06|12|23

Dependable Distributed Systems Master of Science in Engineering in Computer Science

AA 2023/2024

LECTURE 29: CONSENSUS IN PRESENCE OF BYZANTINE PROCESSES

Byzantine Tolerant Consensus

Ideally, we would like to obtain the same properties we get in the crash prone environment

- *Termination*: Every correct process eventually decides some value
- *Validity*: If a process decides *v*, then *v* was proposed by some process
- *Integrity*: No process decides twice
- *Agreement*: No two correct processes decide differently

However…

- We cannot require anything from Byzantine
- The validity property must be adapted as Byzantine processes may invent values or claim to have proposed different values

Thus…

- We restrict the specification only to correct processes
- We define two different versions of validity (weak and strong)

Weak Byzantine Consensus Specification

Module 5.10: Interface and properties of weak Byzantine consensus

⚠ NOTE: Weak Validity allows to decide an arbitrary value if some process is Byzantine.

Properties:

WBC1: *Termination:* Every correct process eventually decides some value.

WBC2: *Weak validity:* If all processes are correct and propose the same value v, then no correct process decides a value different from v; furthermore, if all processes are correct and some process decides v, then v was proposed by some process.

WBC3: *Integrity:* No correct process decides twice.

WBC4: *Agreement:* No two correct processes decide differently.

Strong Byzantine Consensus

Module 5.11: Interface and properties of (strong) Byzantine consensus

Module:

Name: ByzantineConsensus, **instance** *bc*.

Events:

Request: $\langle$ *bc*, *Propose* | v $\rangle$: Proposes value v for consensus.

Indication: $\langle$ *bc*, *Decide* | v $\rangle$: Outputs a decided value v of consensus.

Properties:

BC1 and **BC3–BC4:** Same as properties WBC1 and WBC3–WBC4 in weak Byzantine consensus (Module 5.10).

BC2: *Strong validity:* If all correct processes propose the same value v, then no correct process decides a value different from v; otherwise, a correct process may only decide a value that was proposed by some correct process or the special value $\square$.

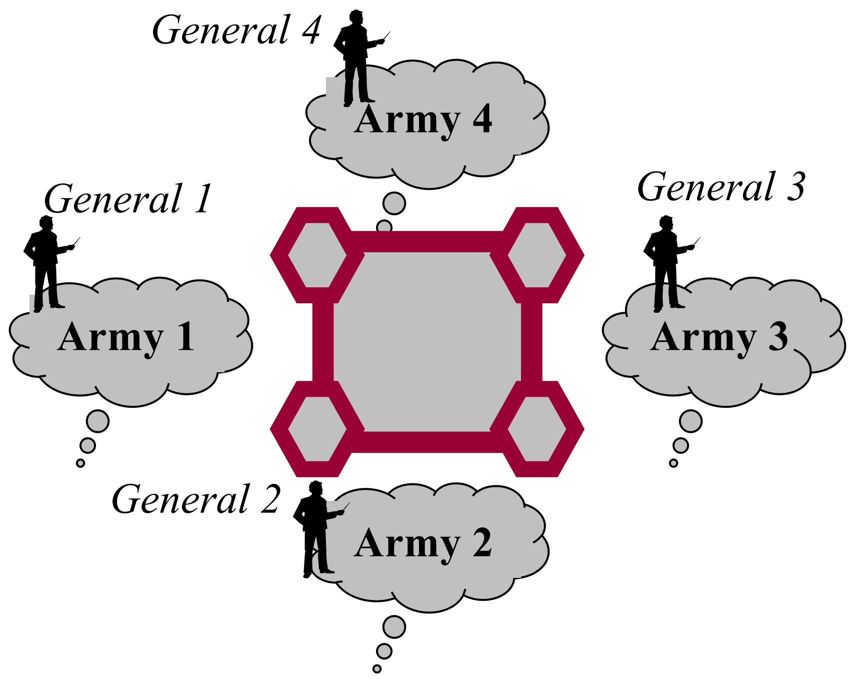
$\uparrow$ predefined DEFAULT value

$\nearrow$ change the set of admissible value

The Byzantine Generals Problem

Abstract the consensus problem in presence of Byzantine processes

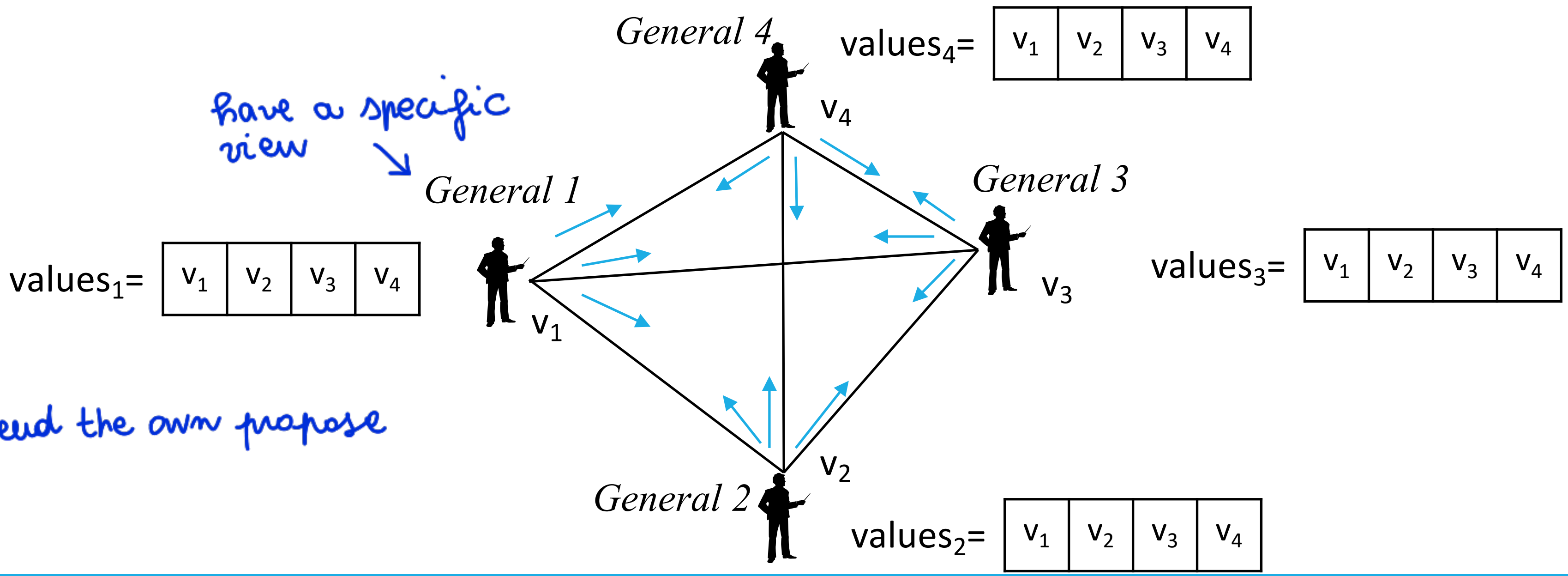
- several divisions of the Byzantine army are camped outside of an enemy city
- generals at the head of each division can communicate by messengers
- observing the enemy each general can propose to attack or retreat
- some generals are traitors
- the army wins only if all loyal generals attack or retreat



An Intuitive Algorithm

1. Each general starts with its own value v(i)
2. v(i) must be communicated by the i-th general to others
3. Each general uses some method for combining the values v(1)..... v(n) into a single plan of action, where n is the number of generals
   - We need to define a function f(values$_i$)

have a specific view


General 4
values$_4$= | v_1 | v_2 | v_3 | v_4
v_4
General 1
values$_1$= | v_1 | v_2 | v_3 | v_4
v_1
General 3
v_3
values$_3$= | v_1 | v_2 | v_3 | v_4
v_2
General 2
values$_2$= | v_1 | v_2 | v_3 | v_4

1) send the own propose
2)

The Byzantine Generals Problem

Goal: f(values_i) must be defined in a way that:

What does it mean "bad plan"?

A. all loyal generals decide upon the same plan of action

B. a small number of traitors cannot cause the loyal generals to adopt a bad plan

An Intuitive Algorithm

OBSERVATIONS

- Condition A is achieved by having all generals use the same method for combining the information
  - f($values_i$) must be deterministic

- Condition B is achieved by using a robust method
  - E.g., if the only decision to be made is whether to attack or retreat, then v(i) can be General i's opinion of which option is best, and the final decision can be based upon a majority vote among them.

The Byzantine Generals Problem

Each general *i* sends its opinion represented by the value *v(i)* to all

Rephrasing the goal:

1. Every loyal general must obtain the same information *v(1),…,v(n)*
2. If the *i*-th general is loyal, then the value that he sends must be used by every loyal general as the value of *v(i)*

Therefore 1. can be rephrased:

1. Any two loyal generals use the same value of *v(i)*

The Byzantine Generals Problem

Both conditions 1 and 2 are expressed on the single value sent by the i-th general. Therefore, we can restrict the problem to how a general communicate its value to loyal generals.

Byzantine Generals Problem: a commanding general must send his order to n-1 lieutenant generals such that:

Interactive Consistency

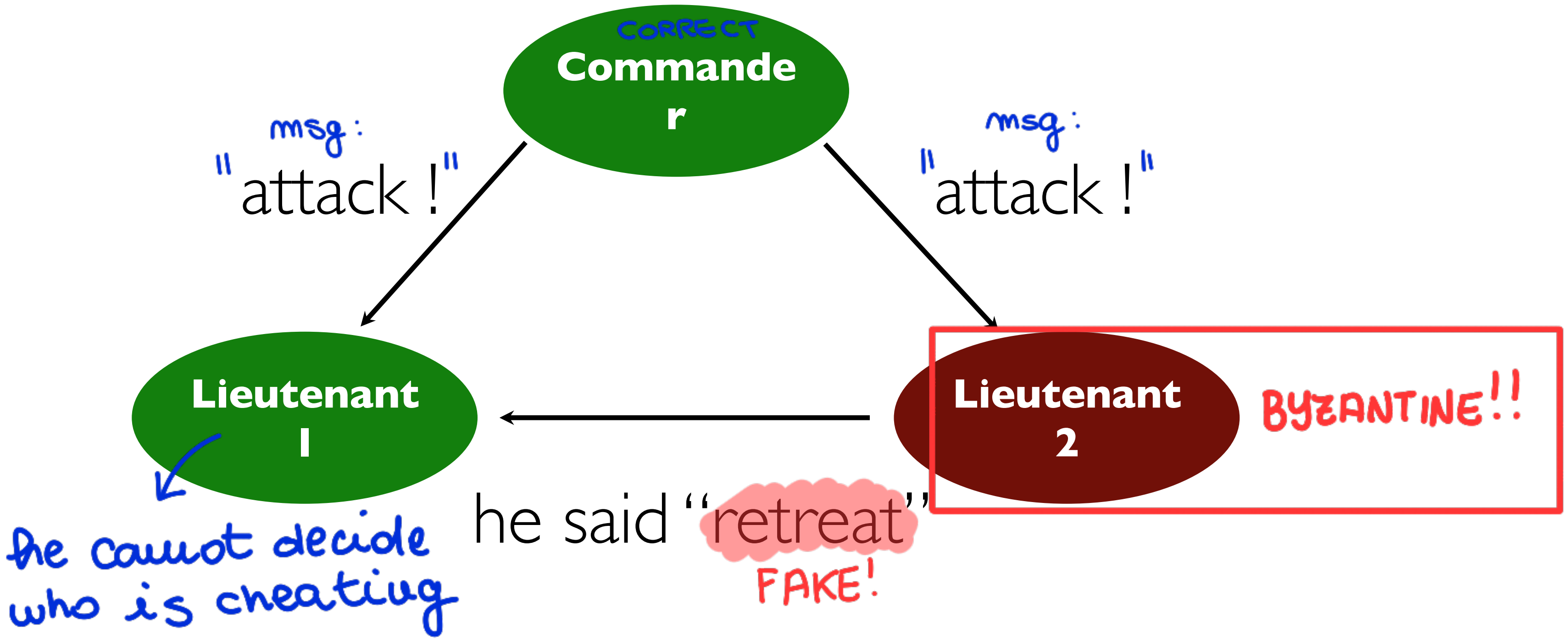
tenente

- All loyal lieutenants obey the same order (IC1)
- If the commanding general is loyal, then every loyal general obeys the order he sends (IC2)
- The order is "Use *v(i)* as my value".

↳ master-slave approach

The Byzantine Generals Problem

Impossibility result: if the generals can send only plain text messages no solution will work unless more than two-thirds of the generals are loyal.



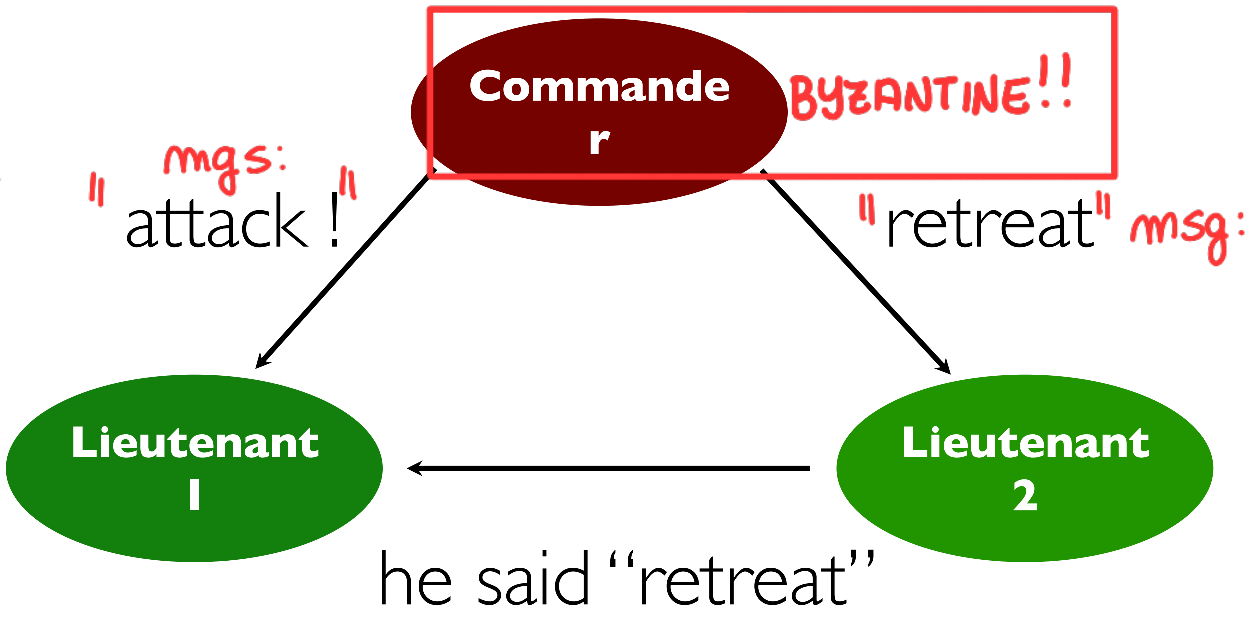
The Byzantine Generals Problem

Impossibility result: if the generals can send only plain text messages no solution will work unless more than two-thirds of the generals are loyal.

to achive CONSENSUS

↓

$N \geq 3f+1$



The Byzantine Generals Problem

This abstract problem captures many of the issues we face if we want to implement a RSM in a byzantine failure model:

- How can correct replicas agree on a common order for client requests ?
- How can correct replicas decide what is the correct answer to a client request ?
- How can correct replicas maintain their state consistent ?

Byzantine reliable consensus

System model:

- Up to f processes can be Byzantine
- $N \geq 3f + 1$ # of processes
- The "Oral Message" communication model is assumed → auth. channel : integrity of msg
  - Every Message that is sent is delivered correctly P2PL
  - Message source is known to the receiver
  - Message omissions can be detected

The default decision for Lieutenants is RETREAT
↓
we have a default value

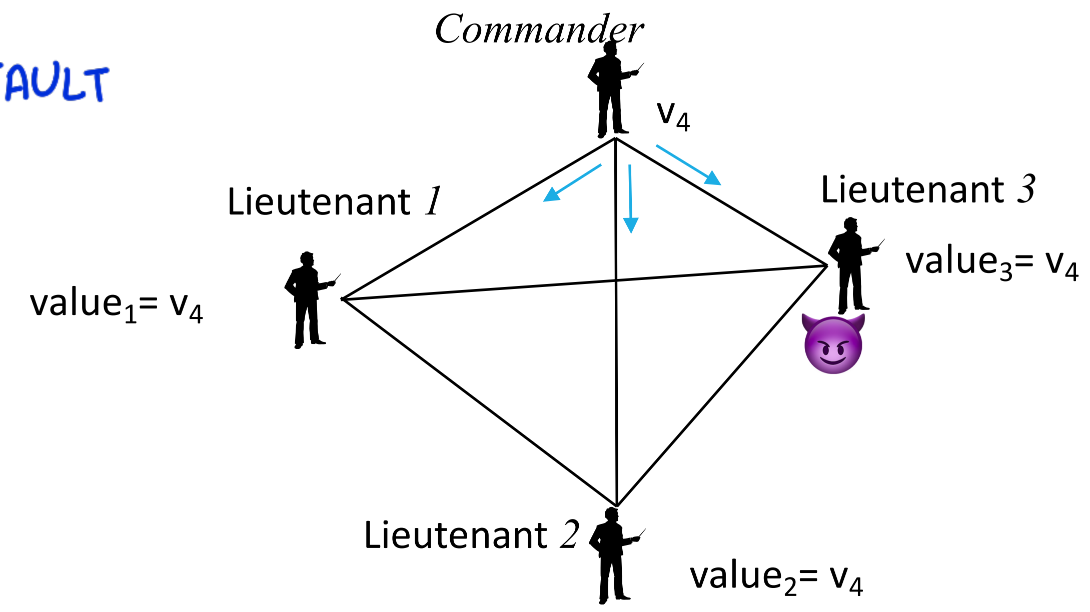
Implementing the "Oral Message" communication abstraction

We define inductively a set of protocols OM(f): # of Byzantine

OM(0)

1. The commander sends his value to every lieutenant.
2. Each lieutenant uses the value he receives from the commander, or uses the value $\boxed{\text{RETREAT}}$ (i.e., $\perp$) if he receives no value. (set a timer)

↓ DEFAULT



Byzantine reliable consensus

We define inductively a set of protocols OM(*f*):

I have a new lieu. for each round

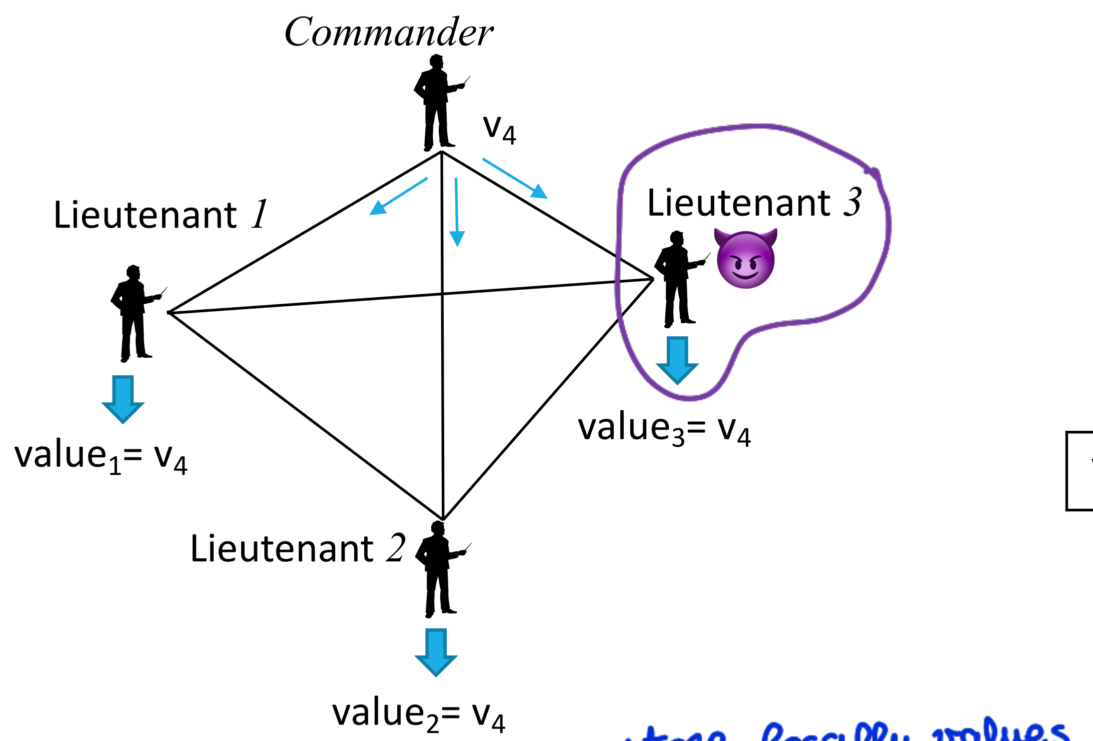
↑

OM(*f*) with *f*>0

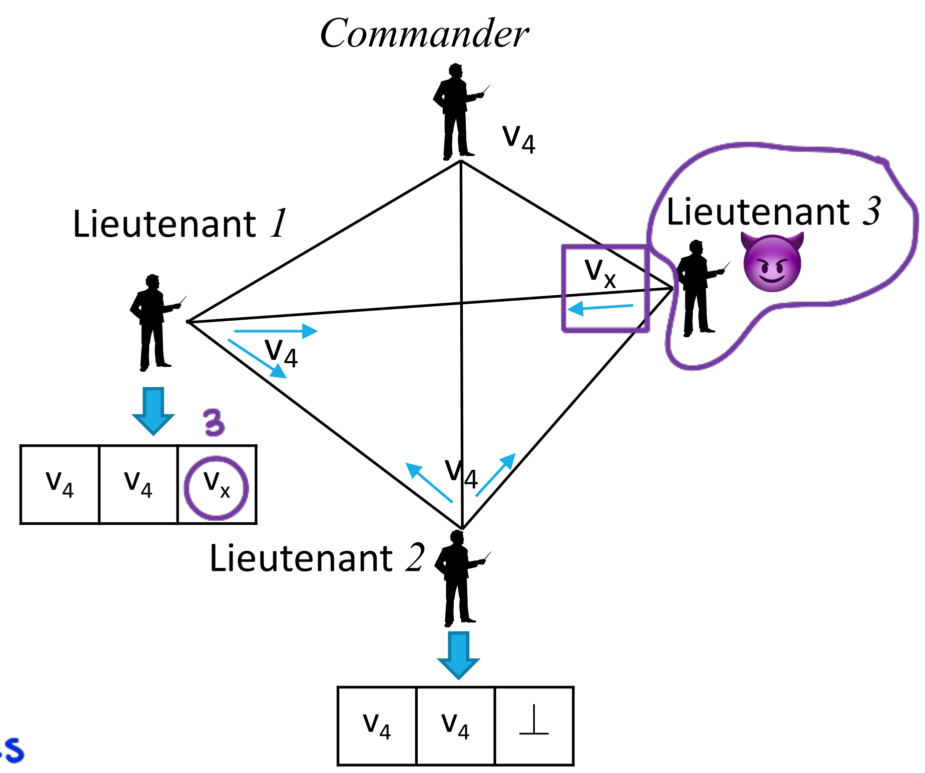
1. The commander sends his value to every lieutenant.
2. For each *i*, let v_i be the value Lieutenant *i* receives from the commander, or else be RETREAT if he receives no value:
   - Lieutenant *i* acts as the commander in algorithm OM(*f* - *1*) to send the value v_i to each of the *N* - 2 other lieutenants.
3. For each *i* and *j* (*j*≠*i*), let v_j be the value Lieut. *i* received from Lieut. *j* in step (2), or else RETREAT if he received no value:
   - Lieutenant *i* uses the value majority (v_l, ..., v_{N-1}).

Implementing the "Oral Message" communication abstraction

OM(1)



STEP 1 ↗ store locally values received from COMM.

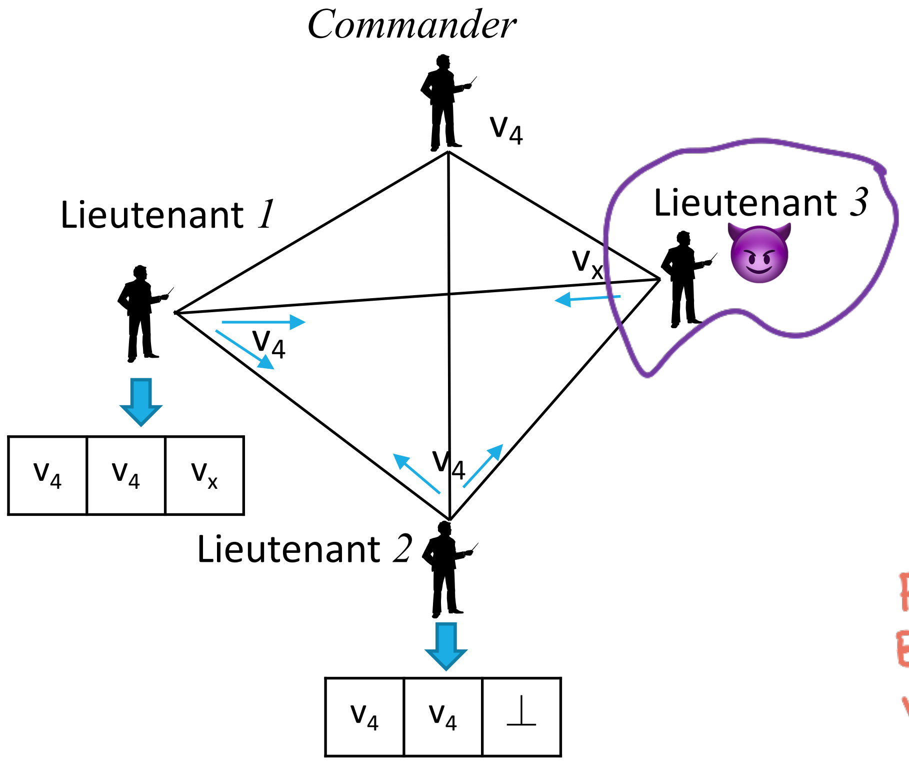


STEP 2

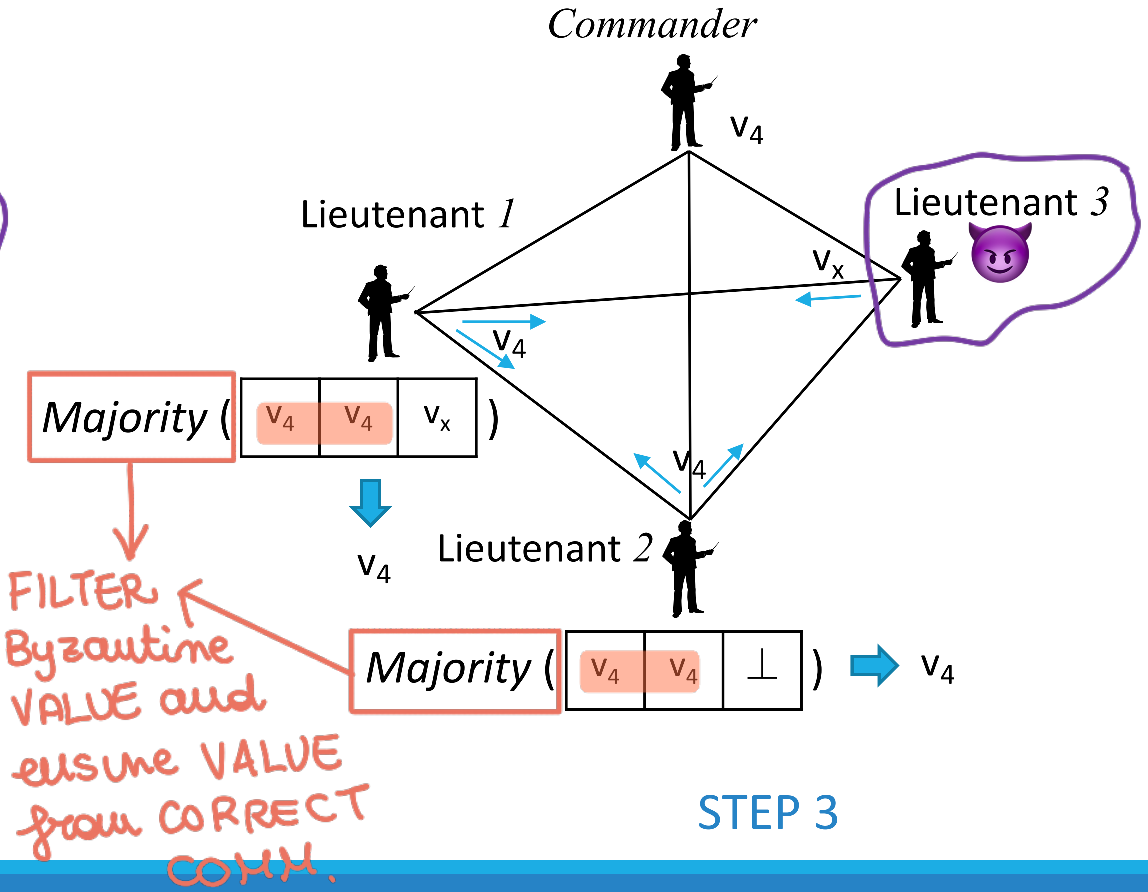
Each Lieutenant starts OM(0)

Implementing the "Oral Message" communication abstraction

OM(1)



STEP 2



Commander
v_4
Lieutenant 1
Lieutenant 3
v_x
v_4
v_4
Lieutenant 2
Majority (v_4 v_4 v_x)
v_4
Majority (v_4 v_4 $\perp$) v_4
FILTER Byzantine VALUE and ensure VALUE from CORRECT COMM.

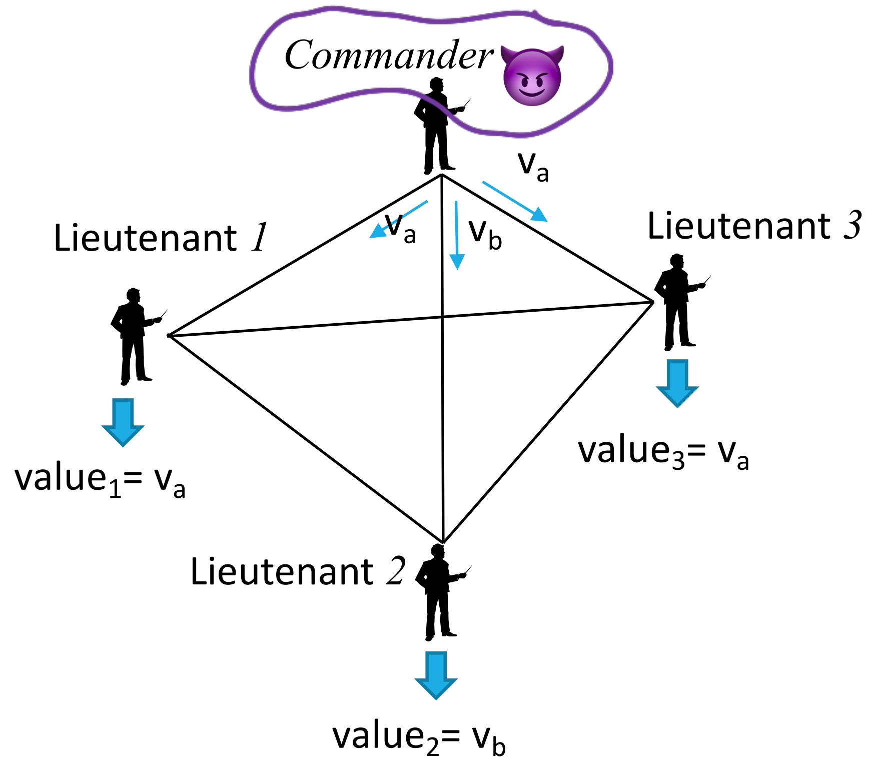
STEP 3

Each Lieutenant starts OM(0)

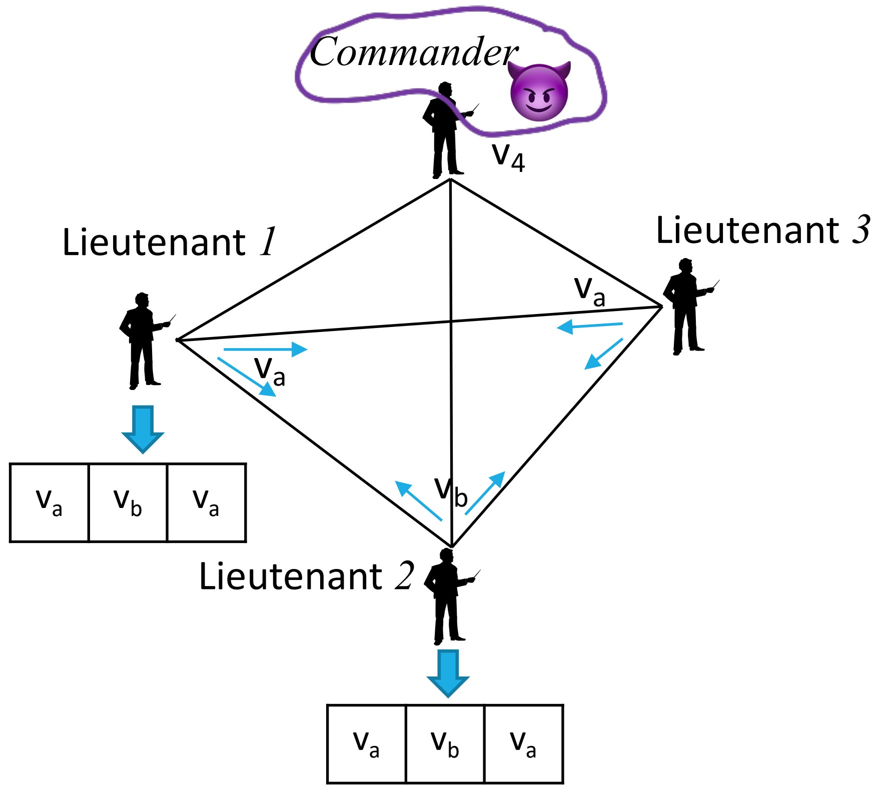
Implementing the ”Oral Message” communication abstraction

OM(1)

BYZANTINE



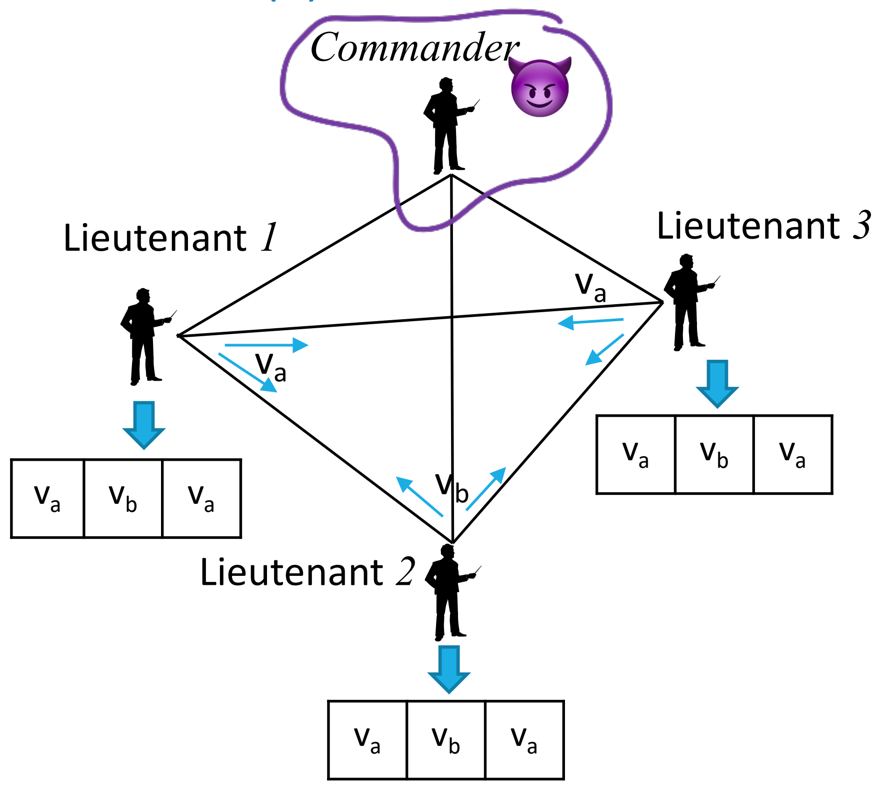
STEP 1



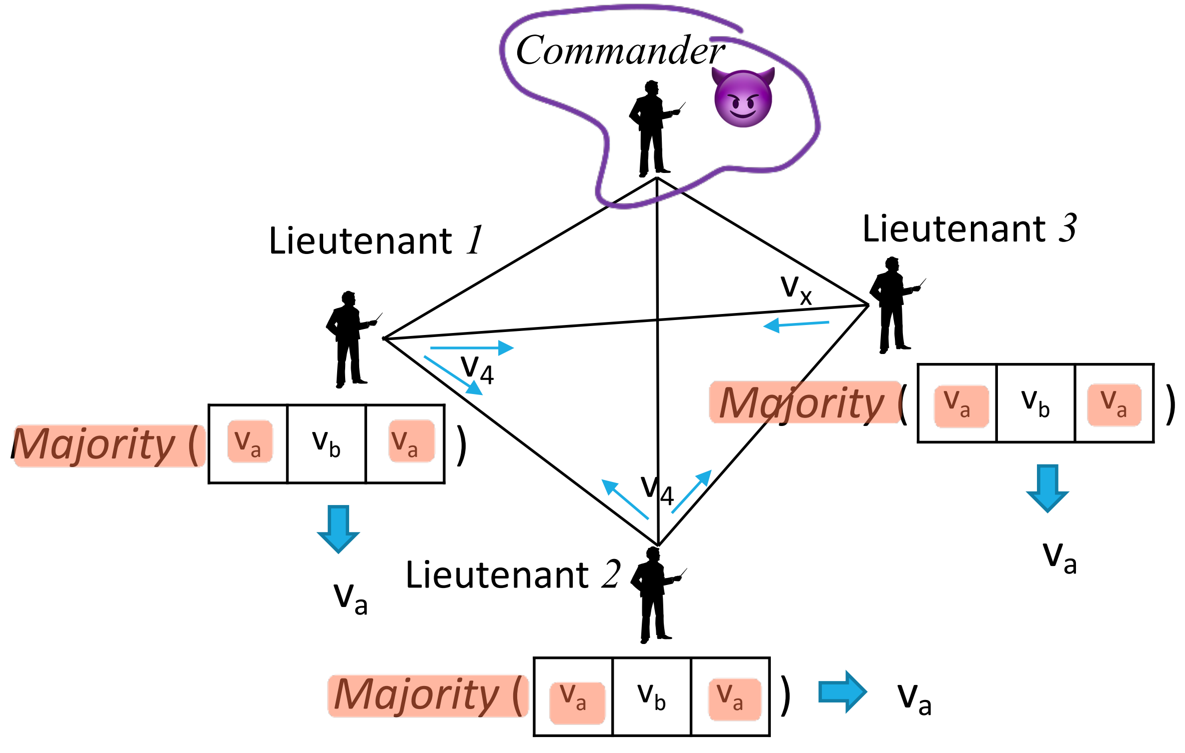
STEP 2

Implementing the "Oral Message" communication abstraction

OM(1)



STEP 2



Commander
Lieutenant 1
Lieutenant 3
Lieutenant 2
v_x
v_4
v_4
Majority (v_a | v_b | v_a) → v_a
Majority (v_a | v_b | v_a) → v_a
Majority (v_a | v_b | v_a) → v_a

STEP 3

Byzantine reliable consensus

Strongly inefficient algorithm

- Recursion creates a large number of messages

The complexity lies in the fact that a Byzantine process can easily change the content of messages

What happens if we use message authentication codes ? SOL: CRYPTOGRAPHY

- A solution for 3 processes exists !

Byzantine reliable consensus

The commander signs and sends his value (*v*:0) to every lieutenant.

For each i:

- If Lieutenant *i* receives a message of the form *v*:0 from the commander and he has not yet received any order, then
  - V_i={*v*}
  - Sends message *v*:0:*i* to every other lieutenant
- If Lieutenant *i* receives a message of the form *v*:0:j_l:....:J_k and *v* is not in V_i then
  - adds *v* to V_i;
  - if $k<f$ sends the message *v*:0:j_l:....:j_k:*i* to every lieutenant other than j_l ..... j_k

For each *i*: when Lieutenant *i* receives no more messages, he obeys the order *choice*(V_i).

References

Leslie Lamport, Robert Shostak, and Marshall Pease "The Byzantine Generals Problem " in ACM TOPLAS 1982

Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/The-Byzantine-Generals-Problem.pdf