NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Prof. Sandip Chakraborty

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 27: State Machine Replication as Distributed Consensus

CONCEPTS COVERED

- State Machine Replication as a Consensus
- Synchronous vs Asynchronous Consensus with Crash Faults

KEYWORDS

- **Crash Fault Tolerance**
- **Paxos**

State Machine Replication as Consensus

- There is a natural reason to use state machine replication-based consensus over permissioned blockchains
    - The network is closed, the nodes know each other, so state replication is possible among the known nodes
    - Avoid the overhead of mining - do not need to spend anything (like power, time, bitcoin) other than message passing
    - However, consensus is still required - machines can be faulty or behave maliciously

State Machine Replication as Consensus

- There is a natural reason to use state machine repli ned bloc
  - T er, so s odes
  - A pend a essage I
  - H can be faulty or be aliciously

But, we need a bit redesign !

State Machine Replication as Consensus

- Ther chine
  rep oned
  blo
  - her, so
    nodes
  - spend
    message
  - H es can be
    faulty or be e maliciously

But, we need a bit redesign !

Crypto is the saver

Crypto + Distributed Consensus = Consensus for Permissioned Blockchain

State Machine Replication as Consensus

- Classical Distributed Consensus Algorithms (**Paxos**, **RAFT**, **Byzantine Agreement**) are based on State Machine Replication
  - Let us (re)visit those algorithms

Faults in a Distributed Systems

- **Crash Faults**: The node stops operating – hardware or software faults
  - In an asynchronous system: You do not know whether messages have been delayed or the node is not responding
  - Rely on majority voting – progress as and when you have received the confirmation from the majority
  - Propagation of the consensus information – nodes on a slow network will receive it eventually

Faults in a Distributed Systems

- **Byzantine Faults:** Nodes misbehave – send different information to different peers (partition the network)
    - More difficult to handle
    - More suitable for blockchains

Asynchronous Consensus with Crash Faults

- Remember the **FLP Impossibility**
  - Give priority to safety over liveness

- Guarantees the followings --
  - **Validity**: If all correct process proposes the same value v, then any correct process decides v
  - **Agreement:** No two correct processes decide differently
  - **Termination**: Every correct process eventually decides

Asynchronous Consensus with Crash Faults

- Guarantees the followings --
  - **Validity**: If all correct process proposes the same value v, then any correct process decides v **(Unlikely to happen in PoW)**
  - **Agreement:** No two correct processes decide differently **(Safety – Not in PoW)**
  - **Termination**: Every correct process eventually decides **(Liveness – Priority in PoW)**

CFT Consensus

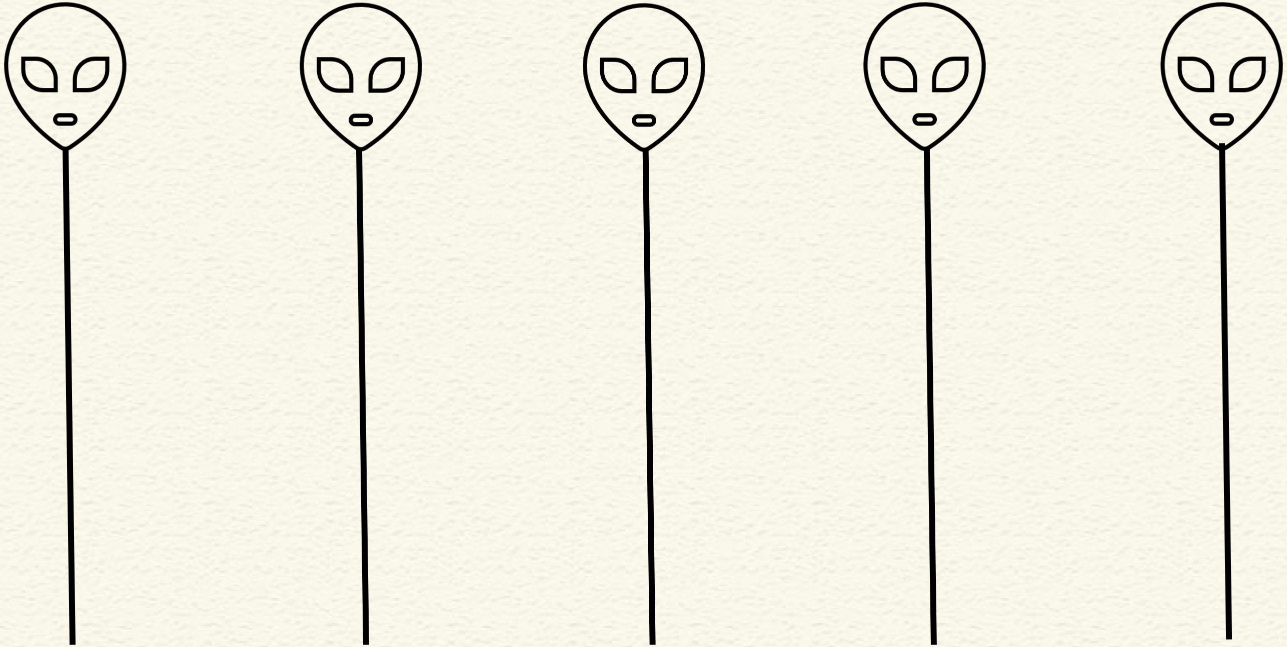
- CFT Consensus
  - **Paxos** (Proposed by Lamport, the most fundamental CFT) -- used in DynamoDB
  - **RAFT** (Much simpler than Paxos) -- Used in Fabric Transaction Ordering

CFT Consensus

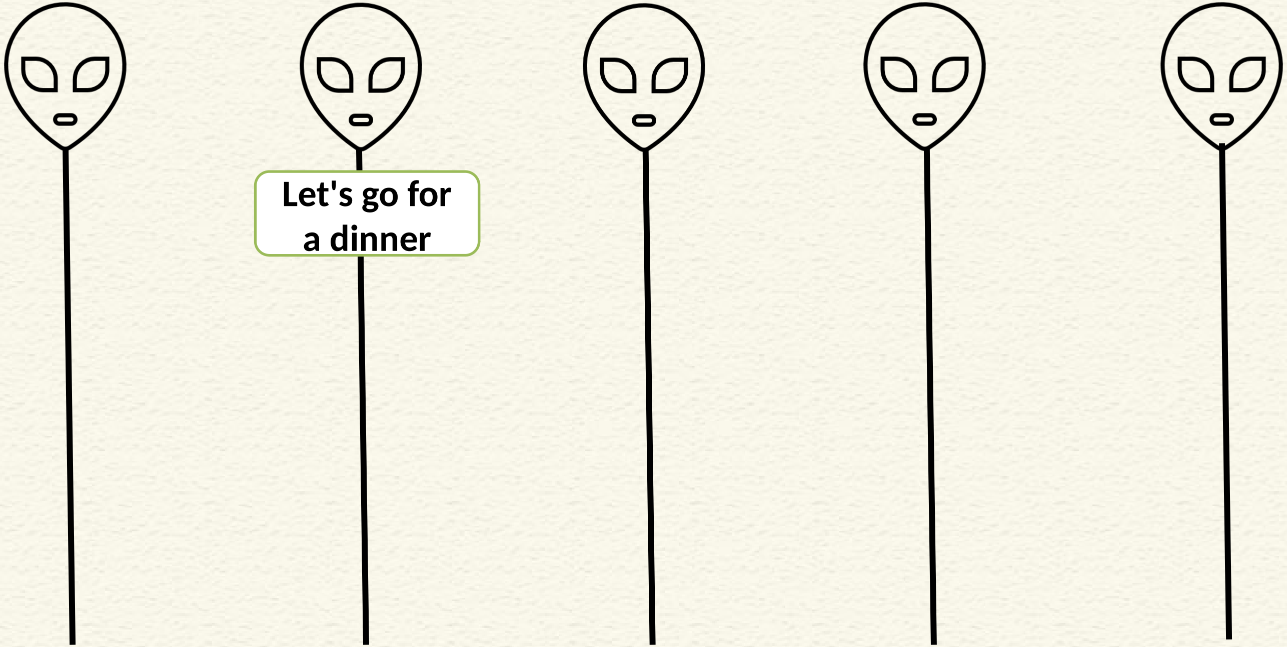
We'll see how Paxos works

- CFT Consensus
  - **Paxos** (Proposed by Lamport, the most fundamental CFT) -- used in DynamoDB
  - **RAFT** (Much simpler than Paxos) -- Used in Fabric Transaction Ordering

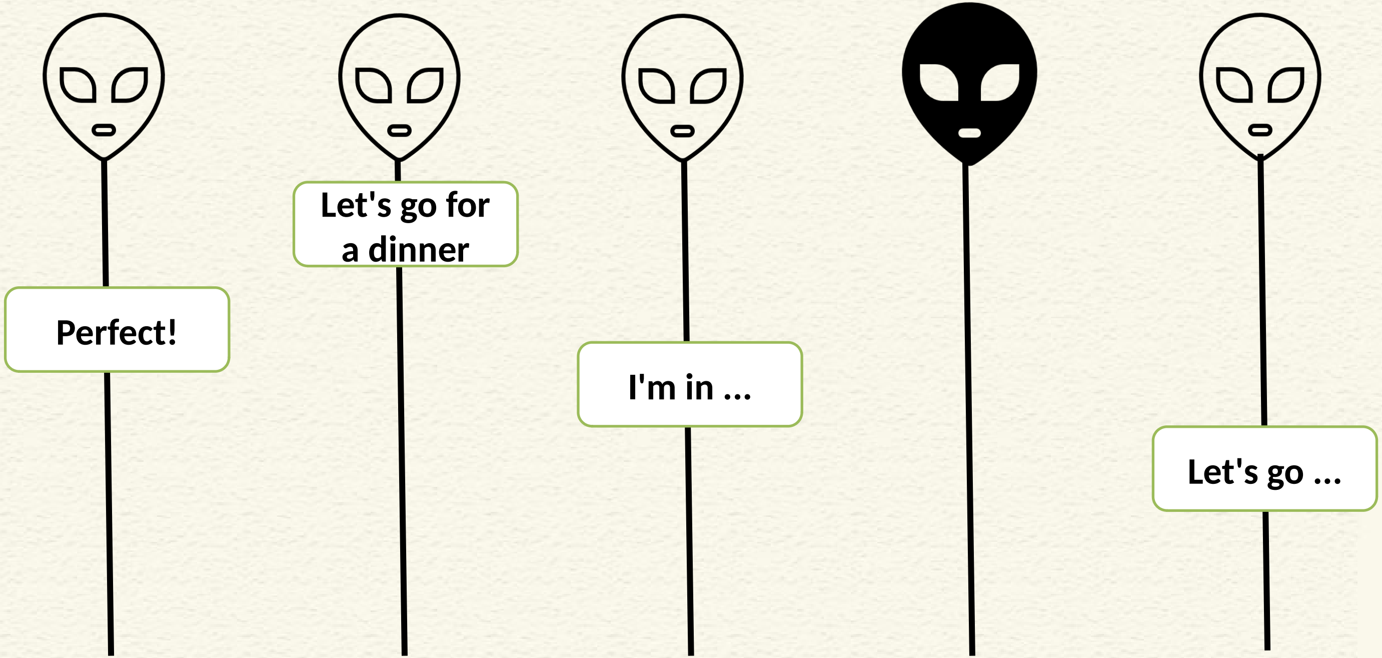
CFT in a Synchronous System



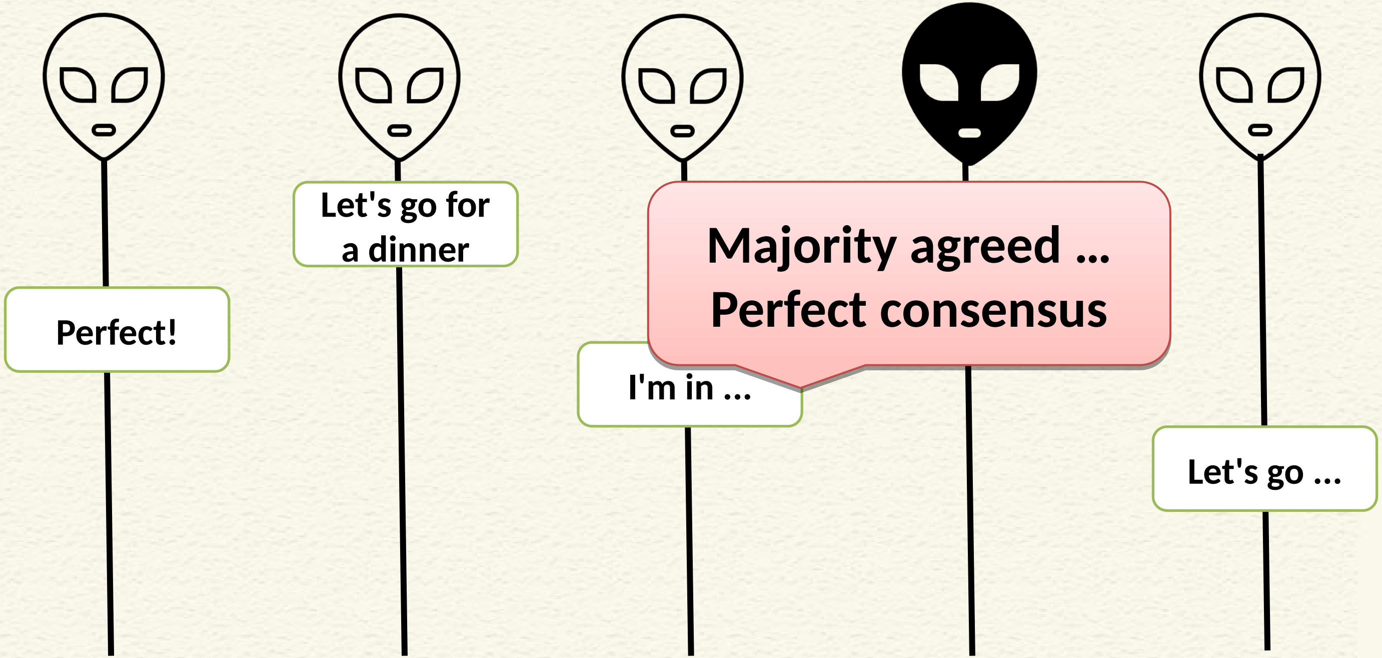
CFT in a Synchronous System



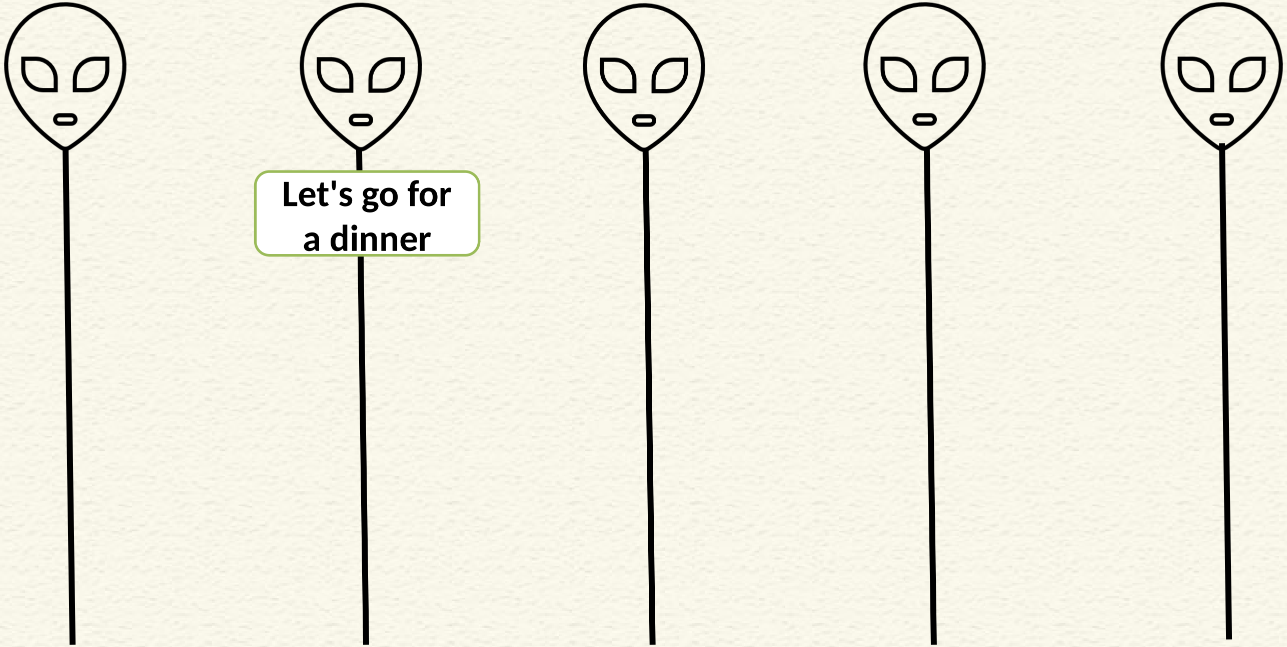
CFT in a Synchronous System



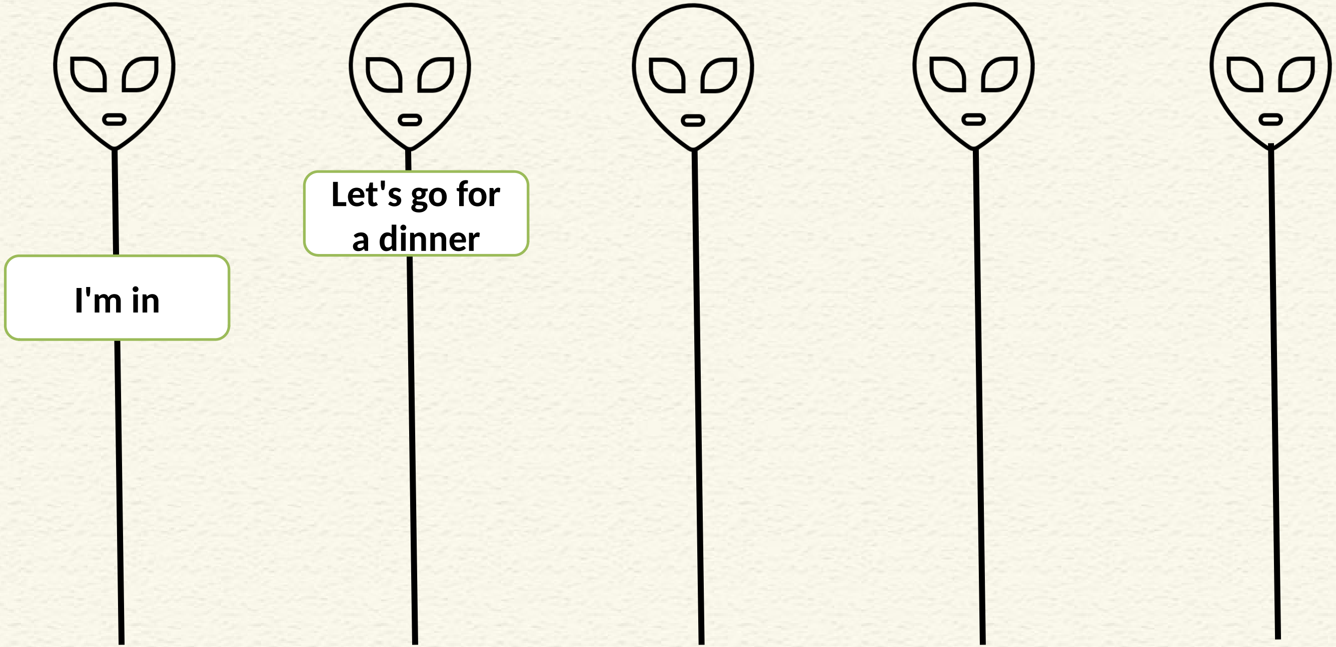
CFT in a Synchronous System



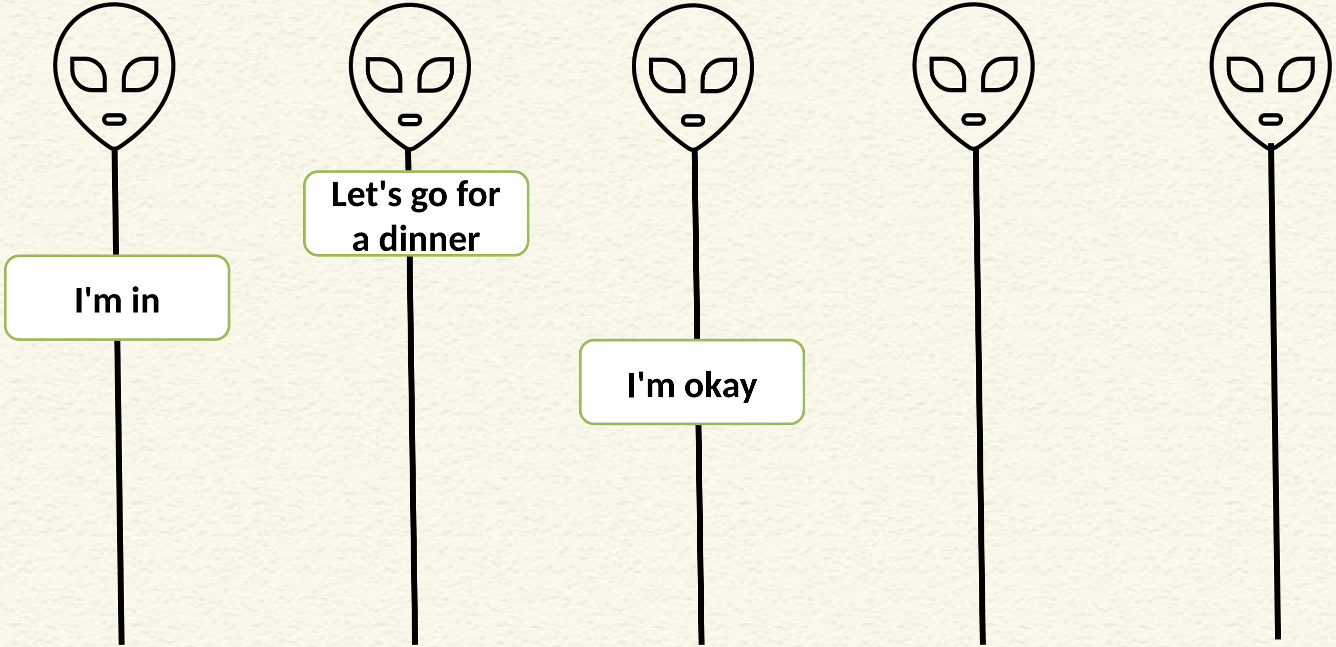
CFT in an Asynchronous System



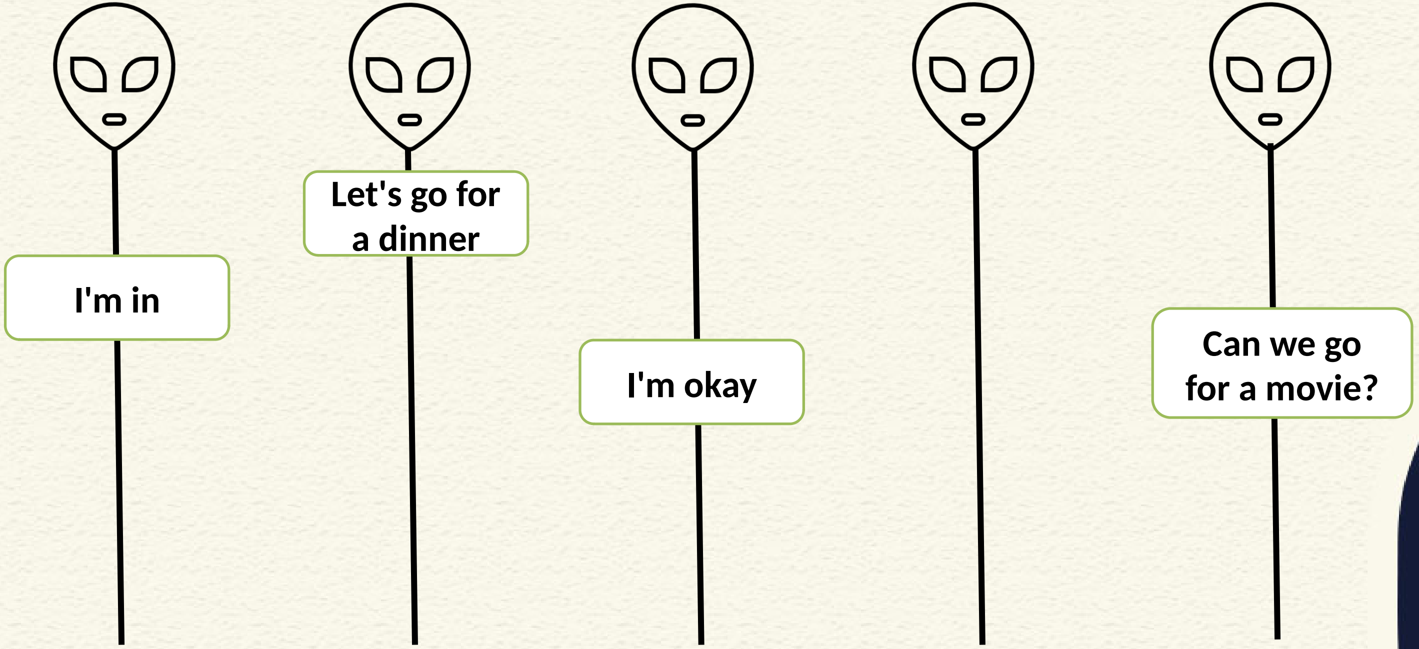
CFT in an Asynchronous System



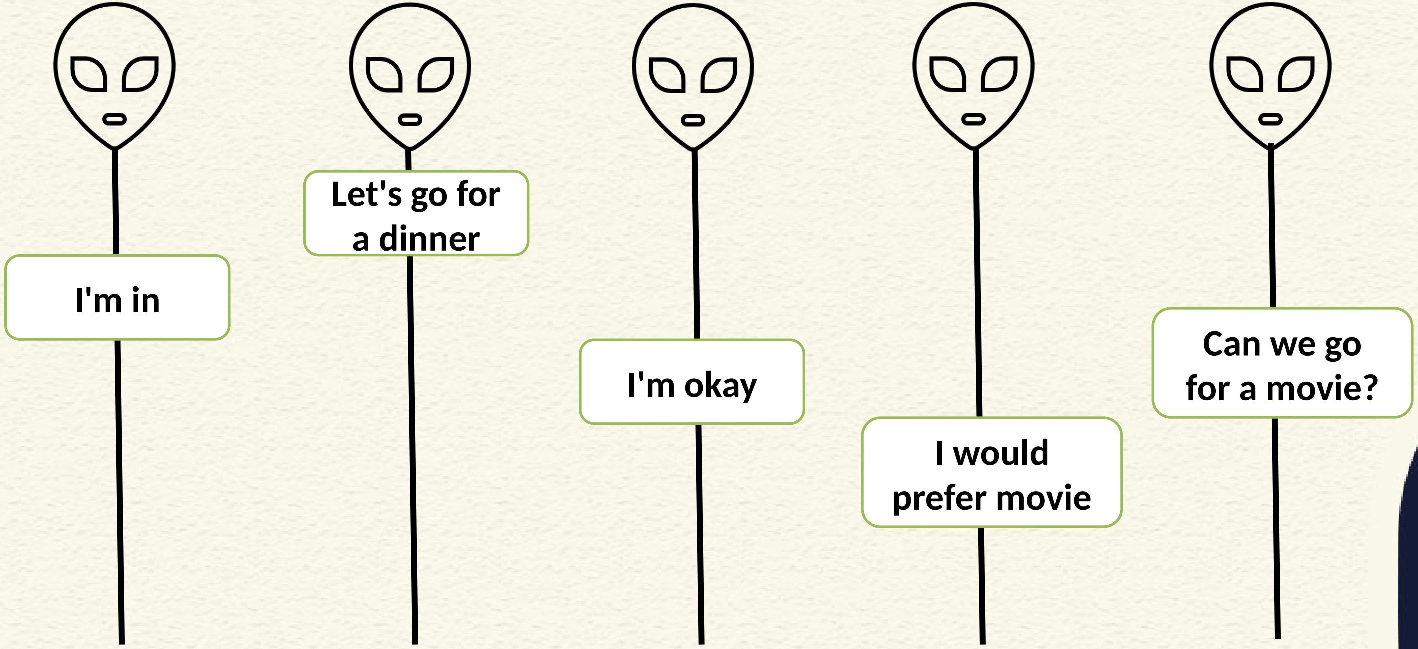
CFT in an Asynchronous System



CFT in an Asynchronous System



CFT in an Asynchronous System



I'm in

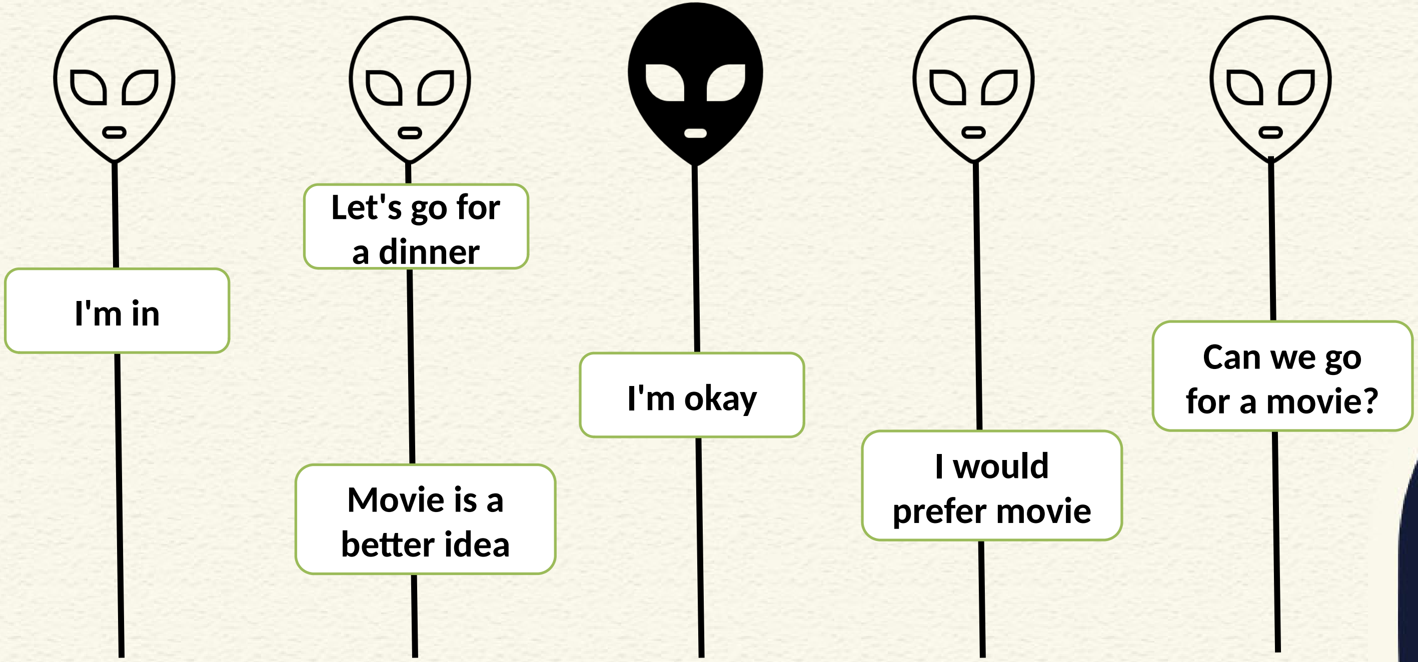
Let's go for a dinner

I'm okay

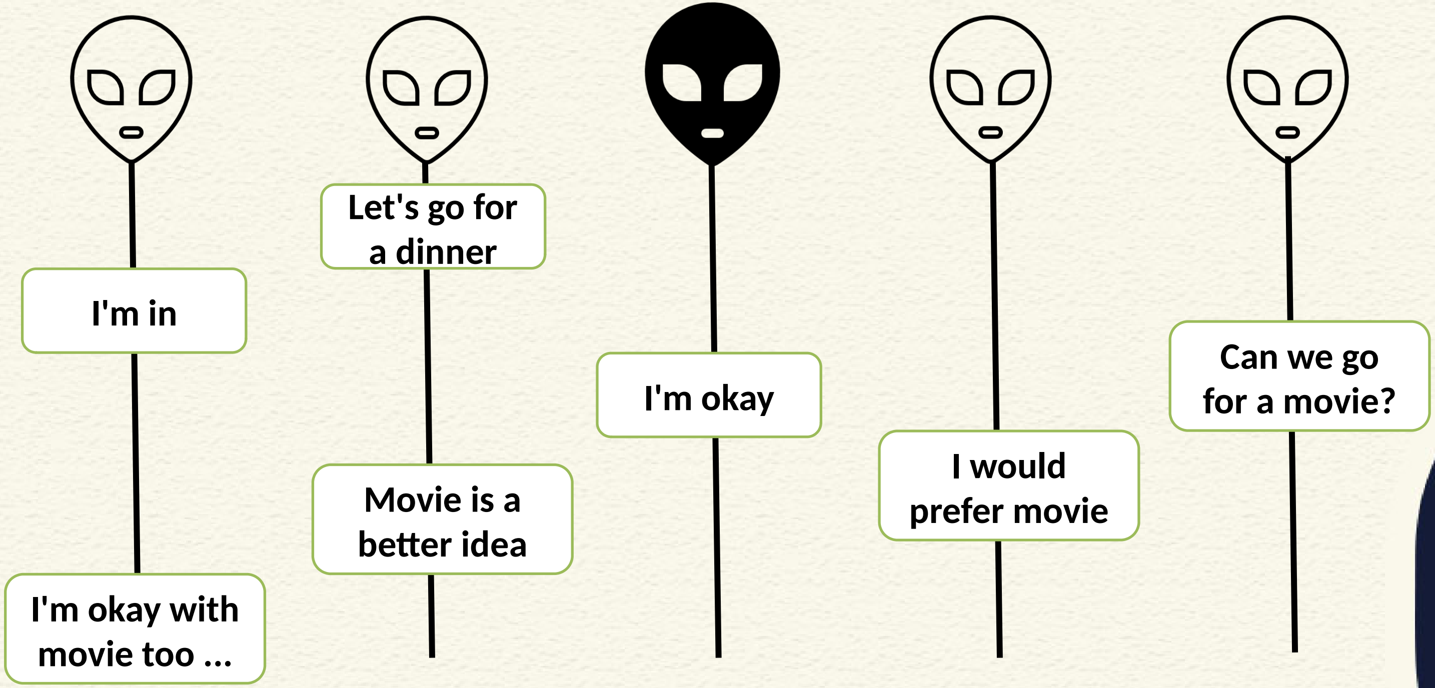
I would prefer movie

Can we go for a movie?

CFT in an Asynchronous System



CFT in an Asynchronous System



I'm in

I'm okay with movie too ...

Let's go for a dinner

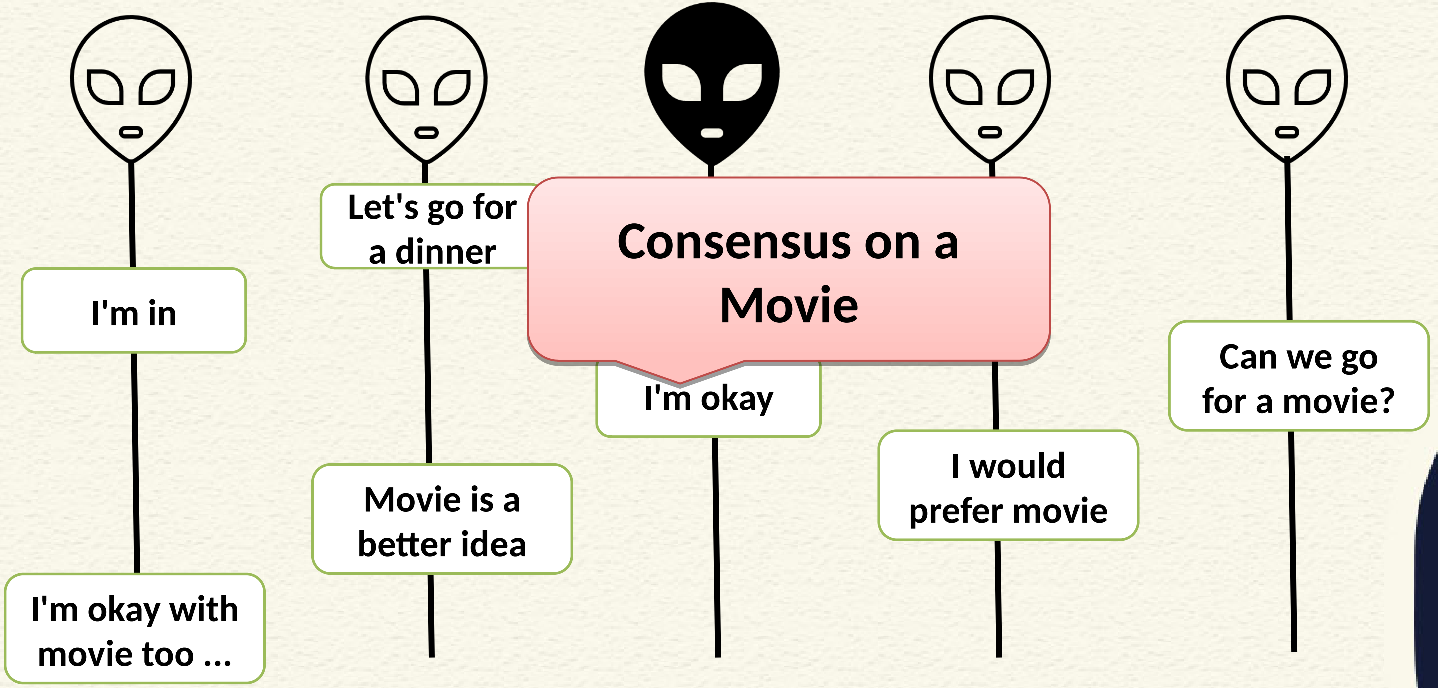
Movie is a better idea

I'm okay

I would prefer movie

Can we go for a movie?

CFT in an Asynchronous System



I'm in

I'm okay with movie too ...

Let's go for a dinner

Movie is a better idea

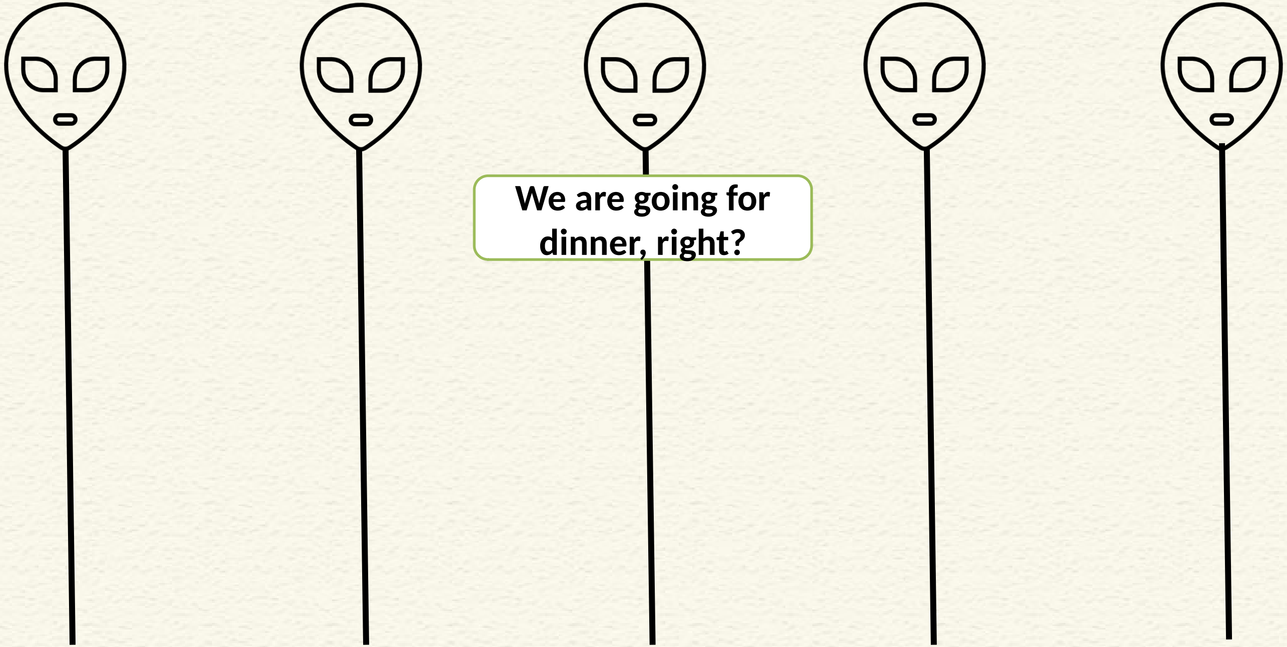
I'm okay

I would prefer movie

Can we go for a movie?

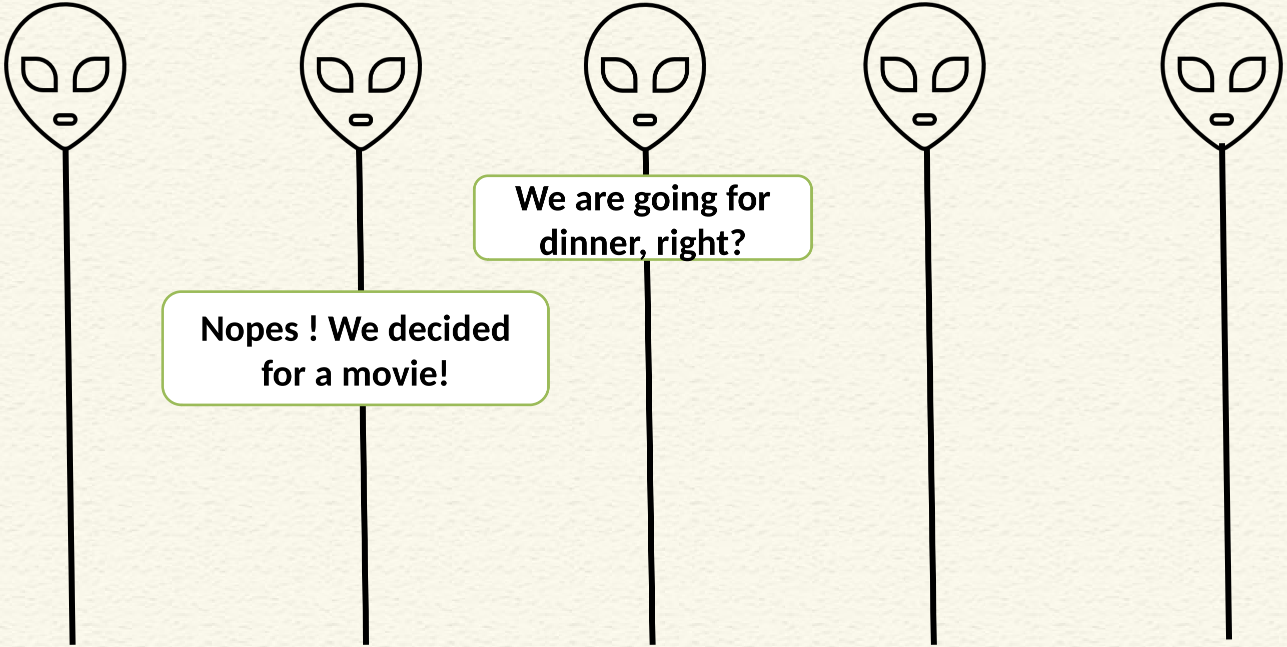
Consensus on a Movie

CFT in an Asynchronous System



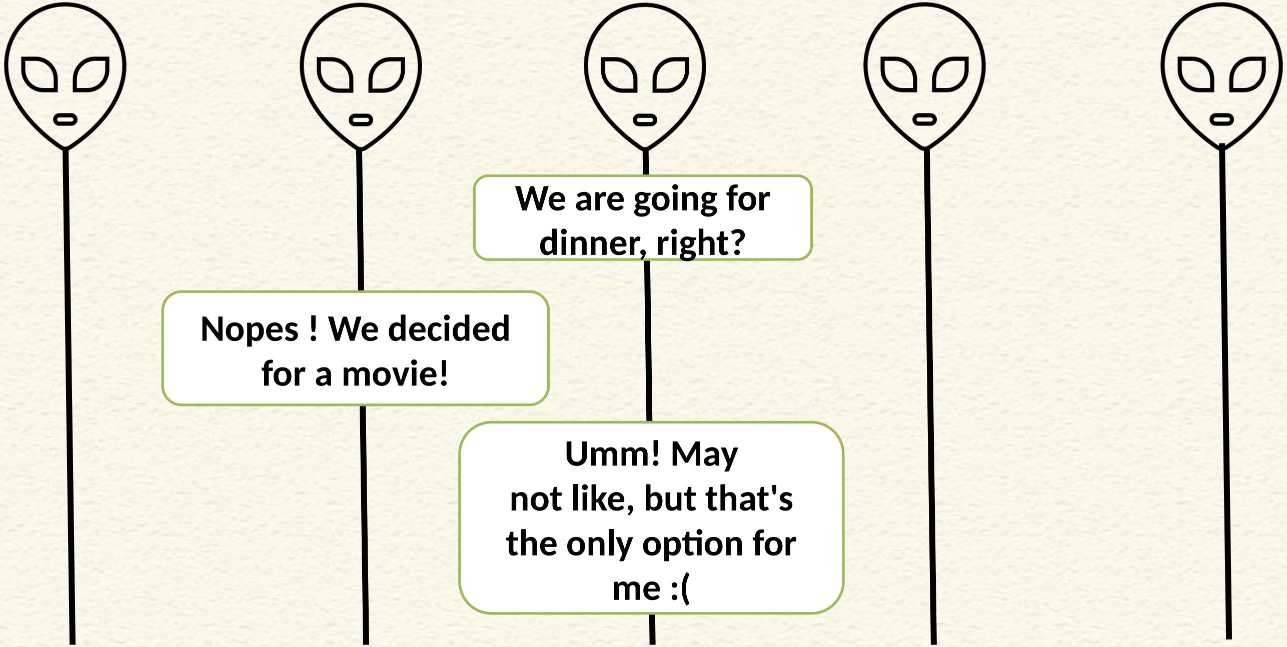
We are going for dinner, right?

CFT in an Asynchronous System



We are going for dinner, right?
Nopes ! We decided for a movie!

CFT in an Asynchronous System

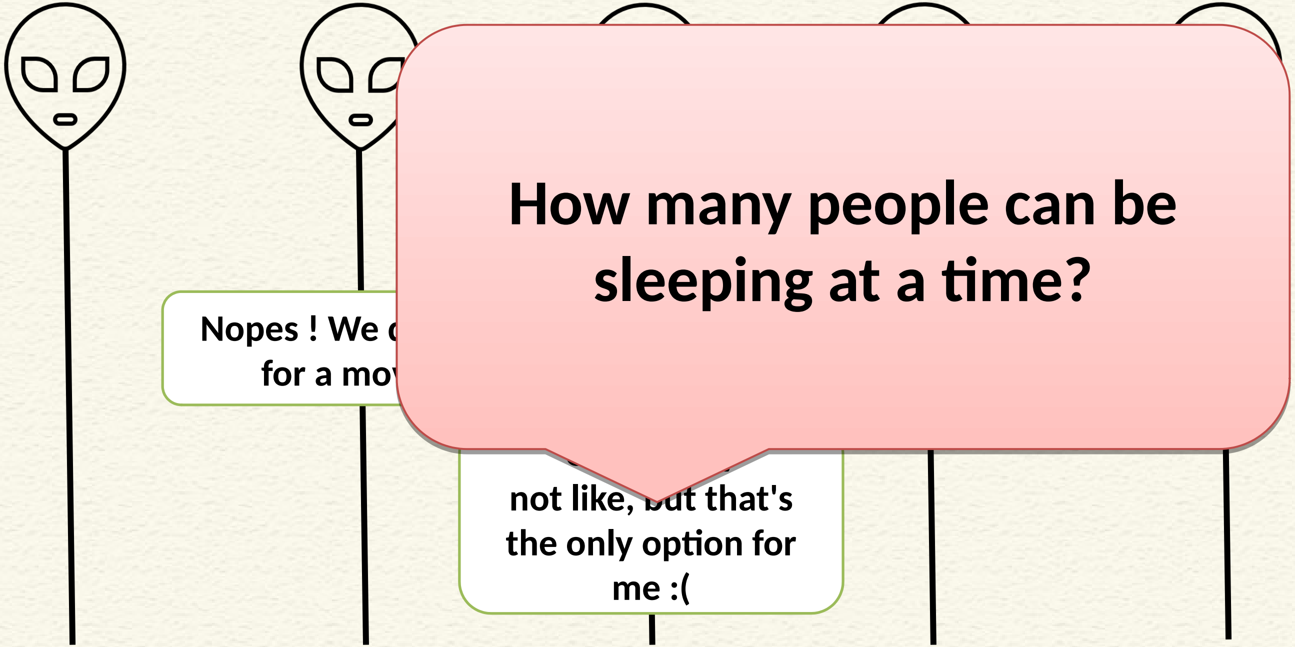


We are going for dinner, right?

Nopes ! We decided for a movie!

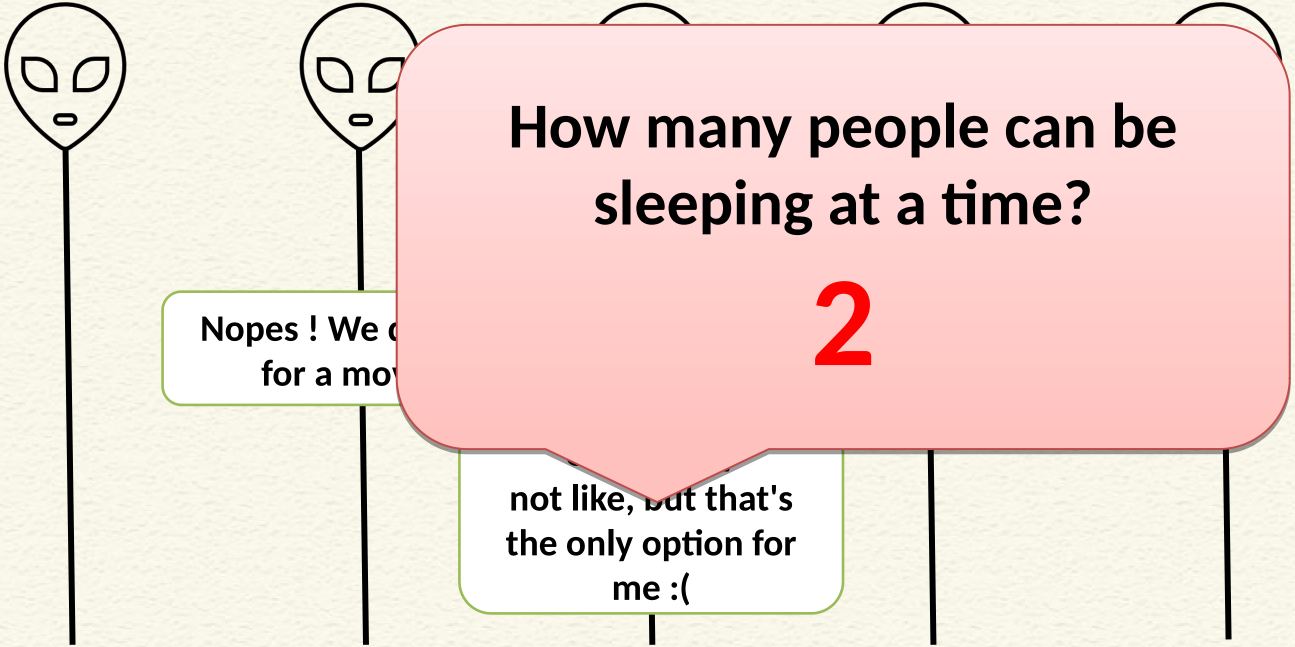
Umm! May not like, but that's the only option for me :(

CFT in an Asynchronous System



How many people can be sleeping at a time?

CFT in an Asynchronous System



How many people can be sleeping at a time?

2

Asynchronous CFT

- If there are F faulty nodes (crash fault), we need atleast 2F+1 nodes to reach consensus

- **Paxos:** A family of distributed algorithms to reach consensus in an asynchronous CFT

What is Paxos?

- We'll discuss vanilla Paxos
- Proposed by Lamport in 1989
- Received a lot of criticism about its proof of correctness
- Accepted in ACM Transactions on Computer Systems in 1998, titled *"The Part-time Parliament"*
- Lamport received the Turing award in 2013

Conclusion

- Consensus is harder on asynchronous environment
- For asynchronous CFT, we need 2F+1 nodes with F crash faults only
- Let's explore Paxos in the next class

Thank you