BYZANTINE FAULT TOLERANCE

BLOCKCHAIN
@ COLUMBIA

What is Byzantine Fault Tolerance?

Byzantine Fault Tolerance is the characteristic which defines a system that tolerates the class of failures that belong to the ***Byzantine Generals' Problem.***

In plain English, BFT measures ***how tolerant a network is to faulty nodes, <u>assuming nothing of their actions</u>, and that many are faulty (but not majority).***

Byzantine Behavior = malicious, unpredictable, arbitrary, colluding.

BLOCKCHAIN
@ COLUMBIA

What is Byzantine Fault Tolerance?

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some (one or more faults) of its components. If its operating quality decreases, the decrease is proportional to the severity of the failure.

When **Bitcoin** was invented, one of its breakthroughs was the use of Proof-of-Work as a probabilistic solution to the Byzantine Generals Problem.

Byzantine Generals Problem (1982) (Lamport)

Setup:

1. There is a general with parts of their army surrounding a city with the intent to attack it.
2. After each lieutenant observes the city from their side, they must decide what to do, simply to attack or retreat.
3. A majority of lieutenants need to attack/retreat or else they'll fail.
4. Some of the lieutenants/commander aren't loyal and can send mixed messages or collude with other disloyal lieutenants.

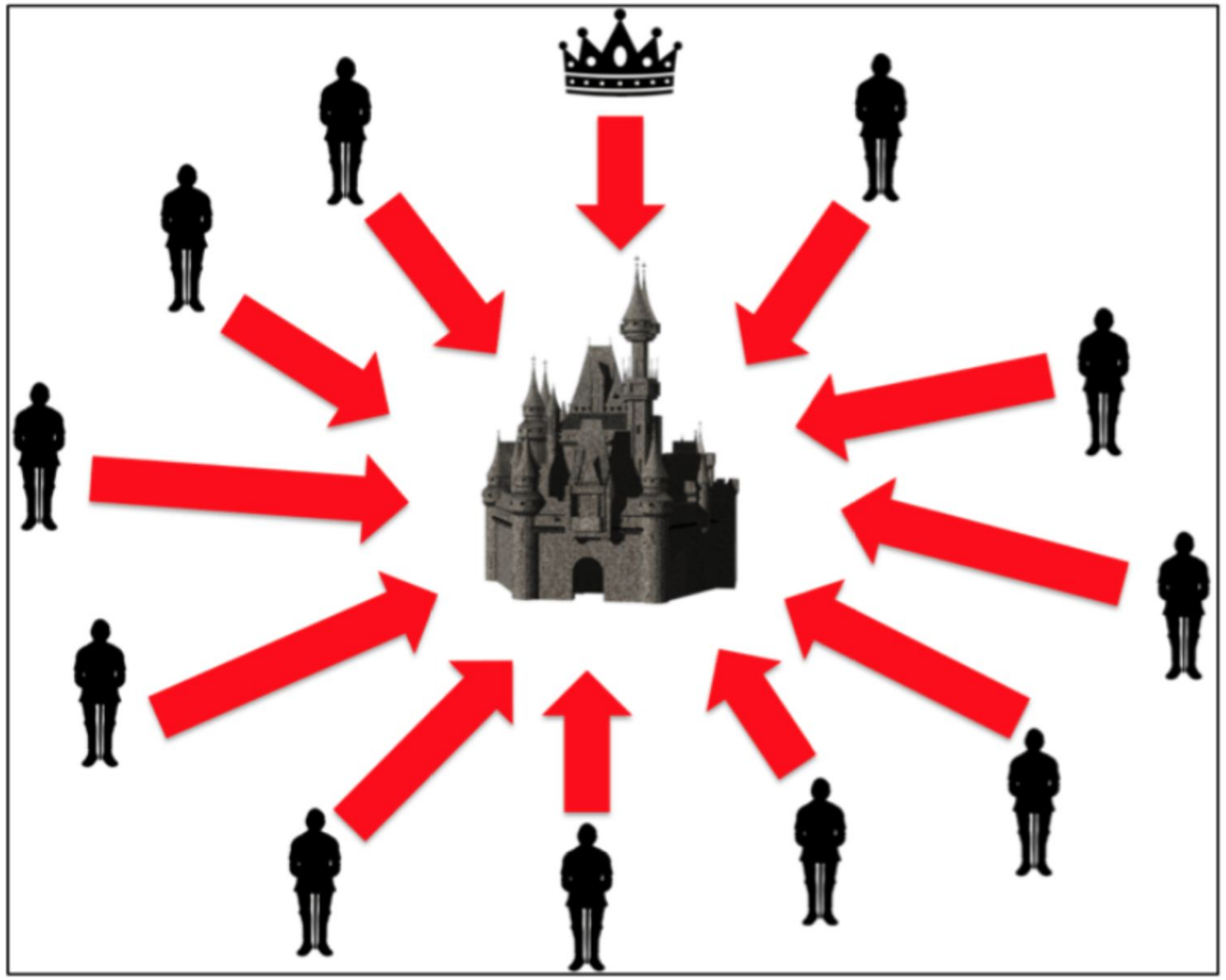
Byzantine Generals Problem

Problem: A commanding general must send an order to his lieutenants such that:

1. All loyal lieutenants obey the same order.
2. If the commanding general is loyal, then every loyal lieutenant obeys the order he sends.

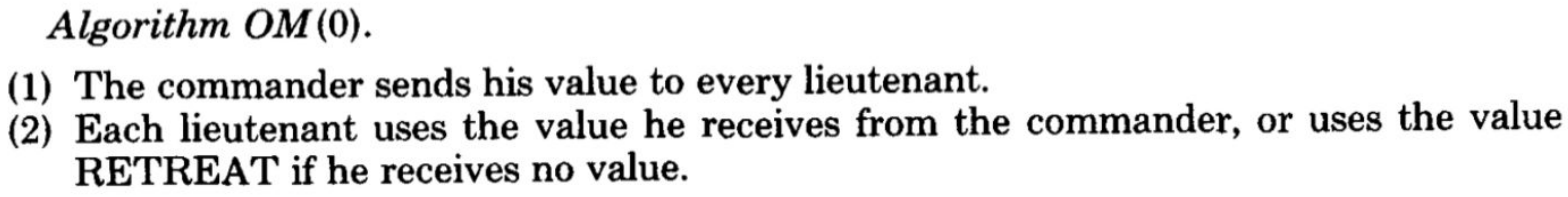
Theorem: For any m, Algorithm OM(m) reaches consensus if there are more than 3m generals and at most m traitors. *Consensus is impossible to achieve if ⅓ or more of the generals are traitors.*

Byzantine Generals Problem



BLOCKCHAIN
@ COLUMBIA

All Loyal Generals

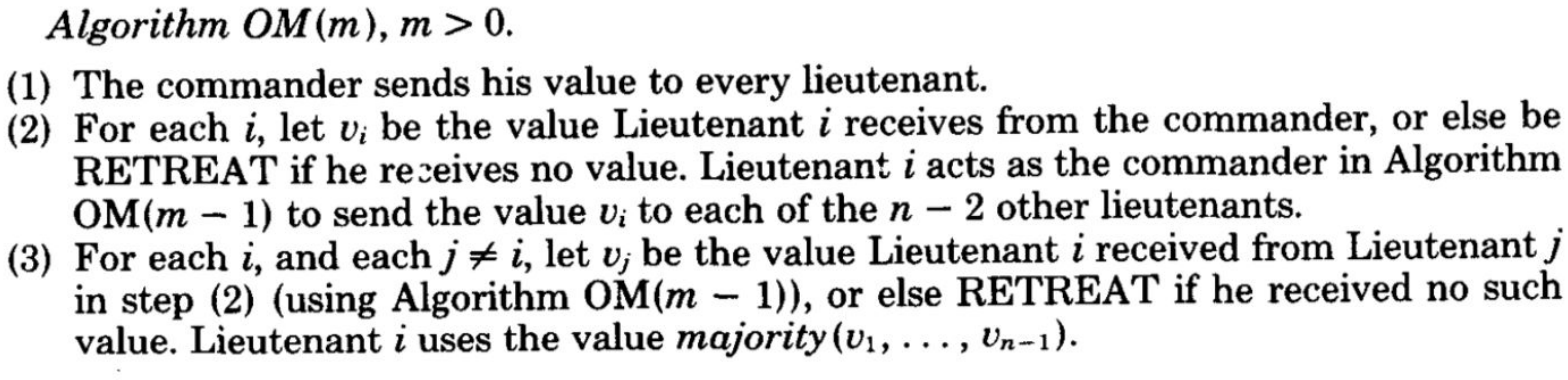


Algorithm OM(0).

(1) The commander sends his value to every lieutenant.
(2) Each lieutenant uses the value he receives from the commander, or uses the value RETREAT if he receives no value.

Everyone sees that the right thing to do is attack.
Everyone notifies everyone else that they're going to attack.
Everyone attacks. Pretty Simple

At least one traitor



Algorithm OM(m), m > 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($m - 1$) to send the value v_i to each of the $n - 2$ other lieutenants.

(3) For each i, and each $j \neq i$, let v_j be the value Lieutenant i received from Lieutenant j in step (2) (using Algorithm OM($m - 1$)), or else RETREAT if he received no such value. Lieutenant i uses the value *majority*$(v_1, \ldots, v_{n-1})$.

$n < 3m + 1$

No solution exists for $n = 3m$

L1 is getting different results from C and L2 and can't tell who's the traitor.

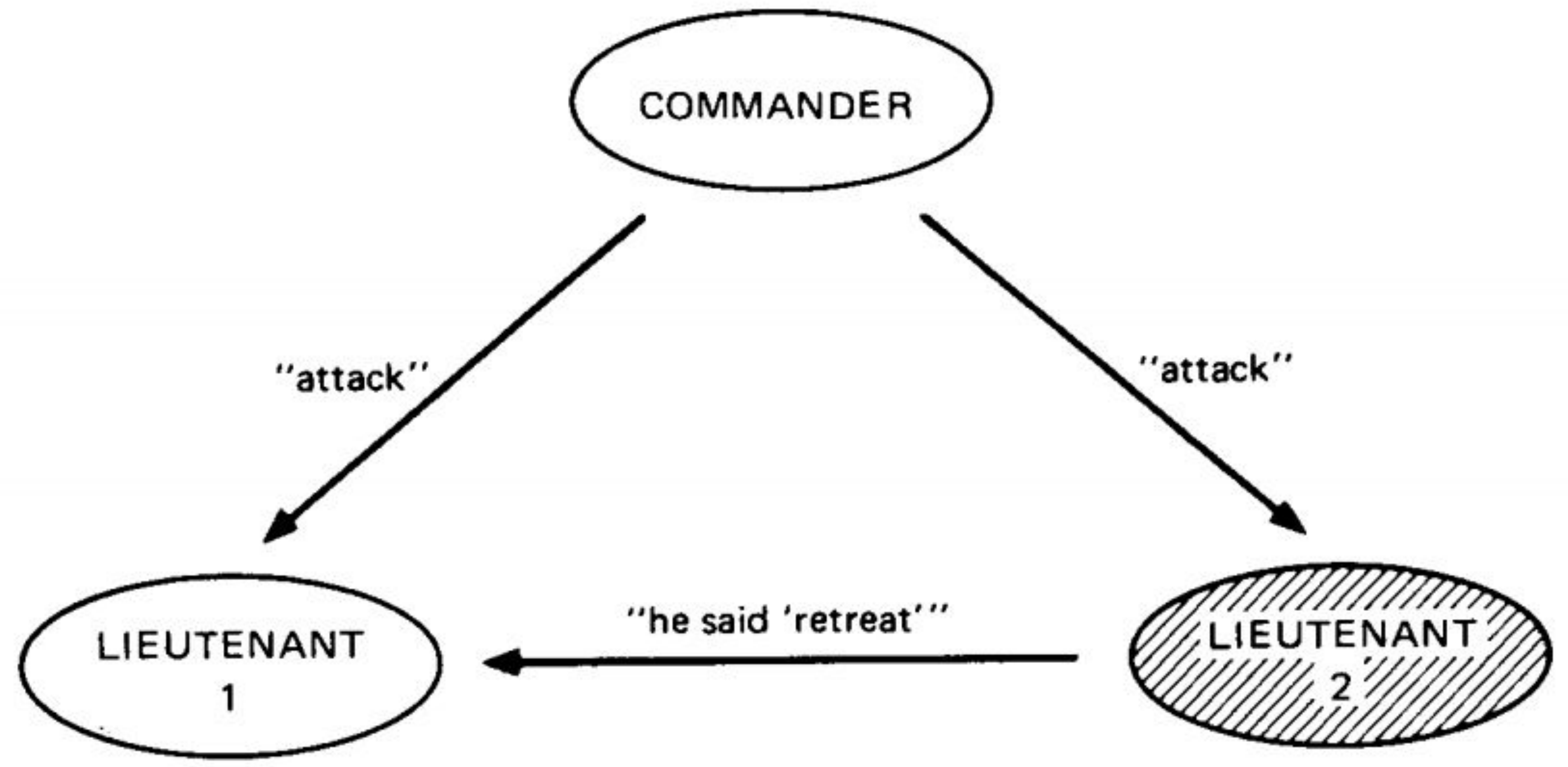


Fig. 1. Lieutenant 2 a traitor.

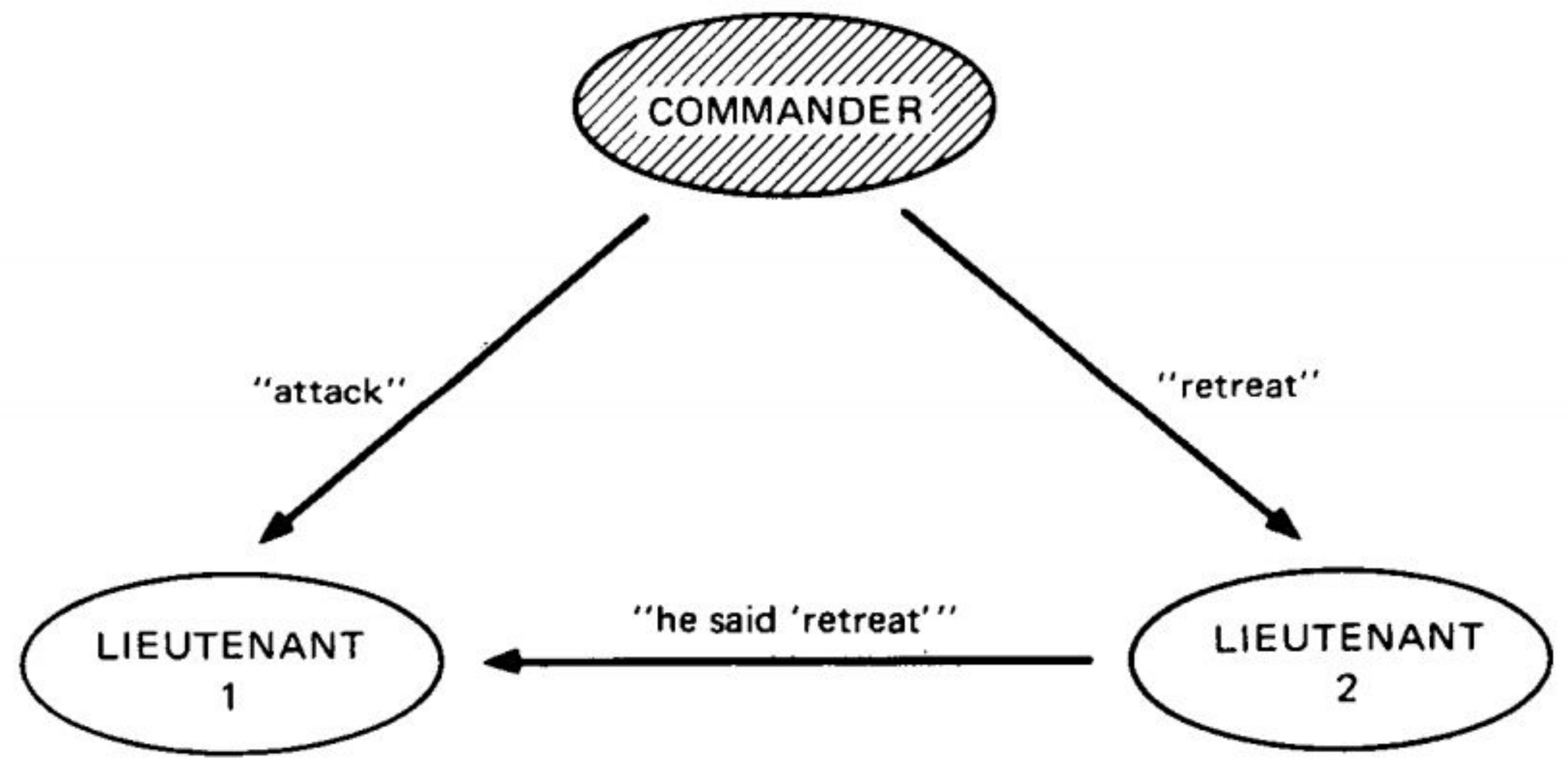
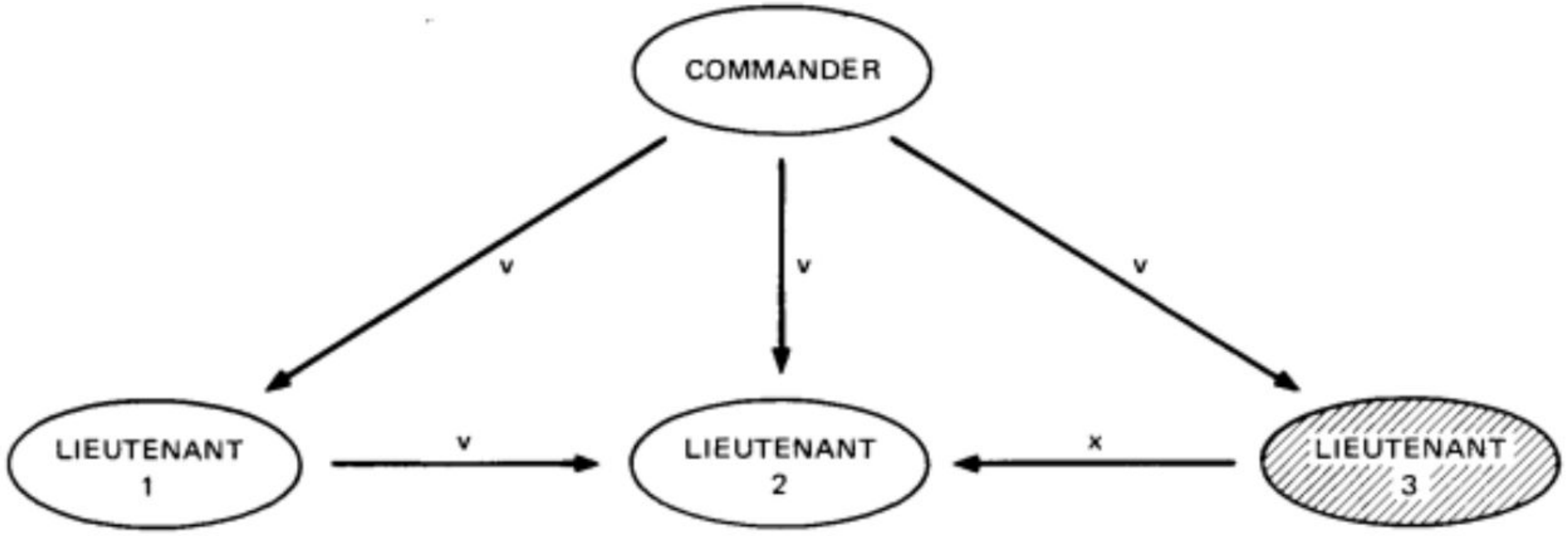


Fig. 2. The commander a traitor.

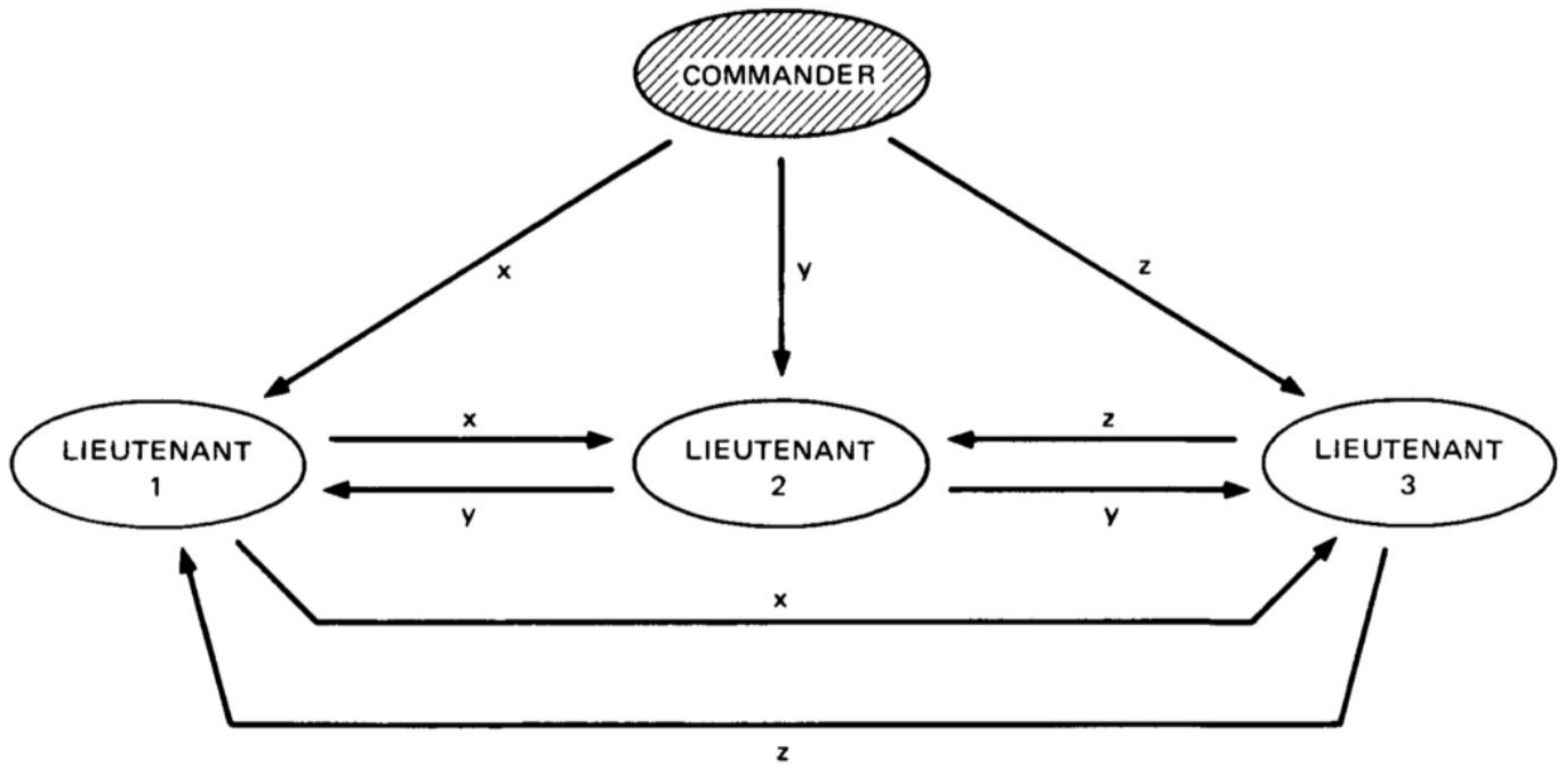
BLOCKCHAIN @ COLUMBIA

$n \geq 3m + 1$



(From L2 Perspective) L2 ← majority(v,v,x) == v

$n \geq 3m + 1$



L1 ← majority(x,y,z) | L2 ← majority(x,y,z) | L3 ← majority(x,y,z)

Since mixed commands, they all retreat.

Practical Byzantine Fault Tolerance (1999)

Practical Byzantine Fault Tolerance (PBFT) (Castro, Liskov) is an approach to fault tolerance containing a **quorum rule** and a **multi-phase learning process.** It guarantees both **liveness** and **safety**, but it relies on a weak form of **synchrony.**

Safety = Nothing harmful can happen to the network. Malicious nodes can't alter the decisions of 'loyal' nodes.

Liveness = something good eventually happens.

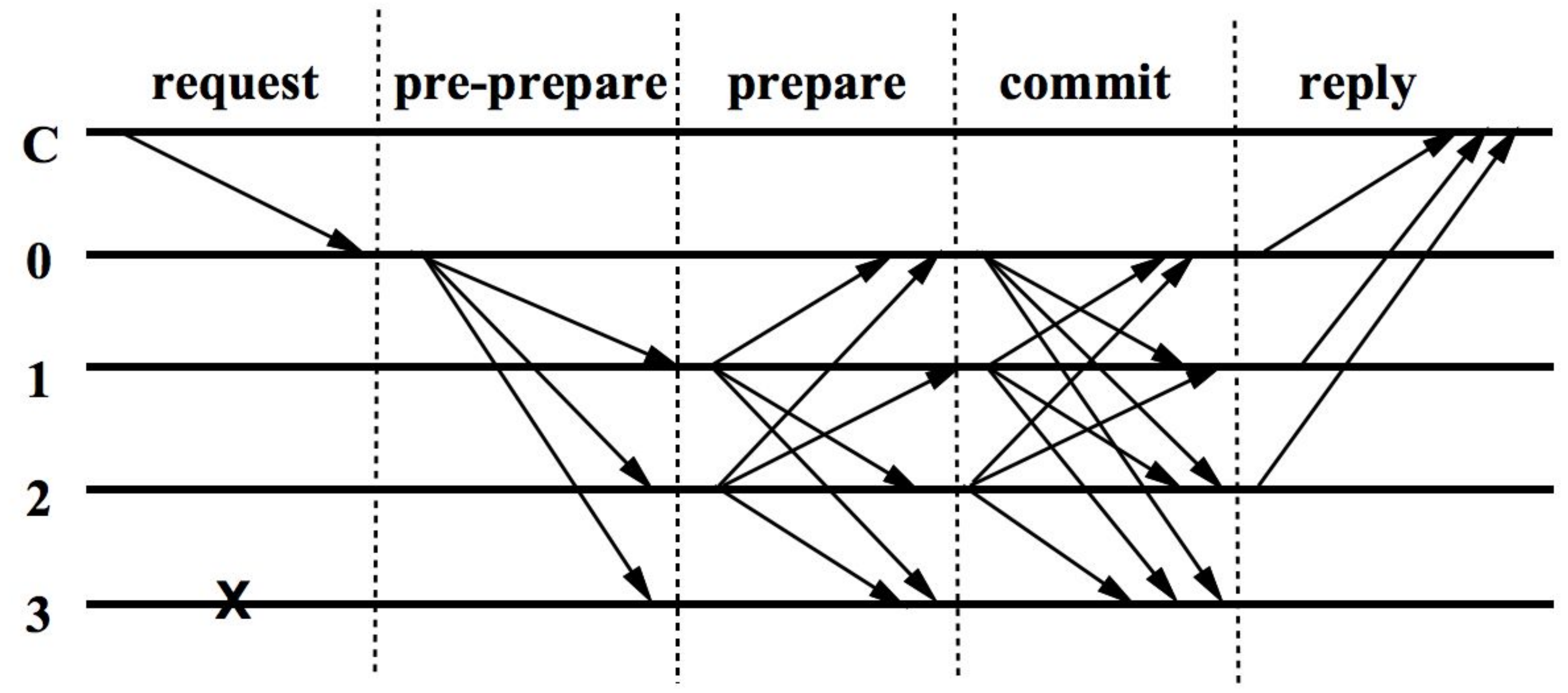
Synchronous vs Asynchronous BFT

Synchronous = Communication happening at roughly the same time (bounded delay).

Asynchronous = Communication that doesn’t have to happen at the same time.

PBFT only somewhat prevents Denial of Service (DoS/DDoS) attacks. It can ensure liveness but not efficiency. Makes it difficult to implement well.

Practical Byzantine Fault Tolerance



1. A **leader** is chosen at random and changed for every call (round robin).
2. A **client** sends a request to the leader node to invoke a service operation.
3. The leader node multicasts the request to the backup nodes.
4. The nodes execute the request and then send a reply to the client.
5. The client awaits m + 1 replies from different nodes with the same result. This result is the result of the operation.

BLOCKCHAIN
@ COLUMBIA

How does this relate to blockchain?

Bitcoin became the first successful distributed network that was DDoS-proof and sufficiently secure from a 51% attack (2m + 1).

The incentives behind proof-of-work make it very costly to attempt an attack on the Bitcoin network.

This is the basis for next week's lecture about **Consensus Mechanisms**, which must be BFT to a certain extent.