Lecture 31

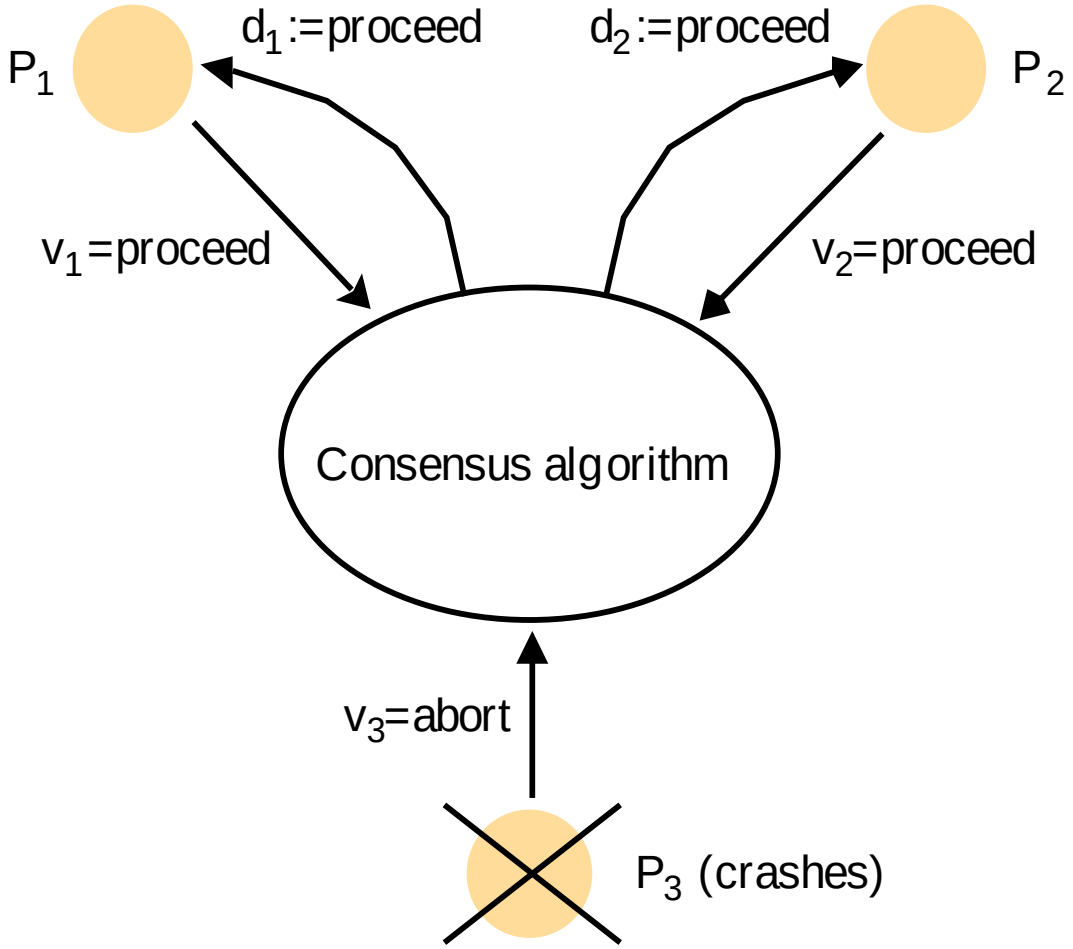
- Administration

Concensus

Consensus problem

- Every process pi begins in the undecided state and propose a single value vi, drawn from a set D. The processes communicate with one another, exchanging values. Each process then sets the value of a decision variable di. In doing so it enters the decided state, in which it may no longer change di.
- Requirements:
  - Termination: Eventually each correct process sets its decision variable.
  - Agreement: The decision value of all correct processes is the same: if pi and pj are correct and have entered the decided state, then di=dj
  - Integrity: if the correct processes all proposed the same value, then any correct process in the decided state has chosen that value. This condition can be loosen. For example, not necessarily all of them, may be some of them.

Consensus for three processes



Byzantine general problem (proposed in 1982)

- Three or more generals are to agree to attack or to retreat. One, the commander, issues the order. The others, lieutenants to the commander, are to decide to attack or retreat.
- But one or more of the general may be treacherous-that is, faulty. If the commander is treacherous, he proposes attacking to one general and retreating to another. If a lieutenant is treacherous, he tells one of his peers that the commander told him to attack and another that they are to retreat.

Byzantine general problem

- **Byzantine general problem** is different from consensus in that a distinguished process supplies a value that the others are to agree upon, instead of each of them proposing a value.
- Requirements:
  - Termination: eventually each correct process sets its decision variable.
  - Agreement: the decision value of all correct processes is the same.
  - Integrity: If the commander is correct, then all correct processes decide on the value that the commander proposed.

  If the commander is correct, the integrity implies agreement; but the commander need not be correct.

Interactive consistency

- **Interactive consistency problem :** Another variant of consensus, in which every process proposes a single value. Goal of this algorithm is for the correct processes to agree on a decision vector of values, one for each process.
- Requirements:
  - **Termination**: eventually each correct process sets it decision variable.
  - **Agreement**: the decision vector of all correct processes is the same.
  - **Integrity**: If pi is correct, then all correct processes decide on vi as the ith component of the vector.

All equivalent

- BG → IC
- IC → C
- C → BG

Reliable Total Order also solves it --- again NOT possible in an asynchronous system

Synchronous Systems

Consensus in a synchronous system

- Basic multicast protocol assuming up to f of the N processes exhibit crash failures.

- Each correct process collects proposed values from the other processes. This algorithm proceeds in f+1 rounds, in each of which the correct processes Basic-multicast the values between themselves.
  - At most f processes may crash, by assumption.
  - At worst, all f crashes during the round, but the algorithm guarantees that at the end of the rounds all the correct processes that have survived have the same final set of values are in a position to agree.

Consensus in a synchronous

Algorithm for process $p_i \in g$; algorithm proceeds in $f + 1$ rounds

On initialization

$Values_i^1 := \{v_i\}$; $Values_i^0 = \{\}$;

In round r $(1 \le r \le f + 1)$

B-multicast$(g, Values_i^r - Values_i^{r-1})$; // Send only values that have not been sent

$Values_i^{r+1} := Values_i^r$;

while (in round r)

{

On B-deliver(V_j) *from some* p_j

$Values_i^{r+1} := Values_i^{r+1} \cup V_j$;

}

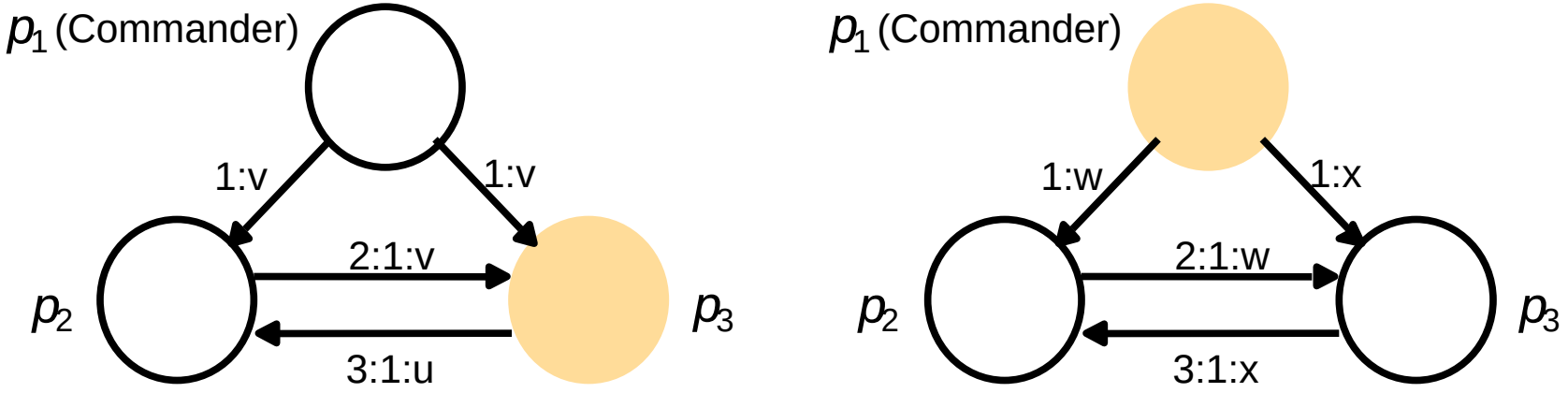
After $(f + 1)$ *rounds*

Assign $d_i = minimum(Values_i^{f+1})$;

At most f crashes can occur, and there are f+1 rounds. So we can compensate up to f crashes.

Any algorithm to reach consensus despite up to f crash failures requires at least f+1 rounds of message exchanges, no matter how it is constructed.

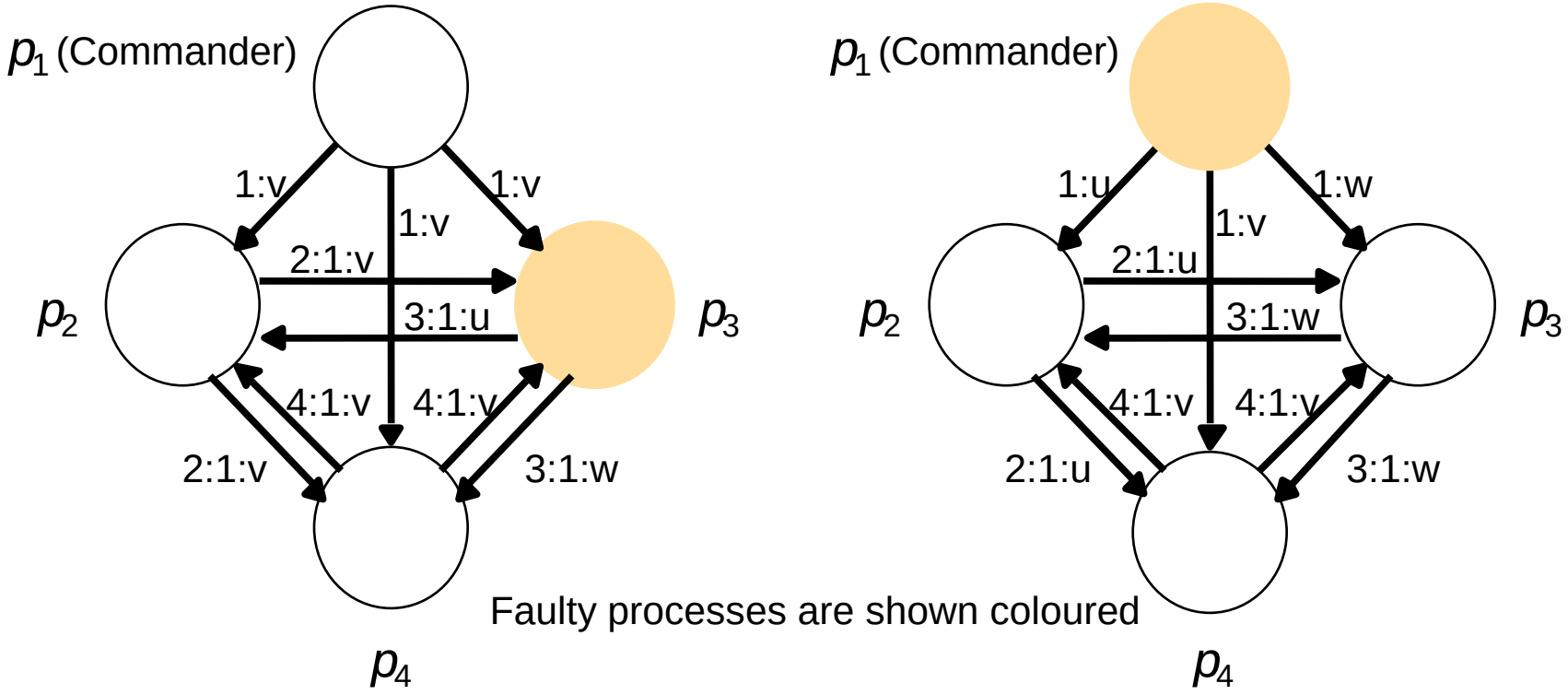
Three byzantine general problem



Faulty processes are shown coloured

3:1:u: first number indicates source, the second number indicates Who says. From P3, P1 says u.
If solution exists, P2 bound to decide on v when commander is correct. If no solution can distinguish between correct and faulty commander, p2 must also choose the value sent by commander. By Symmetry, P3 should also choose commander, p2 does the same thing. But it contradicts with agreement.
No solution if N<=3f. All because a correct general can not tell which process is faulty.

Four byzantine generals



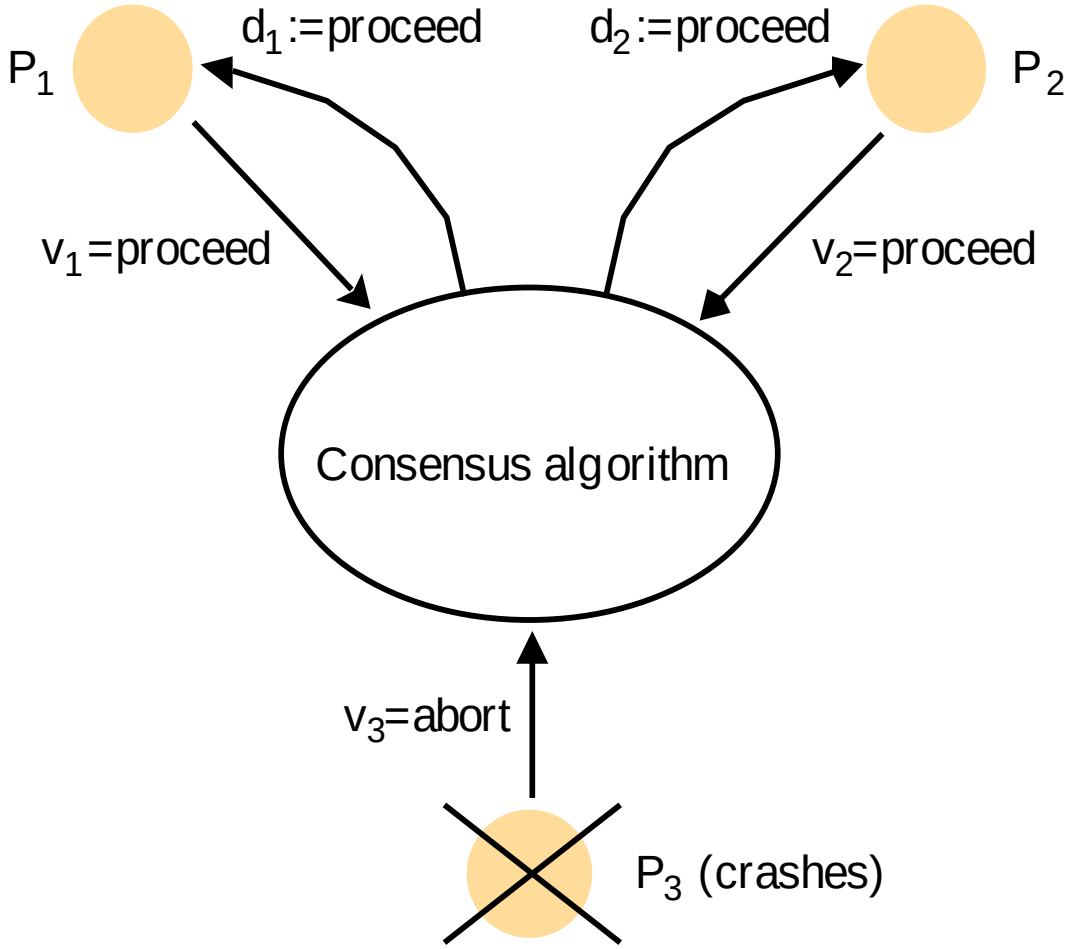
Faulty processes are shown coloured

First round: the commander sends a value to each of the lieutenants.

Second round: each of the lieutenants sends the value it received to its peers.

A lieutenant receives a value from the commander, plus N-2 values from its peers. Lieutenant just applies a simple majority function to the set of values it receives. The faulty process may omit to send a value. If timeouts, the receiver just set null as received value.

Consensus for three processes



Consensus in a synchronous system

- Basic multicast protocol assuming up to f of the N processes exhibit crash failures.

- Each correct process collects proposed values from the other processes. This algorithm proceeds in f+1 rounds, in each of which the correct processes Basic-multicast the values between themselves.
  - At most f processes may crash, by assumption.
  - At worst, all f crashes during the round, but the algorithm guarantees that at the end of the rounds all the correct processes that have survived have the same final set of values are in a position to agree.

Consensus in a synchronous

Algorithm for process $p_i \in g$; algorithm proceeds in $f + 1$ rounds

On initialization

$Values_i^1 := \{v_i\}$; $Values_i^0 = \{\}$;

In round r $(1 \le r \le f + 1)$

B-multicast$(g, Values_i^r - Values_i^{r-1})$; // Send only values that have not been sent

$Values_i^{r+1} := Values_i^r$;

while (in round r)

{

On B-deliver(V_j) *from some* p_j

$Values_i^{r+1} := Values_i^{r+1} \cup V_j$;

}

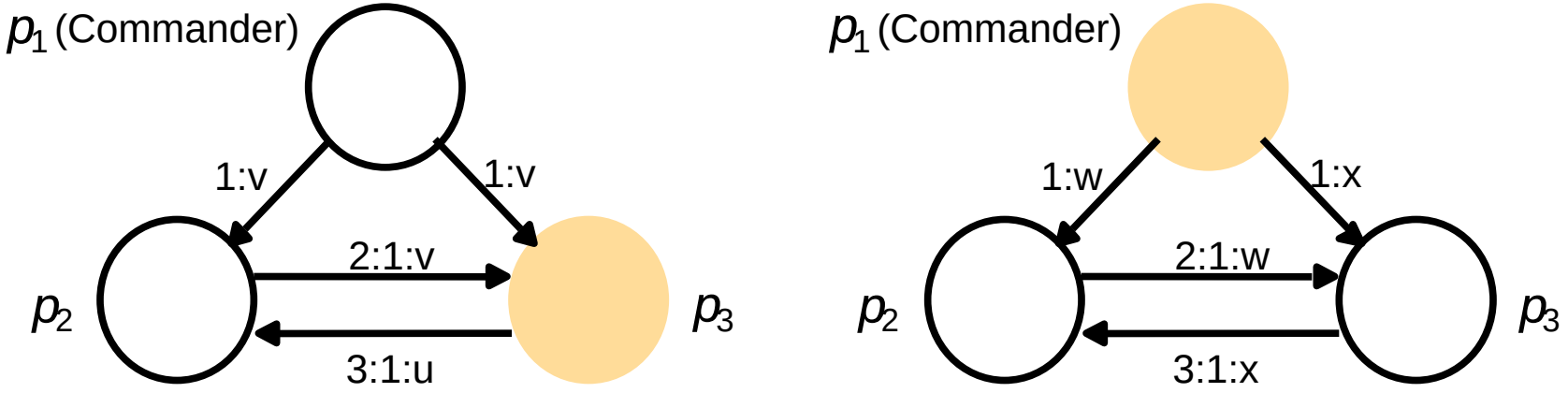
After $(f + 1)$ *rounds*

Assign $d_i = minimum(Values_i^{f+1})$;

At most f crashes can occur, and there are f+1 rounds. So we can compensate up to f crashes.

Any algorithm to reach consensus despite up to f crash failures requires at least f+1 rounds of message exchanges, no matter how it is constructed.

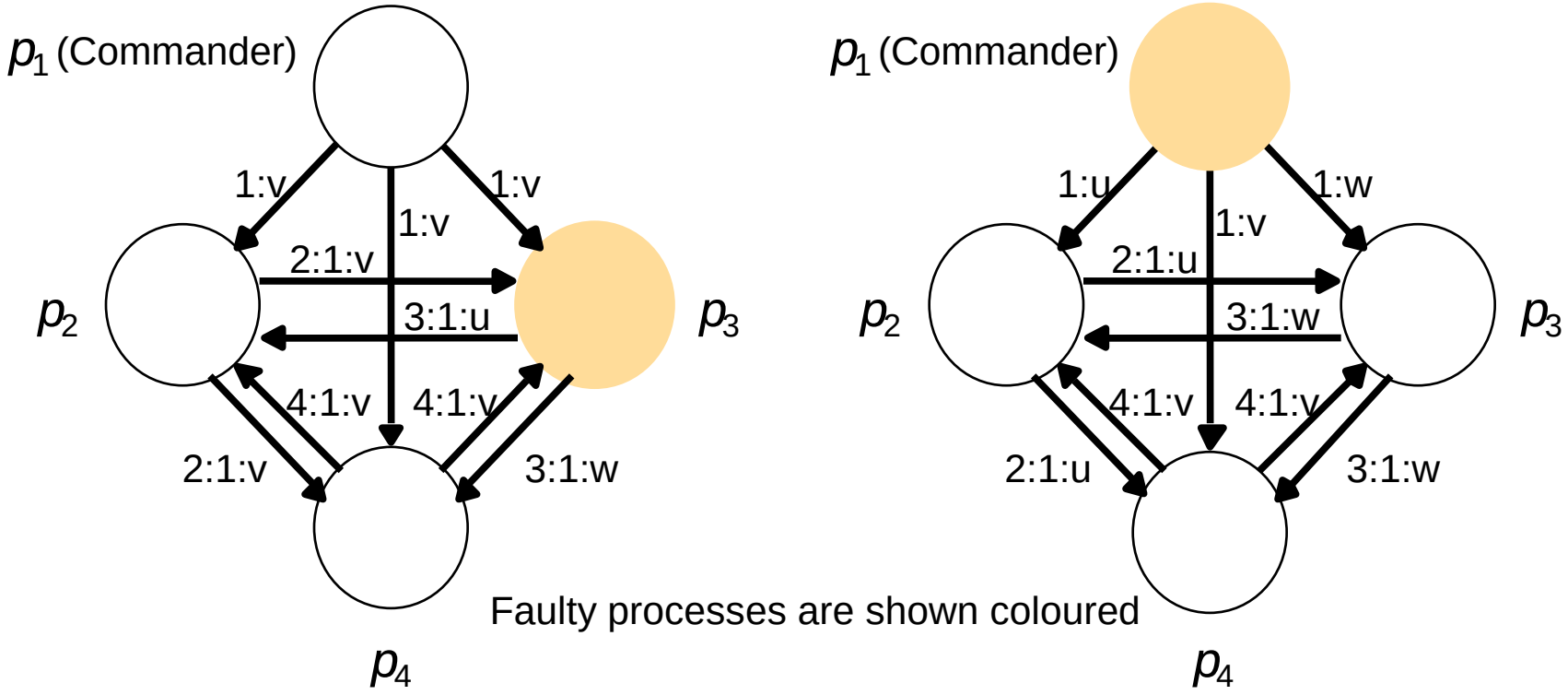
Three byzantine general problem



Faulty processes are shown coloured

3:1:u: first number indicates source, the second number indicates Who says. From P3, P1 says u.
If solution exists, P2 bound to decide on v when commander is correct. If no solution can distinguish between correct and faulty commander, p2 must also choose the value sent by commander. By Symmetry, P3 should also choose commander, p2 does the same thing. But it contradicts with agreement.
No solution if N<=3f. All because a correct general can not tell which process is faulty.

Four byzantine generals



Faulty processes are shown coloured

First round: the commander sends a value to each of the lieutenants.

Second round: each of the lieutenants sends the value it received to its peers.

A lieutenant receives a value from the commander, plus N-2 values from its peers. Lieutenant just applies a simple majority function to the set of values it receives. The faulty process may omit to send a value. If timeouts, the receiver just set null as received value.