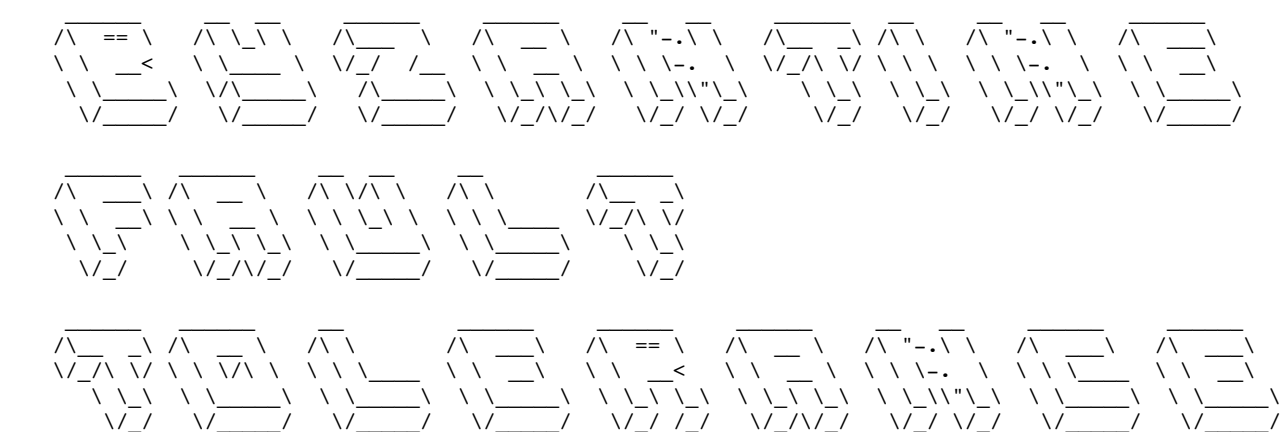
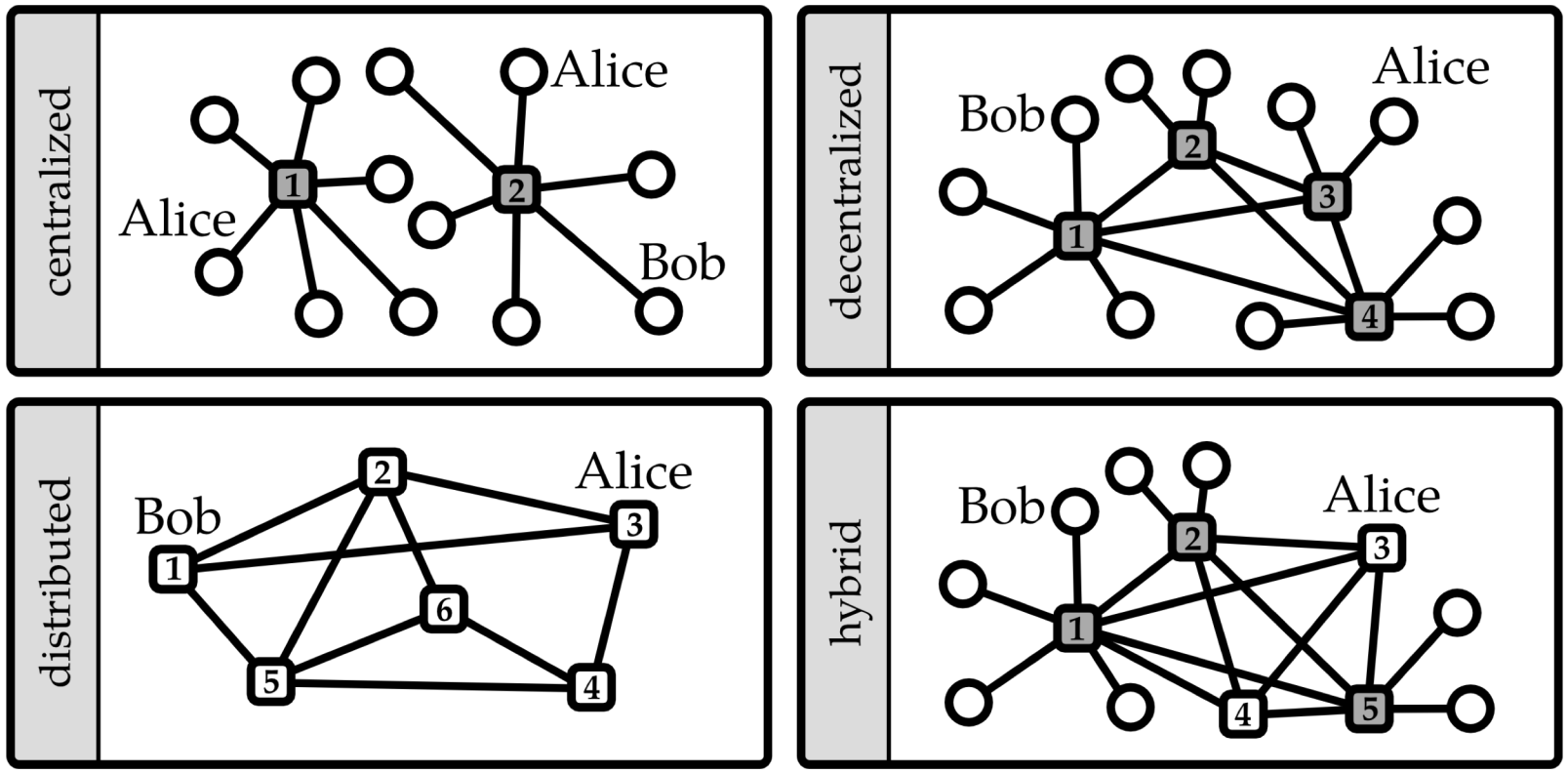


Centralized vs. Decentralized vs. Distributed




Byzantine Empire
Sassanid Empire
North Sea
Baltic Sea
Rhine R.
Dnieper R.
Volga R.
N
W
E
S
ATLANTIC OCEAN
Loire R.
FRANKISH KINGDOM
LOMBARDS (568)
Venice
Ravenna
Danube R.
BULGARS
AVARS
SLAVS
DALMATIA
BALKAN MTS
BALKANS
Black Sea
Aral Sea
Jaxartes R.
Caspian Sea
Rome
(572)
Bari
Constantinople
Corsica
VISIGOTHIC KINGDOM
Sardinia
Calabria
Sicily
BYZANTINE EMPIRE
Ephesus
Strait of Gibraltar
Mediterranean Sea
Antioch
SYRIA
MESOPOTAMIA
Tigris R.
Euphrates R.
Ctesiphon
SASSANID EMPIRE (PERSIA)
Damascus
PALESTINE
Jerusalem
Indus R.
NORTH AFRICA
Alexandria
EGYPT
ARABS
Red Sea
Persian Gulf
Nile R.
Mecca
0 250 500 miles
0 250 500 kilometers

Byzantine Fault and Failure

- “A Byzantine fault is any fault presenting different symptoms to different observers.”
- “A Byzantine failure is the loss of a system service due to a Byzantine fault in systems that require consensus.”
- Honeybee Swarms
- Boeing 777 Airplanes
- Virginia Class Submarines
- SpaceX Dragon Capsule
- Nuclear Power Plants

Byzantine Fault Tolerance

- “The objective of Byzantine fault tolerance is to be able to defend against failures of system components with or without symptoms that prevent other components of the system from reaching an agreement among themselves, where such an agreement is needed for the correct operation of the system.”
- “Remaining correctly operational components of a Byzantine fault tolerant system will be able to continue providing the system's service as originally intended, assuming there are sufficient accurately operating components to maintain the service.”

The Byzantine Generals Problem - Intro

- A thought experiment to abstractly describe an agreement problems
- An agreement protocol is outlined as a solution
- Though very general, it applies to a whole class of challenges with distributed systems
- Described in: "The Byzantine Generals Problem” (1982)
  - Leslie Lamport, Robert Shostak and Marshall Pease

The Byzantine Generals Problem - Problem

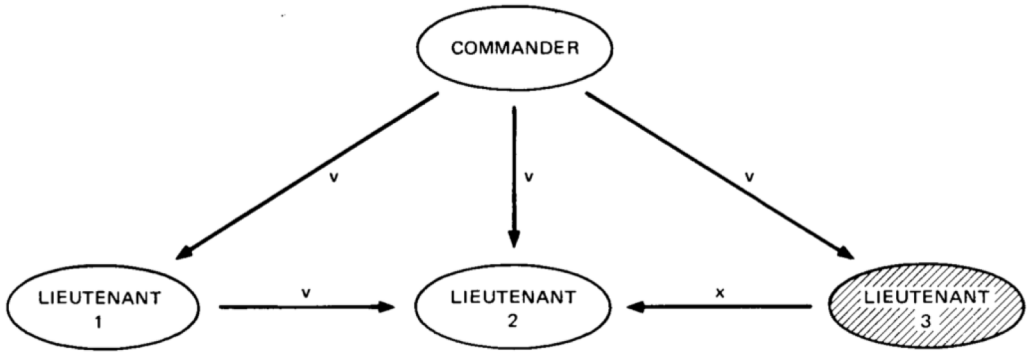
- A large army encircles a city and prepares for a siege
- A commander has to give orders to his lieutenants who each controls a different section of the army
- There are two options – Attack or Retreat
- The army must reach a consensus because a partial attack would be bad for the army
- The general and lieutenants must communicate via messengers
- A certain number of the lieutenants may be traitors
- The commander may be a traitor
- No one is trusted to compile and organize votes centrally (see above)
- The lieutenants may may disagree on which option to choose
- The lieutenants may intentionally or accidentally send different votes to different lieutenants
- The messengers may change votes or fail to deliver messages

The Byzantine Generals Problem - Tests

- A solution to an agreement problem must pass three tests:
  - Termination
    - All non-traitor lieutenants must eventually reach a decision about the order they received
  - Agreement
    - All non-traitor lieutenants must arrive at the same value as to the order they were given
  - Validity
    - If the commander is not a traitor then all lieutenants must arrive at the same order as was originally issued by the commander
- Notice an unusual edge-case? (hint: look at validity)

The Byzantine Generals Problem - Scenarios

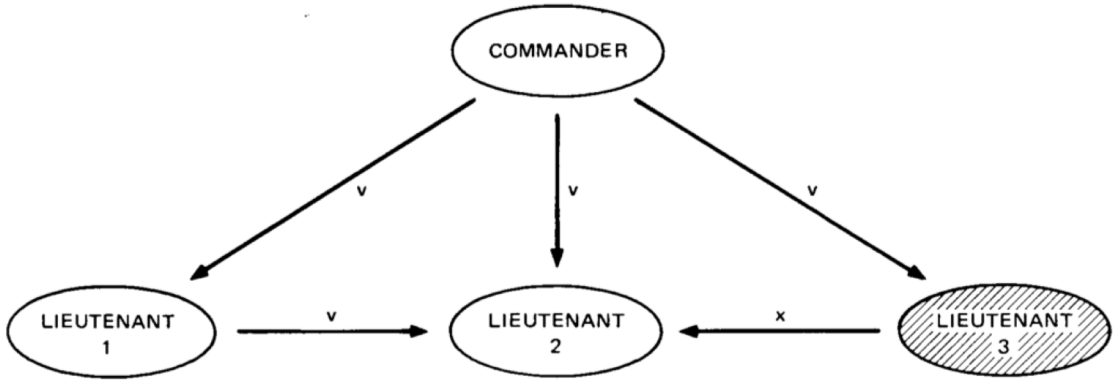
- 4 generals, 3 want to attack
  1 is a traitor and relays a
  false message
  - #2 receives conflicting messages
  - No consensus reached



- 9 generals, 4 want to attack, 4 want to retreat, 1 traitor
  - 1 general could send an attack vote to the 4 that want to attack and a retreat vote to the 4 that want to retreat
  - Both groups will think they are in the majority
  - Half the army destroyed

The Byzantine Generals Problem – Naïve Solution

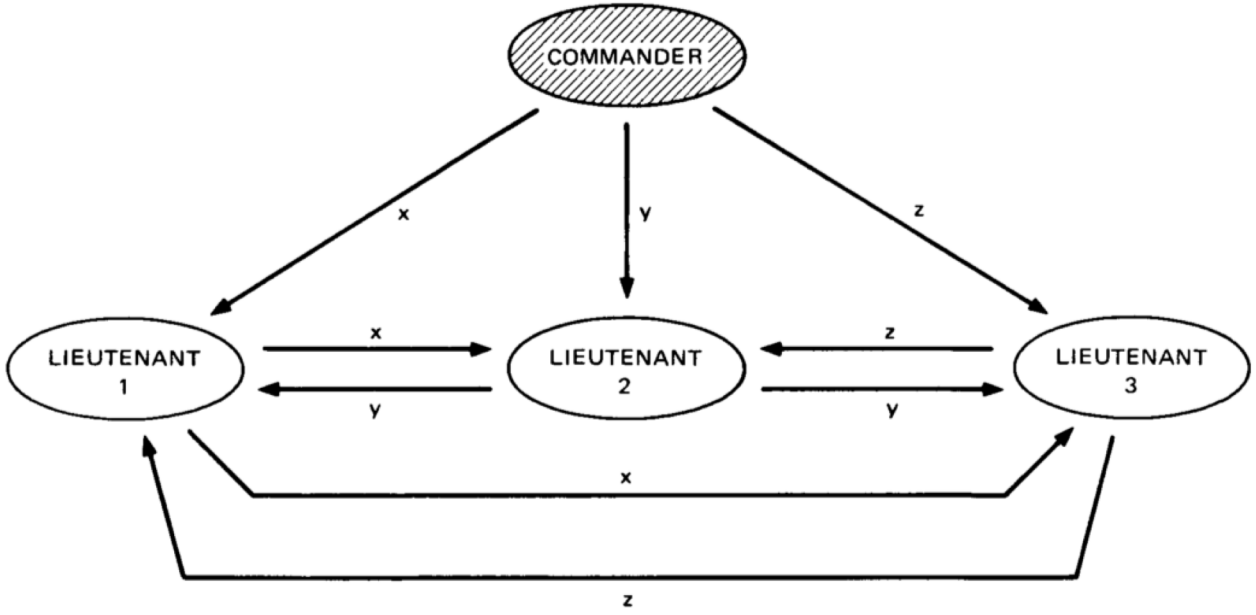
- Follow the majority
- Steps:
  - Commander sends *v* to all Lieutenants
  - L1 sends *v* to L2 | L3 sends *x* to L2
  - L2 ← *majority(v,v,x) == v*
  - The final decision is the majority vote from L1, L2, L3 and as a result consensus has been achieved



- "The important thing to remember is that the goal is for the majority of the lieutenants to choose the **same** decision, not a specific one."

The Byzantine Generals Problem – Naïve Solution

- Follow the majority, have a default
- Steps:
  - Commander sends *x, y, z* to L1, L2, L3
  - Dissemination
    - L1 sends *x* to L2, L3
    - L2 sends *y* to L1, L3
    - L3 sends *z* to L1, L2
  - Decisions
    - L1 ← majority(x,y,z)
    - L2 ← majority(x,y,z)
    - L3 ← majority(x,y,z)



- They all have the same value and thus consensus is reached. Take a moment here to reflect that even if *x, y, z* are all different the value of *majority(x, y, z)* is the same for all 3 Lieutenants. In the case x,y,z are totally different commands, we can assume that they act on the default option *retreat*.

However

- Those only show 1 bad actor
- A bad source working with 1 bad collaborator can split the group
- The actual algorithm is more complicated:
  - http://marknelson.us/2007/07/23/byzantine/
  - The three tests of the agreement problem will be satisfied
  - As long as the number of faulty actors is less than 1/3 of the actors

Trust

- Fundamentally this is all about trust
- How do you reach a decision (consensus) when you may not trust the authenticity or accuracy of the information?
- How do you create a system that allows you to reach a decision (consensus) in that kind of environment?
- What can you do if you **can** reliably reach a decision (consensus) across distributed groups of untrusted participants?

Further Reading

- The original Byzantine General's Problem (1982) - https://www.microsoft.com/en-us/research/uploads/prod/2016/12/The-Byzantine-Generals-Problem.pdf
- Byzantine Fault Tolerance, from Theory to Reality (2003) - https://www.cs.indiana.edu/classes/p545/post/lec/fault-tolerance/Driscoll-Hall-Sivencrona-Xumsteg-03.pdf
- The Real Byzantine Generals (2004) - https://www.researchgate.net/publication/4122503_The_real_Byzantine_Generals

Sources

- https://en.wikipedia.org/wiki/Byzantine_fault_tolerance
- https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419
- http://marknelson.us/2007/07/23/byzantine/