Consensus Algorithm for Crash Failures (MP, synchronous)

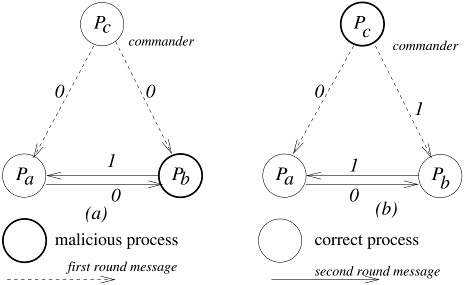
- Up to f $(< n)$ crash failures possible.
- In $f + 1$ rounds, at least one round has no failures.
- Now justify: agreement, validity, termination conditions are satisfied.
- Complexity: $O(f + 1)n^2$ messages
- $f + 1$ is lower bound on number of rounds

```
(global constants)
integer: f;                              // maximum number of crash failures tolerated
(local variables)
integer: x ⟵ local value;

(1) Process P_i (1 ≤ i ≤ n) executes the Consensus algorithm for up to f crash failures:
(1a) for round from 1 to f + 1 do
(1b)      if the current value of x has not been broadcast then
(1c)            broadcast(x);
(1d)      y_j ⟵ value (if any) received from process j in this round;
(1e)      x ⟵ min(x, y_j);
(1f) output x as the consensus value.
```

Upper Bound on Byzantine Processes (sync)

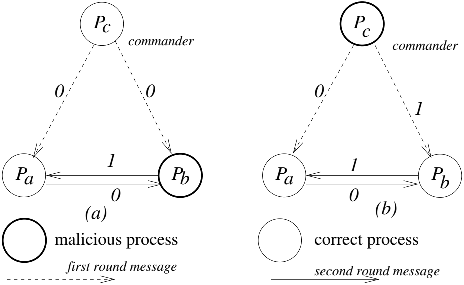
Agreement impossible when $f = 1, n = 3$.



- Taking simple majority decision does not help because loyal commander P_a cannot distinguish between the possible scenarios (a) and (b);
- hence does not know which action to take.
- Proof using induction that problem solvable if $f \leq \lfloor \frac{n-1}{3} \rfloor$. See text.

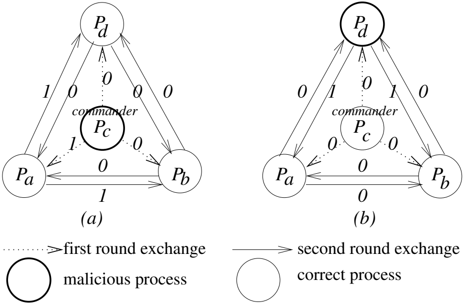
Upper Bound on Byzantine Processes (sync)

Agreement impossible when $f = 1, n = 3$.



- Taking simple majority decision does not help because loyal commander P_a cannot distinguish between the possible scenarios (a) and (b);
- hence does not know which action to take.
- Proof using induction that problem solvable if $f \leq \lfloor \frac{n-1}{3} \rfloor$. See text.

Consensus Solvable when $f = 1, n = 4$



- There is no ambiguity at any loyal commander, when taking majority decision
- Majority decision is over 2nd round messages, and 1st round message received directly from commander-in-chief process.

Byzantine Generals (recursive formulation), (sync, msg-passing)

(variables)
boolean: $v \longleftarrow$ initial value;
integer: $f \longleftarrow$ maximum number of malicious processes, $\leq \lfloor (n-1)/3 \rfloor$;
(message type)
Oral_Msg(v, *Dests*, *List*, *faulty*), where
v is a boolean,
Dests is a set of destination process ids to which the message is sent,
List is a list of process ids traversed by this message, ordered from most recent to earliest,
faulty is an integer indicating the number of malicious processes to be tolerated.

Oral_Msg(f), where $f > 0$:

1. The algorithm is initiated by the Commander, who sends his source value v to all other processes using a *OM*(v, N, $\langle i \rangle$, f) message. The commander returns his own value v and terminates.
2. **[Recursion unfolding:]** For each message of the form *OM*(v_j, *Dests*, *List*, f') received in this round from some process j, the process i uses the value v_j it receives from the source, and using that value, acts as a *new* source. (If no value is received, a default value is assumed.)
   To act as a new source, the process i initiates *Oral_Msg*($f' - 1$), wherein it sends
   OM(v_j, *Dests* $- \{i\}$, *concat*($\langle i \rangle$, L), $(f' - 1)$)
   to destinations not in *concat*($\langle i \rangle$, L)
   in the next round.
3. **[Recursion folding:]** For each message of the form *OM*(v_j, *Dests*, *List*, f') received in Step 2, each process i has computed the agreement value v_k, for each k not in *List* and $k \neq i$,corresponding to the value received from P_k after traversing the nodes in *List*, at one level lower in the recursion. If it receives no value in this round, it uses a default value. Process i then uses the value $majority_{k \notin List, k \neq i}(v_j, v_k)$ as the agreement value and returns it to the next higher level in the recursive invocation.

Oral_Msg(0):

1. **[Recursion unfolding:]** Process acts as a source and sends its value to each other process.
2. **[Recursion folding:]** Each process uses the value it receives from the other sources, and uses that value as the agreement value. If no value is received, a default value is assumed.

Relationship between # Messages and Rounds

round number	a message has already visited	aims to tolerate these many failures	and each message gets sent to	total number of messages in round
1	1	f	$n-1$	$n-1$
2	2	$f-1$	$n-2$	$(n-1)\cdot(n-2)$
...	...	...	...	...
x	x	$(f+1)-x$	$n-x$	$(n-1)(n-2)\ldots(n-x)$
$x+1$	$x+1$	$(f+1)-x-1$	$n-x-1$	$(n-1)(n-2)\ldots(n-x-1)$
$f+1$	$f+1$	0	$n-f-1$	$(n-1)(n-2)\ldots(n-f-1)$

Table: Relationships between messages and rounds in the Oral Messages algorithm for Byzantine agreement.

Complexity: $f+1$ rounds, exponential amount of space, and

$$(n-1)+(n-1)(n-2)+\ldots+(n-1)(n-2)..(n-f-1)\text{messages}$$

Bzantine Generals (iterative formulation), Sync, Msg-passing

(variables)
boolean: $v \longleftarrow$ initial value;
integer: $f \longleftarrow$ maximum number of malicious processes, $\leq \lfloor \frac{n-1}{3} \rfloor$;
tree of boolean:

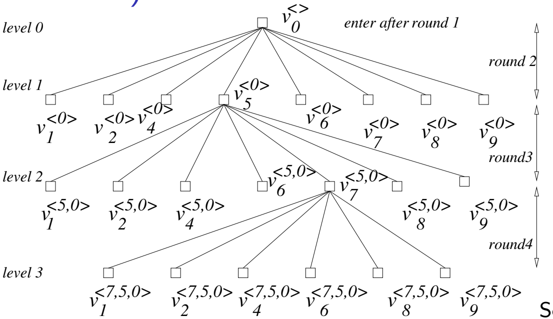
- level 0 root is v^L_{init}, where $L = \langle\rangle$;
- level $h(f \geq h > 0)$ nodes: for each v^L_j at level $h - 1 = sizeof(L)$, its $n - 2 - sizeof(L)$ descendants at level h are $v_k^{concat(\langle j\rangle, L)}$, $\forall k$ such that $k \neq j$, i and k is not a member of list L.

(message type)
OM(v, $Dests$, $List$, $faulty$), where the parameters are as in the recursive formulation.

(1) Initiator (i.e., Commander) initiates Oral Byzantine agreement:
(1a) **send** $OM(v, N - \{i\}, \langle P_i\rangle, f)$ to $N - \{i\}$;
(1b) **return**(v).

(2) (Non-initiator, i.e., Lieutenant) receives Oral Message OM:
(2a) **for** $rnd = 0$ **to** f **do**
(2b) **for** each message OM that arrives in this round, **do**
(2c) **receive** $OM(v, Dests, L = \langle P_{k_1} \ldots P_{k_{f+1-faulty}}\rangle, faulty)$ from P_{k_1};
$//\ faulty + round = f$; $|Dests| + sizeof(L) = n$
(2d) $v^{tail(L)}_{head(L)} \longleftarrow v$; $//\ sizeof(L) + faulty = f + 1$. fill in estimate.
(2e) **send** $OM(v, Dests - \{i\}, \langle P_i, P_{k_1} \ldots P_{k_{f+1-faulty}}\rangle, faulty - 1)$ to $Dests - \{i\}$ **if** $rnd < f$;
(2f) **for** $level = f - 1$ **down to** 0 **do**
(2g) **for** each of the $1 \cdot (n - 2) \cdot \ldots (n - (level + 1))$ nodes v^L_x in level $level$, **do**
(2h) $v^L_x(x \neq i, x \notin L) = majority_{y \notin concat(\langle x\rangle, L); y \neq i}(v^L_x, v_y^{concat(\langle x\rangle, L)})$;

Tree Data Structure for Agreement Problem (Byzantine Generals)



Some branches of the tree at P_3. In this example, $n = 10, f = 3$, commander is P_0.

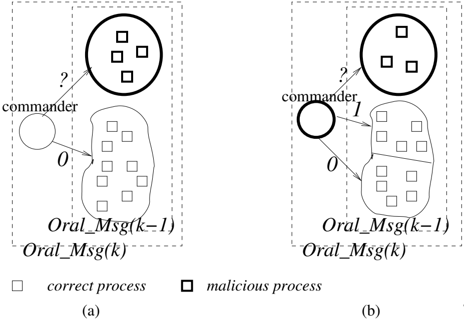
- (round 1) P_0 sends its value to all other processes using *Oral_Msg*(3), including to P_3.
- (round 2) P_3 sends 8 messages to others (excl. P_0 and P_3) using *Oral_Msg*(2). P_3 also receives 8 messages.
- (round 3) P_3 sends $8 \times 7 = 56$ messages to all others using *Oral_Msg*(1); P_3 also receives 56 messages.
- (round 4) P_3 sends $56 \times 6 = 336$ messages to all others using *Oral_Msg*(0); P_3 also receives 336 messages. The received values are used as estimates of the majority function at this level of recursion.

Exponential Algorithm: An example

An example of the majority computation is as follows.

- P_3 revises its estimate of $v_7^{\langle 5,0\rangle}$ by taking $majority(v_7^{\langle 5,0\rangle}, v_1^{\langle 7,5,0\rangle}, v_2^{\langle 7,5,0\rangle}, v_4^{\langle 7,5,0\rangle}, v_6^{\langle 7,5,0\rangle}, v_8^{\langle 7,5,0\rangle}, v_9^{\langle 7,5,0\rangle})$. Similarly for the other nodes at level 2 of the tree.
- P_3 revises its estimate of $v_5^{\langle 0\rangle}$ by taking $majority(v_5^{\langle 0\rangle}, v_1^{\langle 5,0\rangle}, v_2^{\langle 5,0\rangle}, v_4^{\langle 5,0\rangle}, v_6^{\langle 5,0\rangle}, v_7^{\langle 5,0\rangle}, v_8^{\langle 5,0\rangle}, v_9^{\langle 5,0\rangle})$. Similarly for the other nodes at level 1 of the tree.
- P_3 revises its estimate of $v_0^{\langle\rangle}$ by taking $majority(v_0^{\langle\rangle}, v_1^{\langle 0\rangle}, v_2^{\langle 0\rangle}, v_4^{\langle 0\rangle}, v_5^{\langle 0\rangle}, v_6^{\langle 0\rangle}, v_7^{\langle 0\rangle}, v_8^{\langle 0\rangle}, v_9^{\langle 0\rangle})$. This is the consensus value.

Impact of a Loyal and of a Disloyal Commander



The effects of a loyal or a disloyal commander in a system with $n = 14$ and $f = 4$. The subsystems that need to tolerate k and $k - 1$ traitors are shown for two cases. (a) Loyal commander. (b) No assumptions about commander.

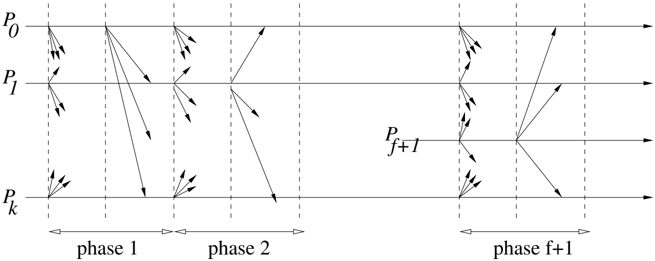
(a) the commander who invokes *Oral_Msg(x)* is loyal, so all the loyal processes have the same estimate. Although the subsystem of $3x$ processes has x malicious processes, all the loyal processes have the same view to begin with. Even if this case repeats for each nested invocation of *Oral_Msg*, even after x rounds, among the processes, the loyal processes are in a simple majority, so the majority function works in having them maintain the same common view of the loyal commander's value.

(b) the commander who invokes *Oral_Msg(x)* may be malicious and can send conflicting values to the loyal processes. The subsystem of $3x$ processes has $x - 1$ malicious processes, but all the loyal processes do not have the same view to begin with.

The Phase King Algorithm

Operation

- Each phase has a unique "phase king" derived, say, from PID.
- Each phase has two rounds:
  1. in 1st round, each process sends its estimate to all other processes.
  2. in 2nd round, the "Phase king" process arrives at an estimate based on the values it received in 1st round, and broadcasts its new estimate to all others.



The Phase King Algorithm: Code

(variables)
boolean: $v \longleftarrow$ initial value;
integer: $f \longleftarrow$ maximum number of malicious processes, $f < \lceil n/4 \rceil$;

(1) Each process executes the following $f+1$ phases, where $f < n/4$:
(1a) **for** $phase = 1$ **to** $f+1$ **do**
(1b) Execute the following Round 1 actions: // actions in round one of each phase
(1c) **broadcast** v to all processes;
(1d) **await** value v_j from each process P_j;
(1e) *majority* $\longleftarrow$ the value among the v_j that occurs $> n/2$ times (default if no maj.);
(1f) *mult* $\longleftarrow$ number of times that *majority* occurs;
(1g) Execute the following Round 2 actions: // actions in round two of each phase
(1h) **if** $i = phase$ **then** // only the phase leader executes this send step
(1i) **broadcast** *majority* to all processes;
(1j) **receive** *tiebreaker* from P_{phase} (default value if nothing is received);
(1k) **if** $mult > n/2 + f$ **then**
(1l) $v \longleftarrow majority$;
(1m) **else** $v \longleftarrow tiebreaker$;
(1n) **if** $phase = f+1$ **then**
(1o) output decision value v.

The Phase King Algorithm

- $(f+1)$ phases, $(f+1)[(n-1)(n+1)]$ messages, and can tolerate up to $f < \lceil n/4 \rceil$ malicious processes

Correctness Argument

1. Among $f+1$ phases, at least one phase k where phase-king is non-malicious.
2. In phase k, all non-malicious processes P_i and P_j will have same estimate of consensus value as P_k does.
    1. P_i and P_j use their own majority values (Hint: $\Longrightarrow P_i$'s $mult > n/2 + f$)
    2. P_i uses its majority value; P_j uses phase-king's tie-breaker value. (Hint: P_i"s $mult > n/2 + f$, P_j's $mult > n/2$ for same value)
    3. P_i and P_j use the phase-king's tie-breaker value. (Hint: In the phase in which P_k is non-malicious, it sends same value to P_i and P_j)

    In all 3 cases, argue that P_i and P_j end up with same value as estimate
3. If all non-malicious processes have the value x at the start of a phase, they will continue to have x as the consensus value at the end of the phase.