) a correct process is never suspected by other processes and a faulty process is declared faulty. If the algorithm has not terminated before, it will eventually decide and terminate the algorithm by sending the reliable broadcast
 - Agreement. Consider two successive rounds where the coordinators propose v and v' respectively, To decide there is the need of majority of votes, by definition two majority always intersect. This leads to contradiction.
- Performance
 - If no process fails or it is suspected, $4 \cdot (N-1)$ messages for all correct processes to decide

Module:

Name: Non-BlockingAtomicCommit (nbac).

Events:

Request: $\langle \text{nbacPropose} \mid v \rangle$: Used to propose a value for the commit (0 or 1).

Indication: $\langle \text{nbacDecide} \mid v \rangle$: Used to indicate the decided value for nbac.

Properties:

NBAC1: *Uniform Agreement*: No two processes decide different values.

NBAC2: *Integrity*: No process decides two values.

NBAC3: *Abort-Validity*: 0 can only be decided if some process proposes 0 or crashes.

NBAC4: *Commit-Validity*: 1 can only be decided if no process proposes 0.

NBAC5: *Termination*: Every correct process eventually decides.

Module 6.4 Interfaces and properties of NBAC.

Algorithm 6.3 Consensus-Based NBA C.

Implements:

NonBlockingAtomicCommit (nbac).

Uses:

BestEffortBroadcast (beb).

UniformConsensus (uc);

PerfectFailureDetector (\mathcal{P});upon event $\langle \text{Init} \rangle$ dovoted := \emptyset ;correct := Π ;

proposed := false;

upon event $\langle \text{crash} \mid p_i \rangle$ docorrect := correct $\setminus \{p_i\}$;upon event $\langle \text{nbacPropose} \mid v \rangle$ dotrigger $\langle \text{bebBroadcast} \mid v \rangle$;upon event $\langle \text{bebDeliver} \mid p_i, v \rangle$ doif $v = 0 \wedge \neg \text{proposed}$ thentrigger $\langle \text{ucPropose} \mid 0 \rangle$;

proposed := true;

else

voted := voted $\cup \{p_i\}$;upon (correct \setminus voted = $\emptyset \wedge \neg$ proposed) doif correct $\neq \Pi$ thentrigger $\langle \text{ucPropose} \mid 0 \rangle$;

else

trigger $\langle \text{ucPropose} \mid 1 \rangle$;

proposed := true;

upon event $\langle \text{ucDecide} \mid \text{decided} \rangle$ dotrigger $\langle \text{nbacDecide} \mid \text{decided} \rangle$
