Consensus Algorithms

Consensus

- A fundamental problem in fault-tolerant distribution systems.

References:
https://raft.github.io/
https://en.wikipedia.org/wiki/Consensus_(computer_science)

Consensus

- A fundamental problem in fault-tolerant distribution systems.
- It involves multiple servers agreeing on same values.

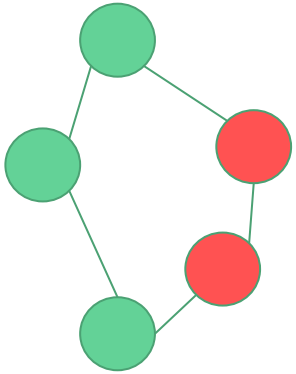
References:
https://raft.github.io/
https://en.wikipedia.org/wiki/Consensus_(computer_science)

Consensus

- A fundamental problem in fault-tolerant distribution systems.
- It involves multiple servers agreeing on same values.
- A fundamental problem in distributed computing is to achieve overall system reliability in the presence of a number of faulty processes.

References:
https://raft.github.io/
https://en.wikipedia.org/wiki/Consensus_(computer_science)

Consensus



This distributed network will work correctly.

faulty node

Raft Consensus Algorithms

- Raft is a consensus algorithms for managing a replicated log.

References: https://raft.github.io/raft.pdf

Raft Consensus Algorithms

- Raft is a consensus algorithms for managing a replicated log.
- It’s equivalent to *Paxos* in fault tolerance & performance.

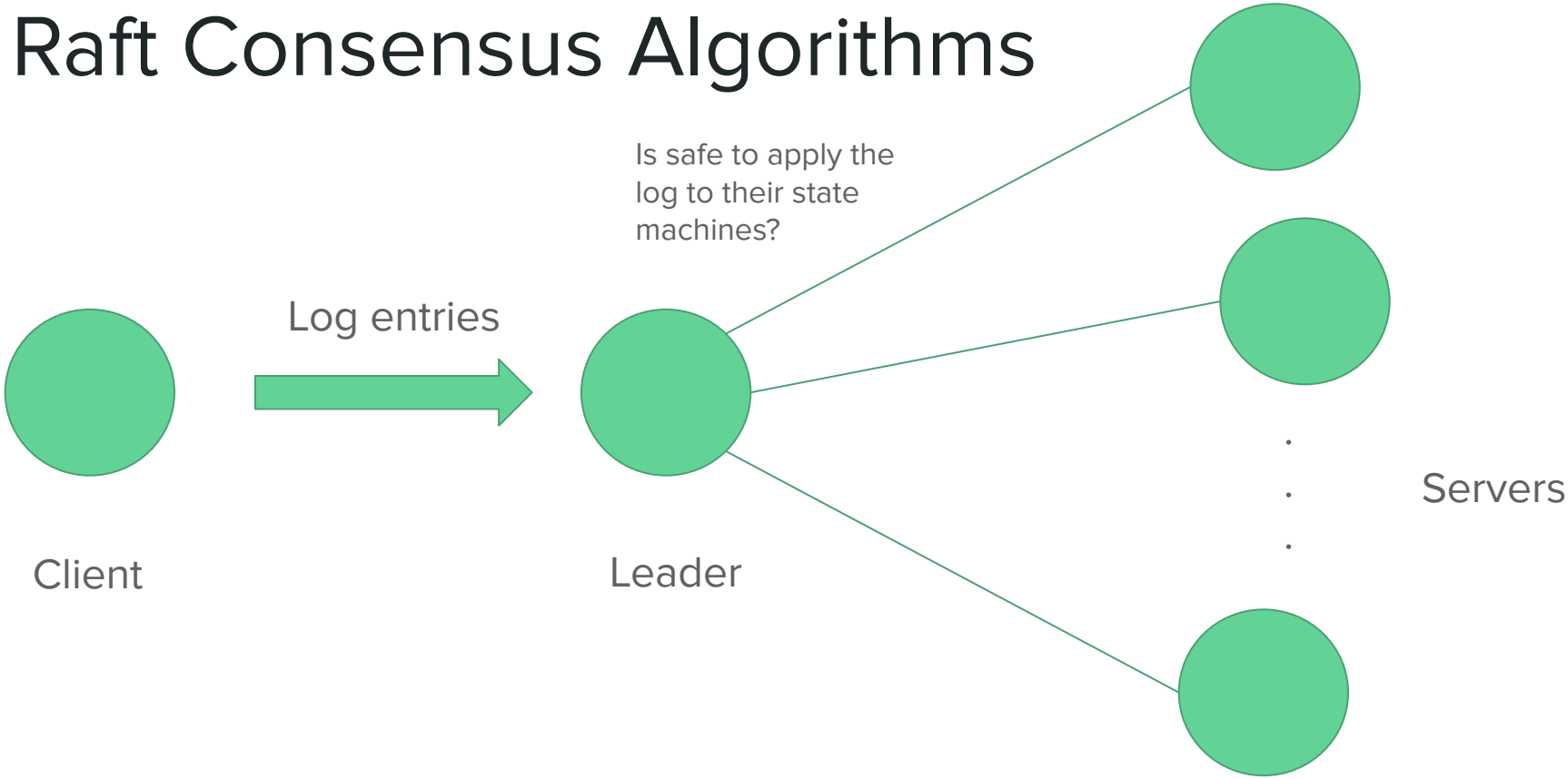
References: https://raft.github.io/raft.pdf

Raft Consensus Algorithms

- Raft is a consensus algorithms for managing a replicated log.
- It's equivalent to *Paxos* in fault tolerance & performance.
- Raft implements consensus by first electing a distinguished leader, then giving the leader complete responsibility for managing the replicated log.

References: https://raft.github.io/raft.pdf

Raft Consensus Algorithms



Need of Leader in Raft Consensus Algorithms

- Leader simplifies the management of the replicated log.

References: https://raft.github.io/raft.pdf

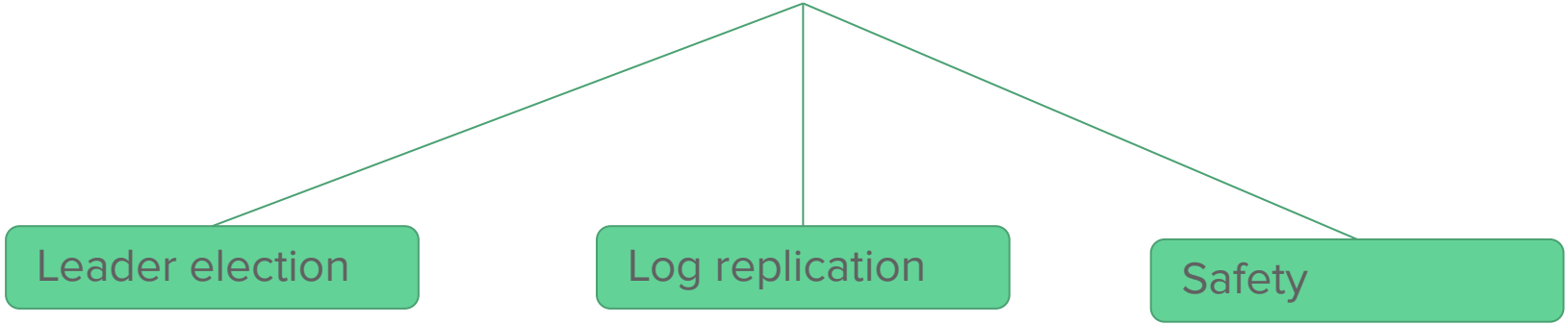
Need of Leader in Raft Consensus Algorithms

- Leader simplifies the management of the replicated log.
- Leader also helps in the decomposition of the consensus problem.

References: https://raft.github.io/raft.pdf

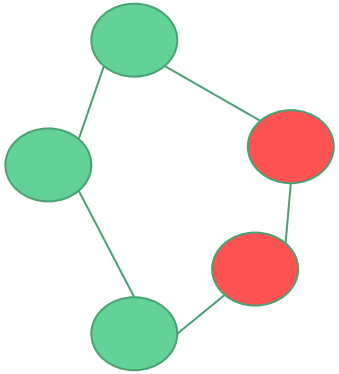
Decomposition of Consensus Problem

Raft decomposed the consensus problem into three relatively independent subproblem –



References: https://raft.github.io/raft.pdf

Raft Basics



can tolerate 2 failure.

References: https://raft.github.io/raft.pdf

Raft Basics

Server can be divided in three states:

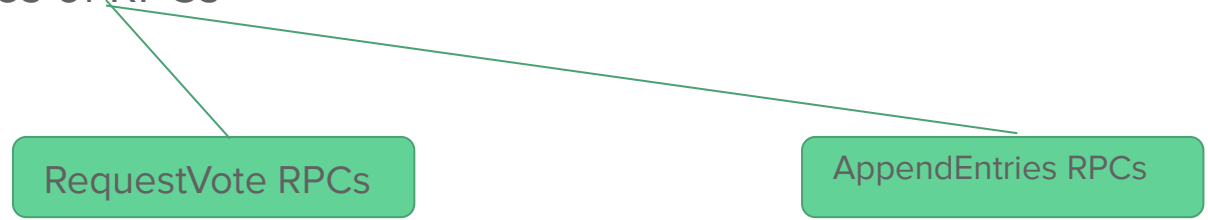
1. *Leader*
2. *Follower*
3. *Candidate*

References: https://raft.github.io/raft.pdf

Raft Server communication

Raft servers communicate using remote procedure calls (RPCs).

Two types of RPCs



References: https://raft.github.io/raft.pdf

More about Raft algorithms

Raft basically using three criteria

1. understandability
2. correctness
3. performance

References: https://raft.github.io/raft.pdf

fault tolerant in Raft algorithms

2f + 1 Ralf nodes tolerates failure of ‘f’ Raft nodes.

References: https://raft.github.io/raft.pdf

Thank you :)