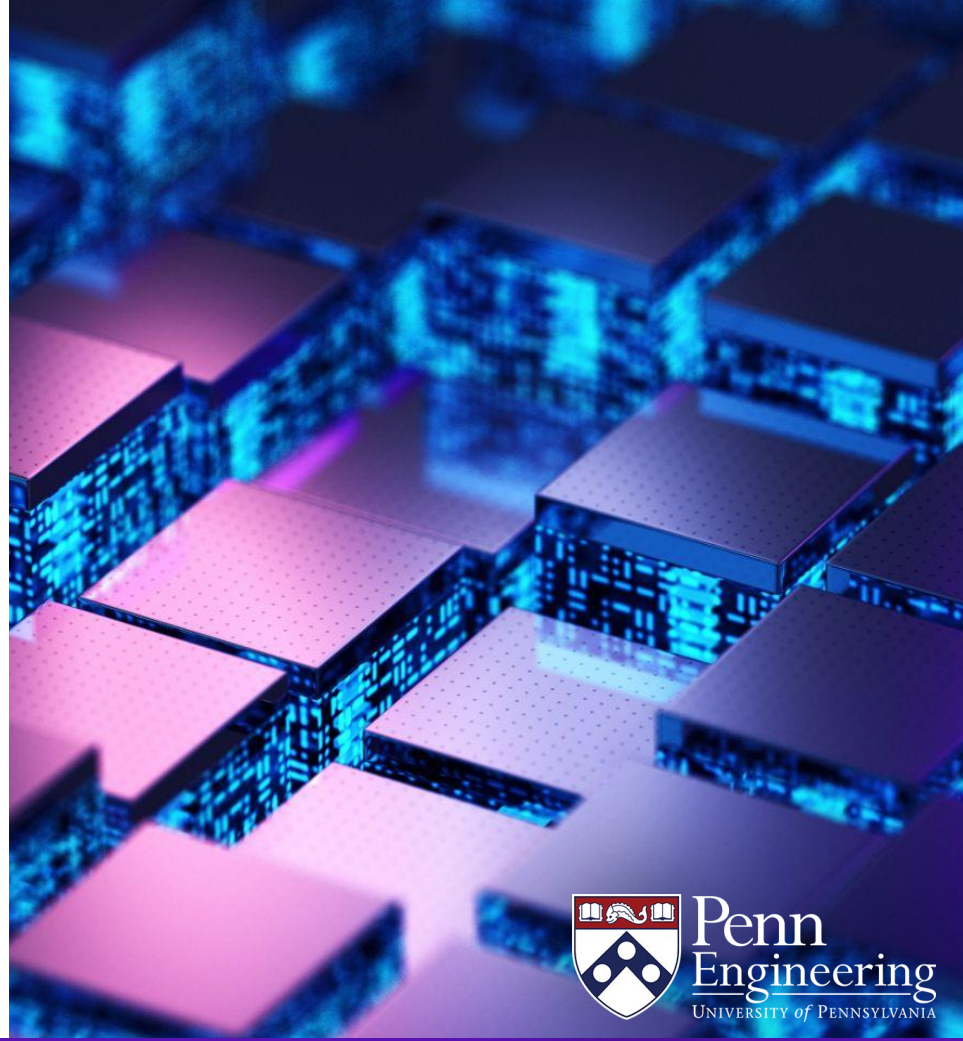
EAS 5830: BLOCKCHAINS

Classical Consensus

Dr. Brett Hemenway Falk



Byzantine Agreement

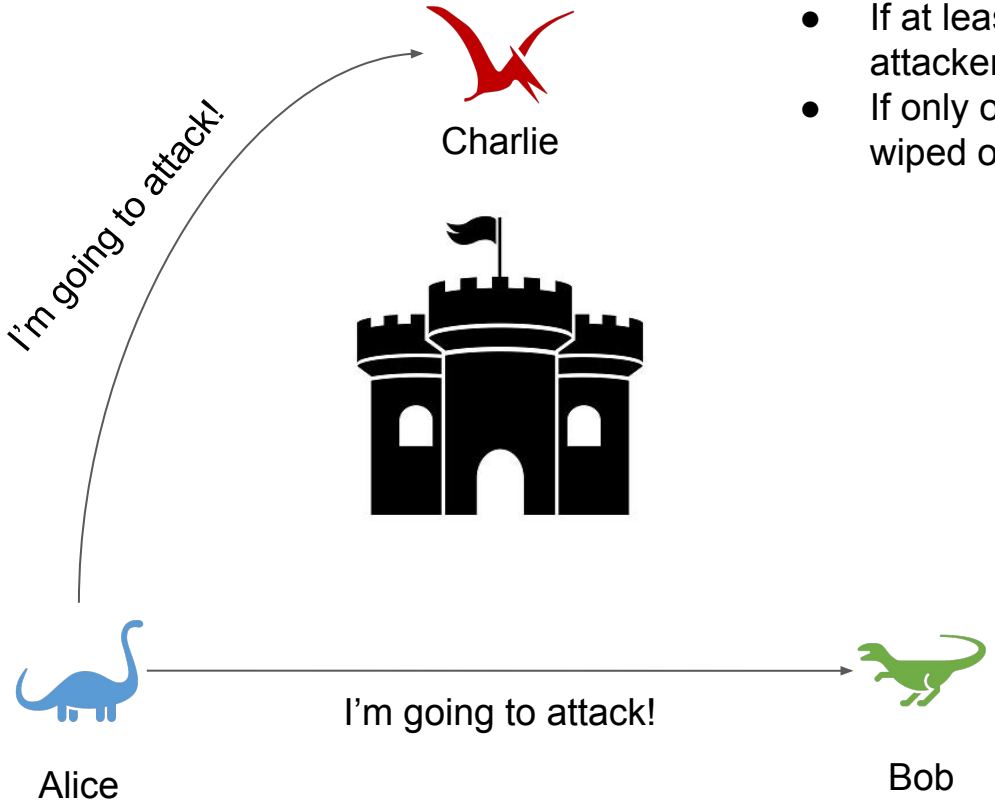
Charlie

- If at least 2-out-of-3 attack, attackers win
- If only one attacks, attacker is wiped out

Alice

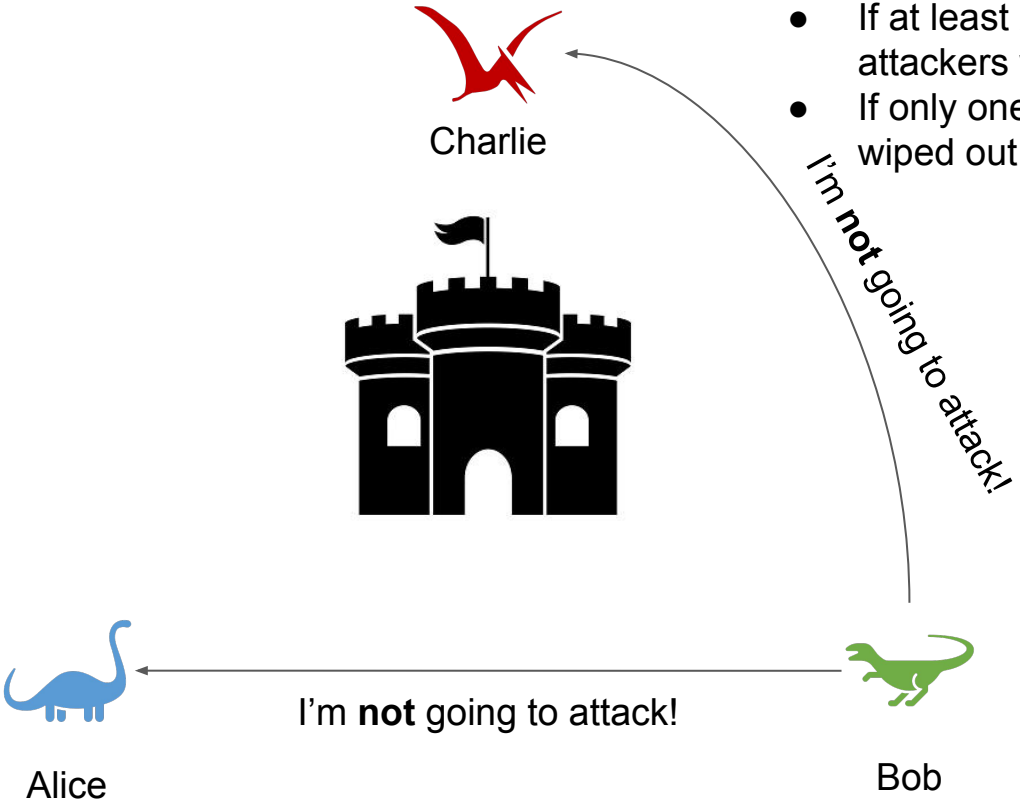
Bob

Byzantine Agreement



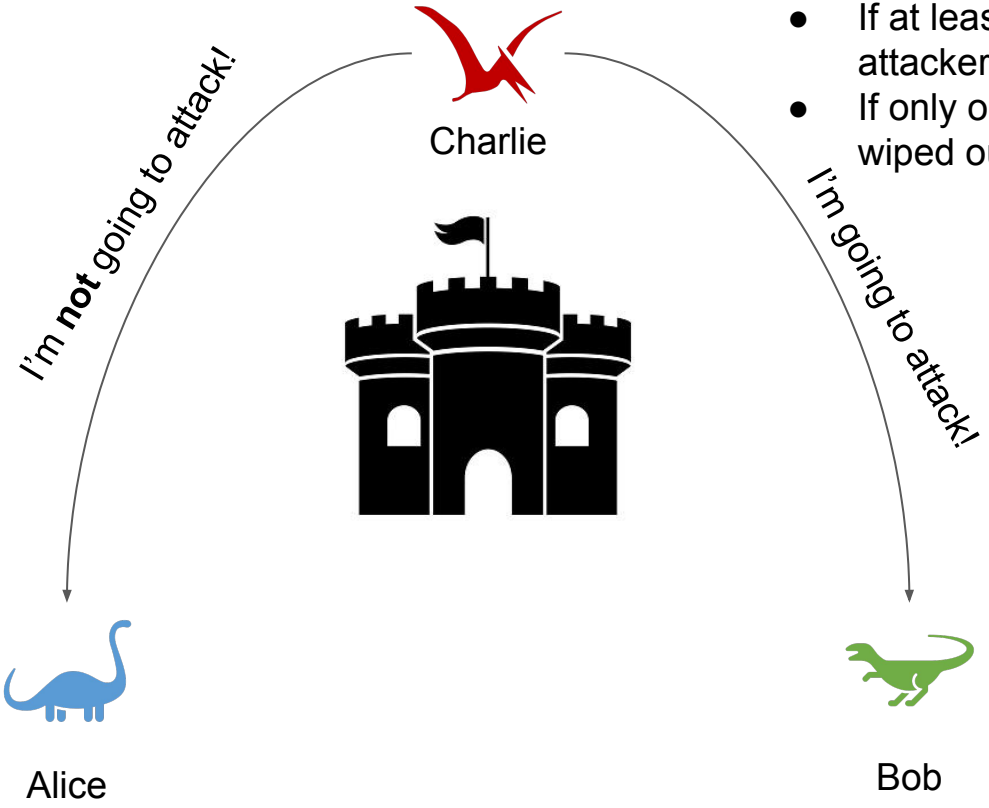
- If at least 2-out-of-3 attack, attackers win
- If only one attacks, attacker is wiped out

Byzantine Agreement



- If at least 2-out-of-3 attack, attackers win
- If only one attacks, attacker is wiped out

Byzantine Agreement



- If at least 2-out-of-3 attack, attackers win
- If only one attacks, attacker is wiped out


BERNSTEIN

Byzantine Agreement is Equivalent to Broadcast

- Broadcast implies BA
  - Every node broadcasts their value
  - Each node takes majority vote
  - Broadcast ensures that participants can't equivocate
- BA implies Broadcast
  - To broadcast, sender sends message to each node individually
  - Nodes run BA to agree on the message from the sender
  - This ensures everyone "received" the same message

Consensus

- Setup:
  - n participants
    - All connected with point-to-point channels
  - Participant i has a value v_i
- Properties:
  - **Termination**
    - All non-faulty processes eventually output some value
  - **Agreement**
    - At the end of the protocol, all honest nodes output the same value
  - **Consistency**
    - If all honest nodes began with the same value, then this is the value they agree upon

FLP Result

No deterministic 1-**crash-**robust consensus algorithm exists with **asynchronous** communication

Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

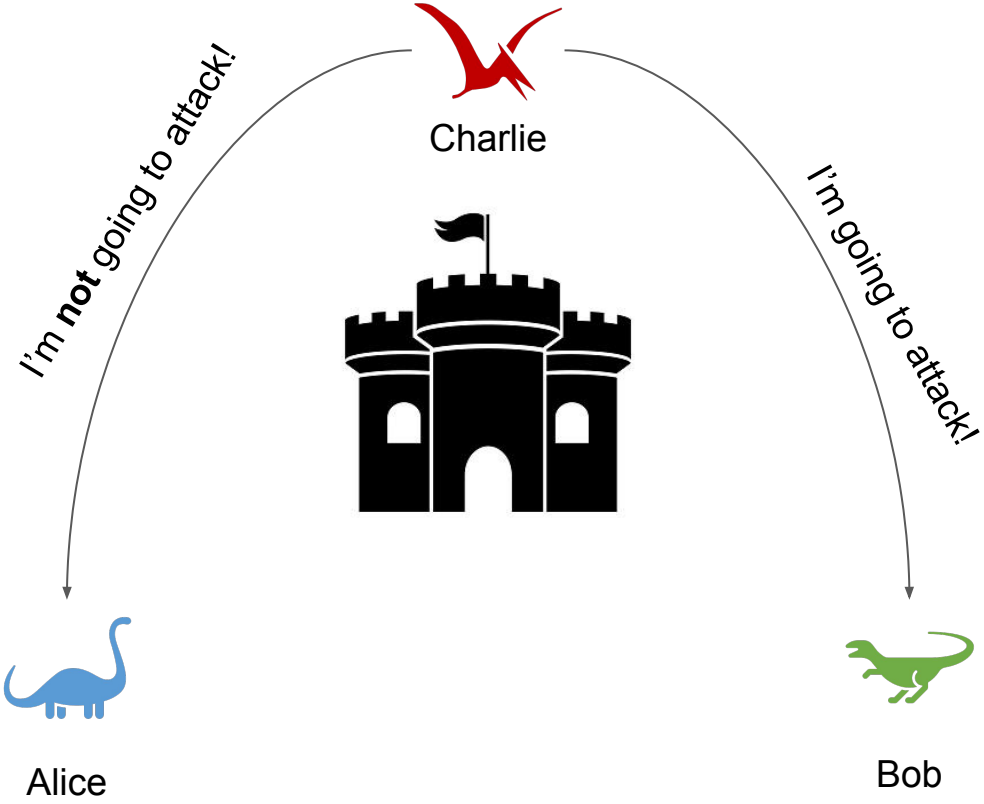
Abstract. The consensus problem involves an asynchronous system of processes, some of which may be unreliable. The problem is for the reliable processes to agree on a binary value. In this paper, it is shown that every protocol for this problem has the possibility of nontermination, even with only one faulty process. By way of contrast, solutions are known for the synchronous case, the "Byzantine Generals" problem.

This work was originally presented at the 2nd ACM Symposium on Principles of Database Systems, March 1983.

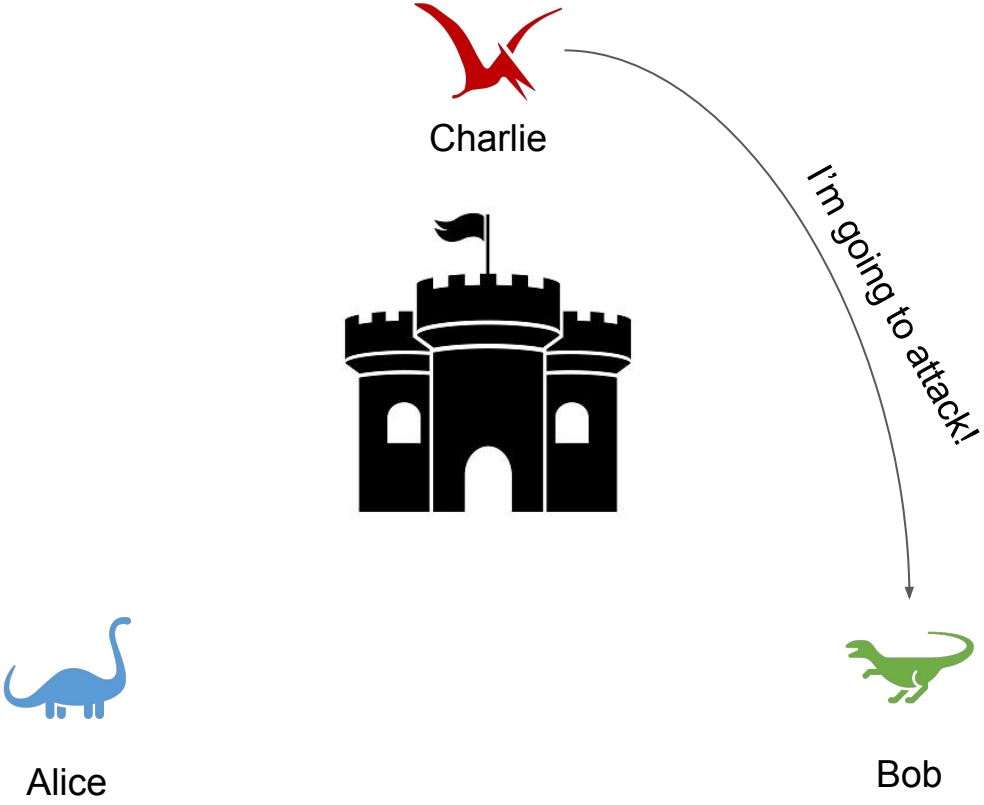
Authors' present addresses: M. J. Fischer, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520; N. A. Lynch, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139; M. S. Paterson, Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

Journal of the Association for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374–382.

Byzantine Agreement

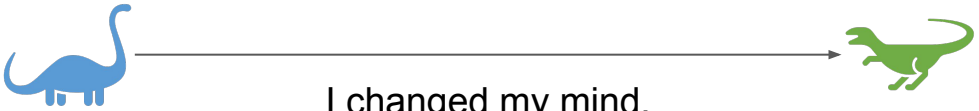


Byzantine Agreement



Byzantine Agreement

Charlie

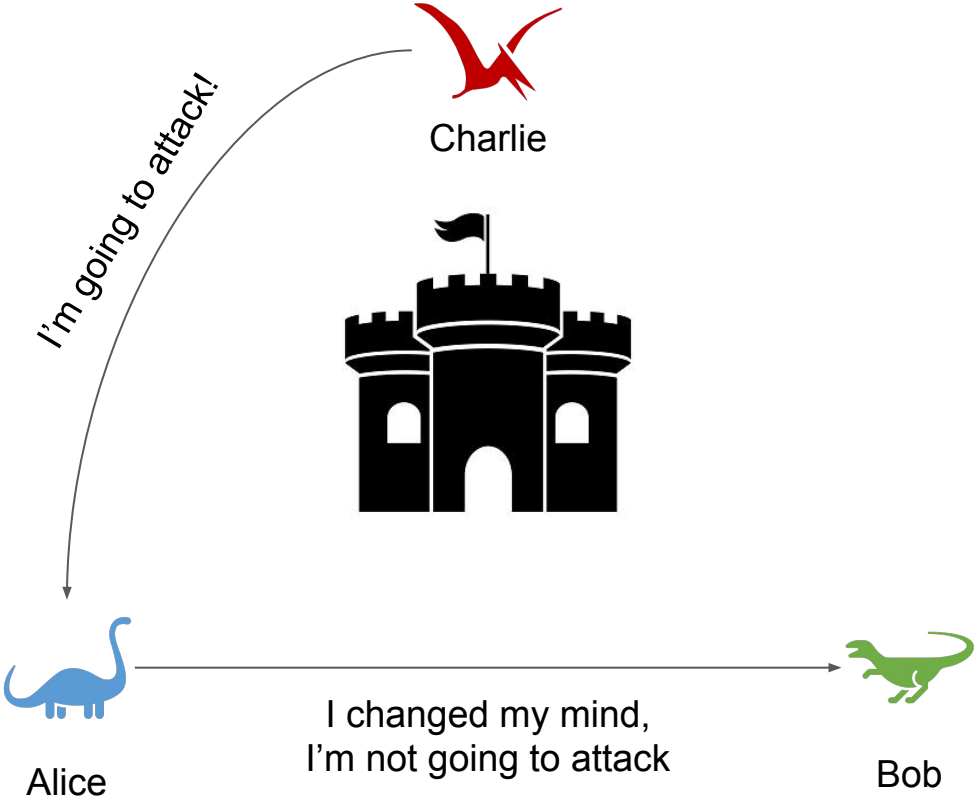


I changed my mind,
I'm not going to attack

Alice

Bob

Byzantine Agreement



FLP Result

In an asynchronous system where there are no bounds on the amount of time a processor might take to complete its work it's impossible to make the distinction between a process that is crashed and one that is taking a long time to respond.

Circumventing the FLP result

- Sacrificing determinism
  - Making the protocol randomized
    - Small chance the protocol fails to terminate?
    - Small chance the protocol fails to satisfy agreement or consistency
- Sacrificing asynchrony
  - Make the network synchronous
    - Assume the nodes have a synchronized clock
    - Protocol proceeds in “rounds”
    - Every node knows which round it’s in
    - If a node is supposed to send a message in a given round, but it doesn’t you can assume it’s dead

Practical Byzantine Fault Tolerance

Miguel Castro and Barbara Liskov
Laboratory for Computer Science,
Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139
{castro,liskov}@lcs.mit.edu

PBFT

- PBFT solves the consensus problem when there are at most f faulty nodes and at least $n = 3f + 1$ nodes total
  - Requires 2/3rds honest nodes
- Requires synchrony
- Read more on PBFT

Permissionless consensus

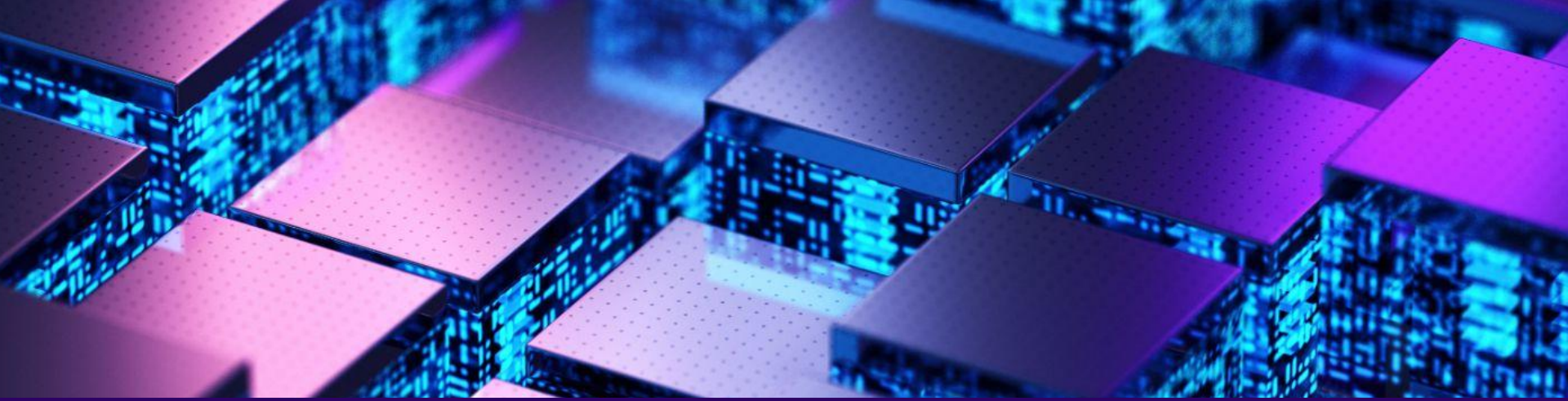
- Classical consensus algorithms assume you know the identity of everyone involved
- What about “permissionless” consensus?

Permissionless consensus

- Classical consensus algorithms assume you know the identity of everyone involved
- What about “permissionless” consensus?
  - Bitcoin

Permissionless consensus

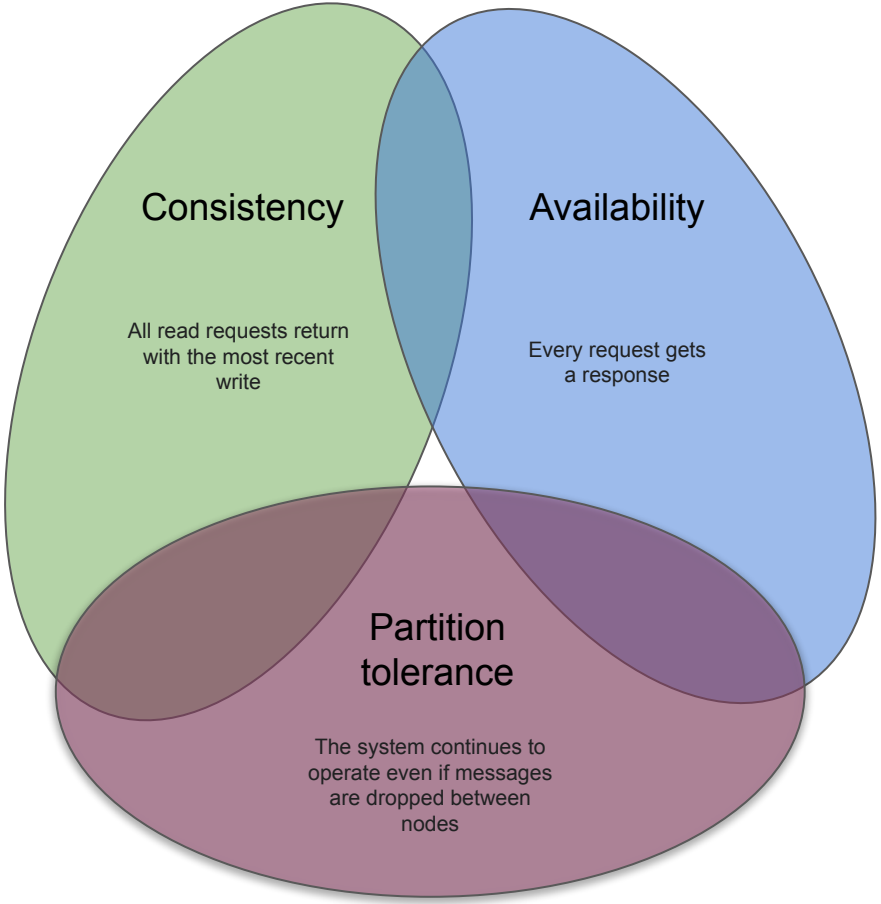
- Classical consensus algorithms assume you know the identity of everyone involved
- What about “permissionless” consensus?
  - Bitcoin
  - “Committee-based consensus”



The CAP Theorem

CAP Theorem

- CAP Theorem says you can't have all three

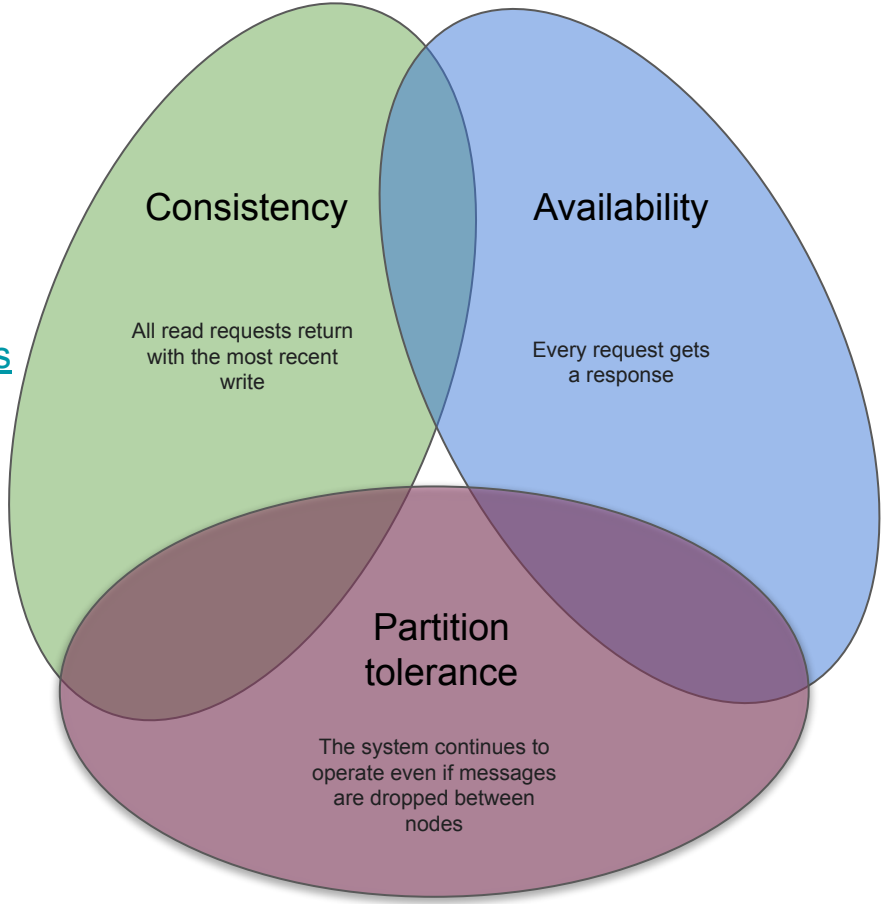


Consensus

- **Termination**
  - All non-faulty processes eventually output some value
- **Agreement**
  - At the end of the protocol, all honest nodes output the same value
- **Consistency**
  - If all honest nodes began with the same value, then this is the value they agree upon

CAP Theorem

- CAP Theorem says you can't have all three
- CAP model is different from FLP
  - In FLP "fail-stop" crashes
  - Partitions mean arbitrary message loss



Blockchain language

- CAP Theorem can be viewed as a Safety / Liveness tradeoff
  - Bitcoin prioritized liveness
  - Cosmos chains prioritize safety