CS4740
CLOUD COMPUTING

Agreement

Prof. Chang Lou, UVA CS, Fall 2025

AGREEMENT

- A set of nodes in a DS often need to agree on something: a decision, the value of a variable, order of events,...
- Example: ATM machine
  - ATM front-end and banking service need to agree on whether to commit or abort my cash withdrawal.

GAME: GREEN CUP, RED CUP

- Three students stand in a line
  - Cup on the head decides the role. Green: **Leader**, Red: **Follower**
  - Can only see the color of cups in front of them
  - Cannot talk with each other or turn around
  - Cannot move in the first 30 seconds
- **Win**: Leader gives me a high-five within one minute after start.
- **Lose**: Leader didn't give me a high-five in time, or followers take moves instead
- Can you design a protocol to win?

TWO TYPES OF AGREEMENT

- **Consensus**: participants need to agree on a value, but they are willing and capable to accept any value.
- **Atomic commitment**: participants need to agree on a value, but they have specific constraints on whether they can accept any particular value.

- Give some examples?

QUIZ

- Question:
  - Decision of when to meet is likely ??? problem.
  - Decision of which zoom link to meet at is likely ??? problem.
- Answer:
  - Decision of when to meet is likely an atomic commitment problem.
  - Decision of which zoom link to meet at is likely a consensus problem.

EXAMPLES – WHICH TYPE IS EACH?

- Lamport's distributed mutual exclusion protocol: nodes agree on who has the lock at any time. ← ???
- ATM example from RPC lecture: ATM front-end and banking service need to agree on whether to commit or abort my cash withdrawal. ← ???
- In Lab1, you design a MapReduce system that all workers agree whether they are in the map or reduce phase. ← ???

EXAMPLES – WHICH TYPE IS EACH?

- Lamport's distributed mutual exclusion protocol: nodes agree on who has the lock at any time. ← Consensus
- ATM example from RPC lecture: ATM front-end and banking service need to agree on whether to commit or abort my cash withdrawal. ← Atomic commitment
- In Lab1, you design a MapReduce system that all workers agree whether they are in the map or reduce phase. ← Consensus

AGREEMENT IS "HARD"

- In the asynchronous system model, it is impossible to guarantee agreement in finite time under all failure scenarios.
- The consensus problem can be approached in practice: there exist protocols to solve consensus under vast majority of plausible failure scenarios.
- That's not the case for atomic commitment: if each participant has their own constraints, then you can't tolerate any one participant's failure.
- In that sense, atomic commitment is "even harder" than consensus.

REMAINDER OF THIS CLASS

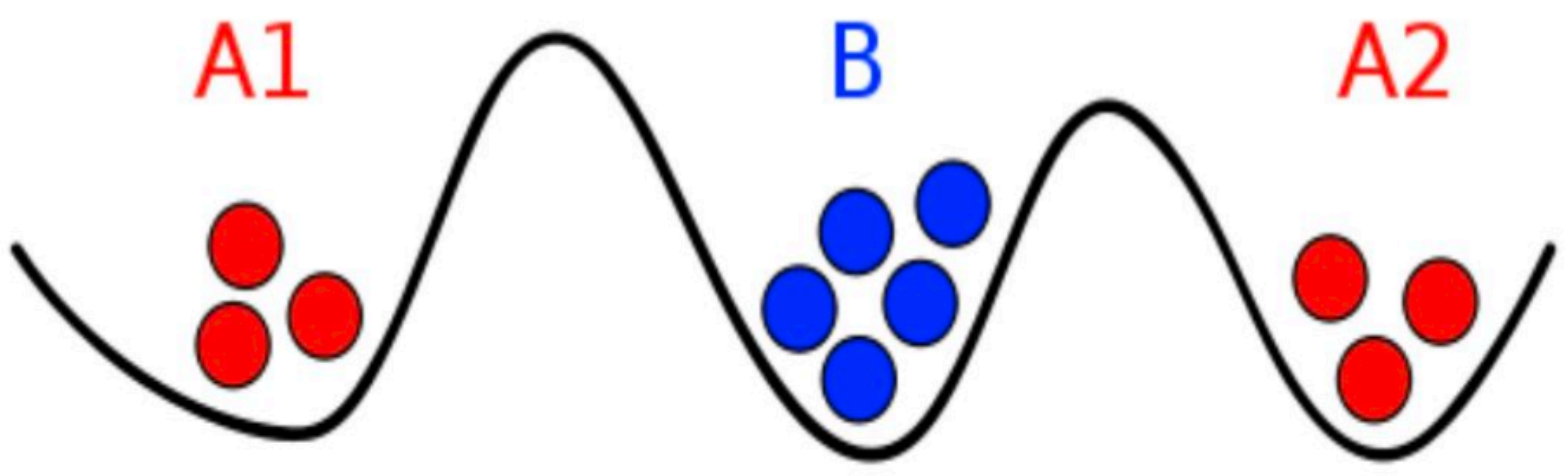
- Focus on two "impossibilities":
  - **FLP**: impossible to have deterministic one-crash-robust consensus with asynchronous communication
  - **CAP**: impossible to achieve consistency, availability and partition-tolerance all

THE FLP IMPOSSIBILITY RESULT

Fischer, Lynch, and Paterson (FLP), 1985

– In an asynchronous system (unordered messages, unbounded communication delays, unbounded processing delays), no protocol can guarantee consensus within a finite amount of time if even a single process can fail by stopping. [FLP-1985]

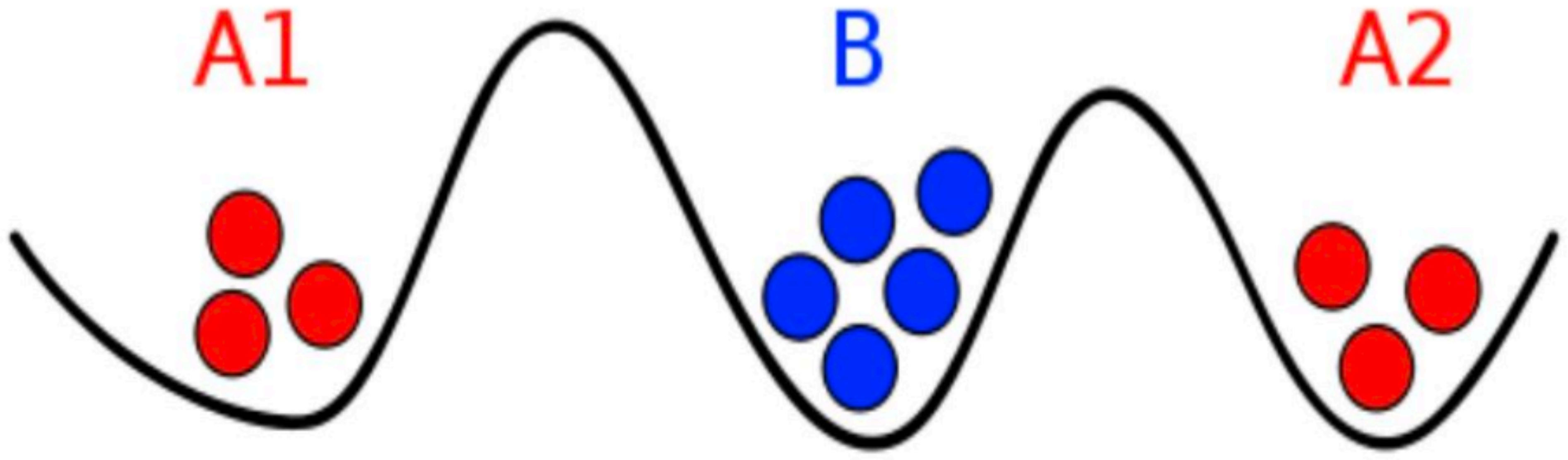
THE TWO-GENERALS PROBLEM



- Two armies, A1 and A2, want to attack a fortified city, B.
- Both armies must attack at the same time to succeed.
- The armies can communicate through messengers, but those can be captured or delayed, so msg. delivery is unreliable.

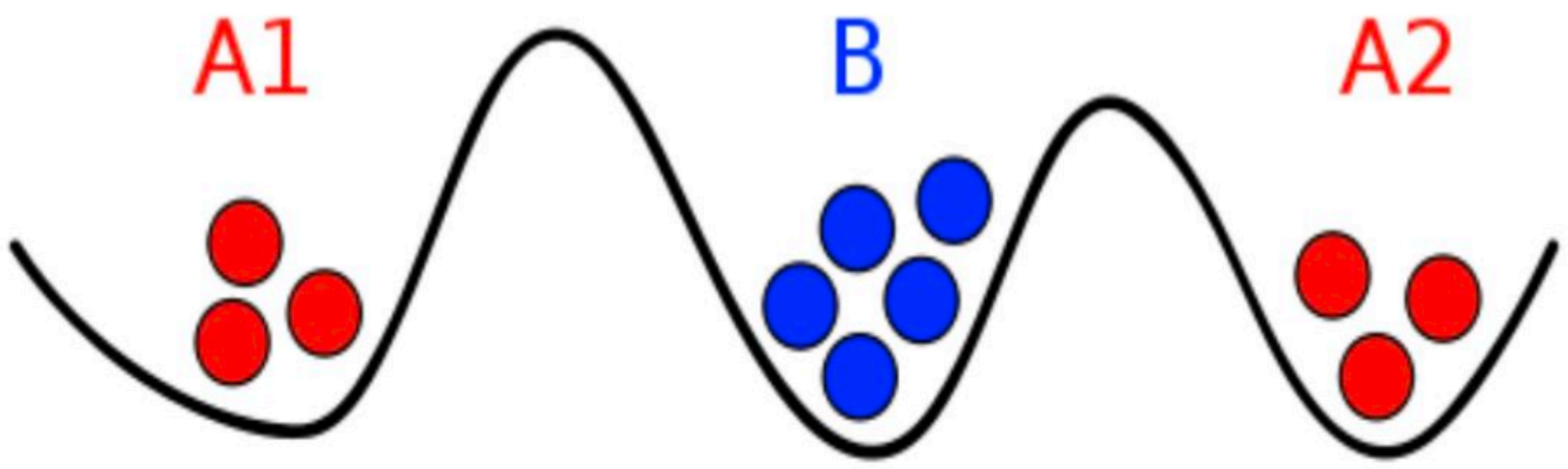
THE TWO-GENERALS PROBLEM

- Three requirements for a solution:
  - Consistency: both armies decide to attack at the same time.
  - Termination: each army decides to attack after a finite number of messages.
  - Validity: the time to attack was proposed by one of the armies.



CASE 1: KNOWN DELAYS, RELIABLE DELIVERY (SYNCHRONOUS SYSTEM MODEL)

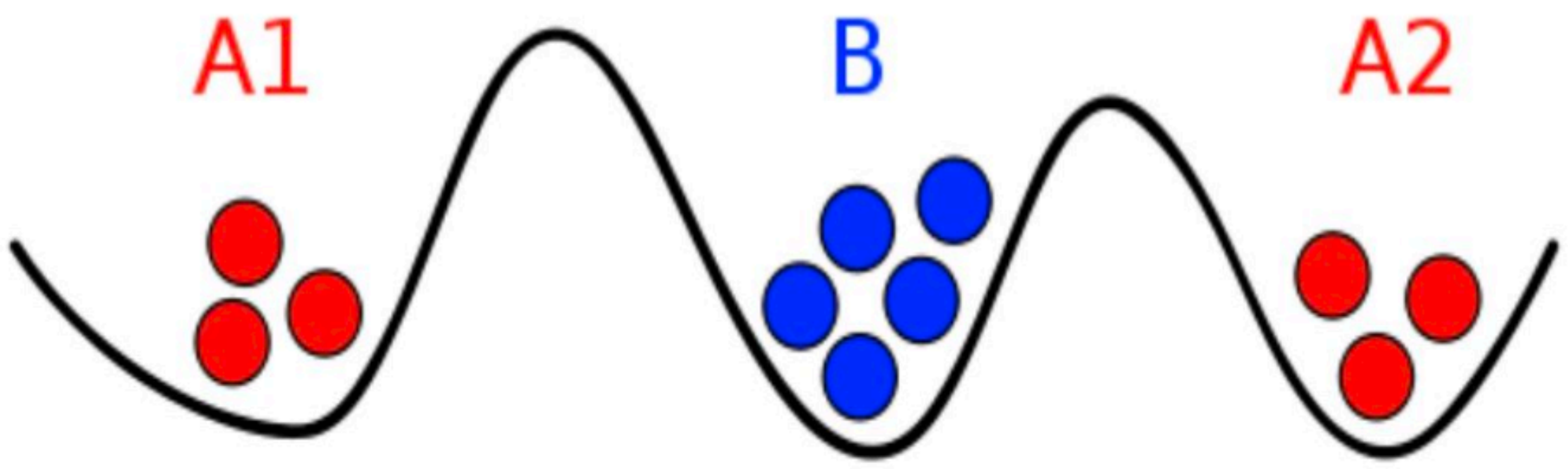
- Protocol:
  - Pre-agree on either A1 or A2 generals proposing the time to attack. Say A1 is the one to propose. A2 will be the one to accept.
  - A1 sets the time of attack to communication delay + some extra time to account for A2's preparation for response.



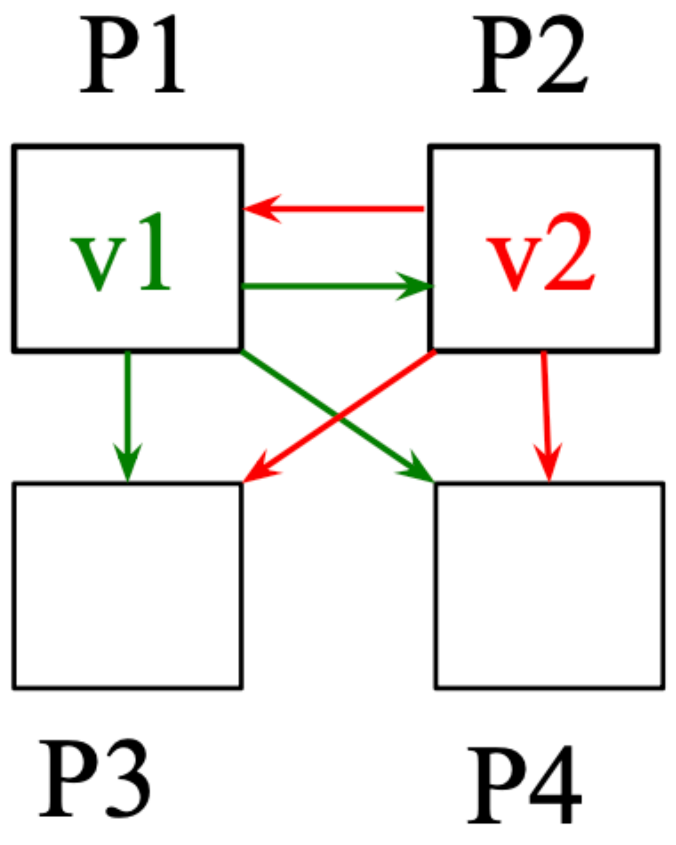
- **So problem is solvable in synchronous networks.**

CASE 2: UNKNOWN DELAYS / UNRELIABLE DELIVERY (ASYNCHRONOUS SYSTEM MODEL)

- Sketch:
  - Need Acks in the protocol.
  - But Acks can be delayed/lost too.
  - Therefore I need more Acks.
  - Therefore, one general can never be sure the other will attack.
  - So they can't be guaranteed to reach agreement.
- Achieving consistency, termination, and validity in the asynchronous model is provably impossible.

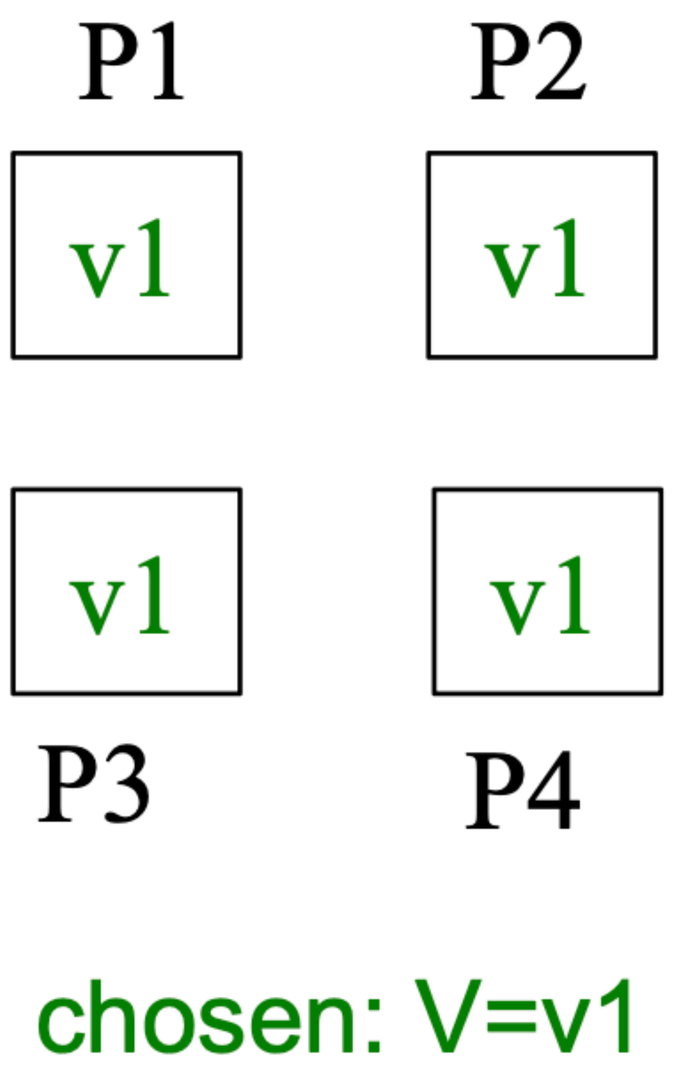


CONSENSUS PROBLEM FORMULATION



- A collection of processes, Pi.
- They propose values Vi (e.g., time to attack, client update, lock requests, ...), and send messages to others to exchange proposals.
- Different processes may propose different values, and they can all accept any of the proposed values.
- Only one of the proposed values, V, will be "chosen" and eventually all processes learn that one chosen value.

CONSENSUS PROBLEM FORMULATION



- A collection of processes, Pi.
- They propose values Vi (e.g., time to attack, client update, lock requests, ...), and send messages to others to exchange proposals.
- Different processes may propose different values, and they can all accept any of the proposed values.
- Only one of the proposed values, V, will be "chosen" and eventually all processes learn that one chosen value.

CONSENSUS PROBLEM FORMULATION

- Three requirements for a solution:
  - consistency:

CONSENSUS PROBLEM FORMULATION

- Three requirements for a solution:
  - consistency: once a value is chosen, the chosen value of all working processes is the same.
  - termination:

CONSENSUS PROBLEM FORMULATION

- Three requirements for a solution:
  - consistency: once a value is chosen, the chosen value of all working processes is the same.
  - termination: eventually they agree on a value (a.k.a., a value is “chosen”).
  - validity:

CONSENSUS PROBLEM FORMULATION

- Three requirements for a solution:
  - consistency: once a value is chosen, the chosen value of all working processes is the same.
  - termination: eventually they agree on a value (a.k.a., a value is “chosen”).
  - validity: the chosen value was proposed by one of the nodes.

CONSENSUS IS IMPOSSIBLE

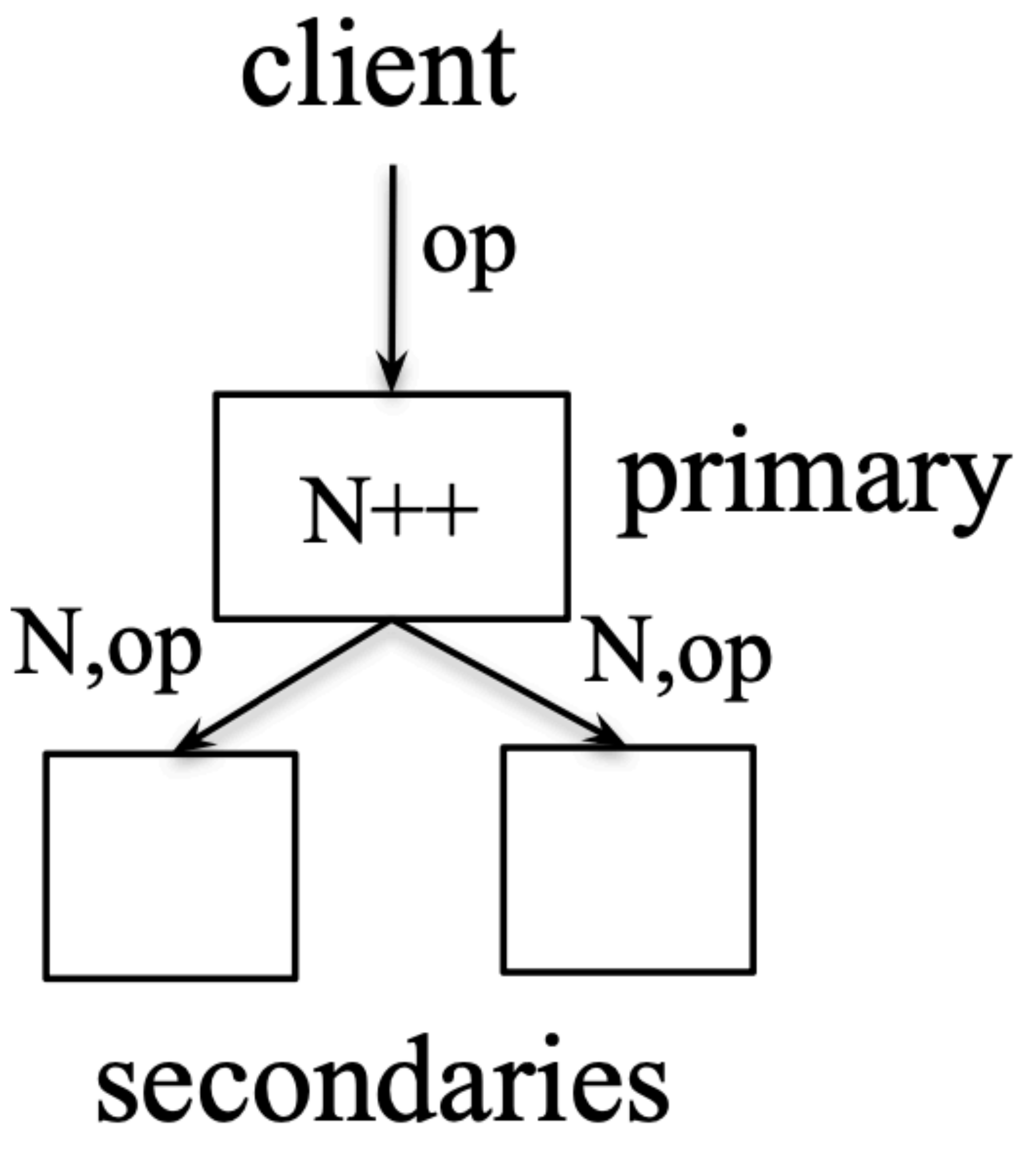
- But, we achieve consensus all the time…

FLP'S STRONG ASSUMPTIONS

- Deterministic actions at each node
  - Randomized algorithms can achieve consensus
- Asynchronous network communication
  - Synchronous or even partial synchrony is sufficient
- All "runs" must eventually achieve consensus
  - In practice, many "runs" achieve consensus quickly
  - In practice, "runs" that never achieve consensus happen vanishingly rarely
  - Both are true with good system designs

CONSENSUS IS PERVASIVE IN DS

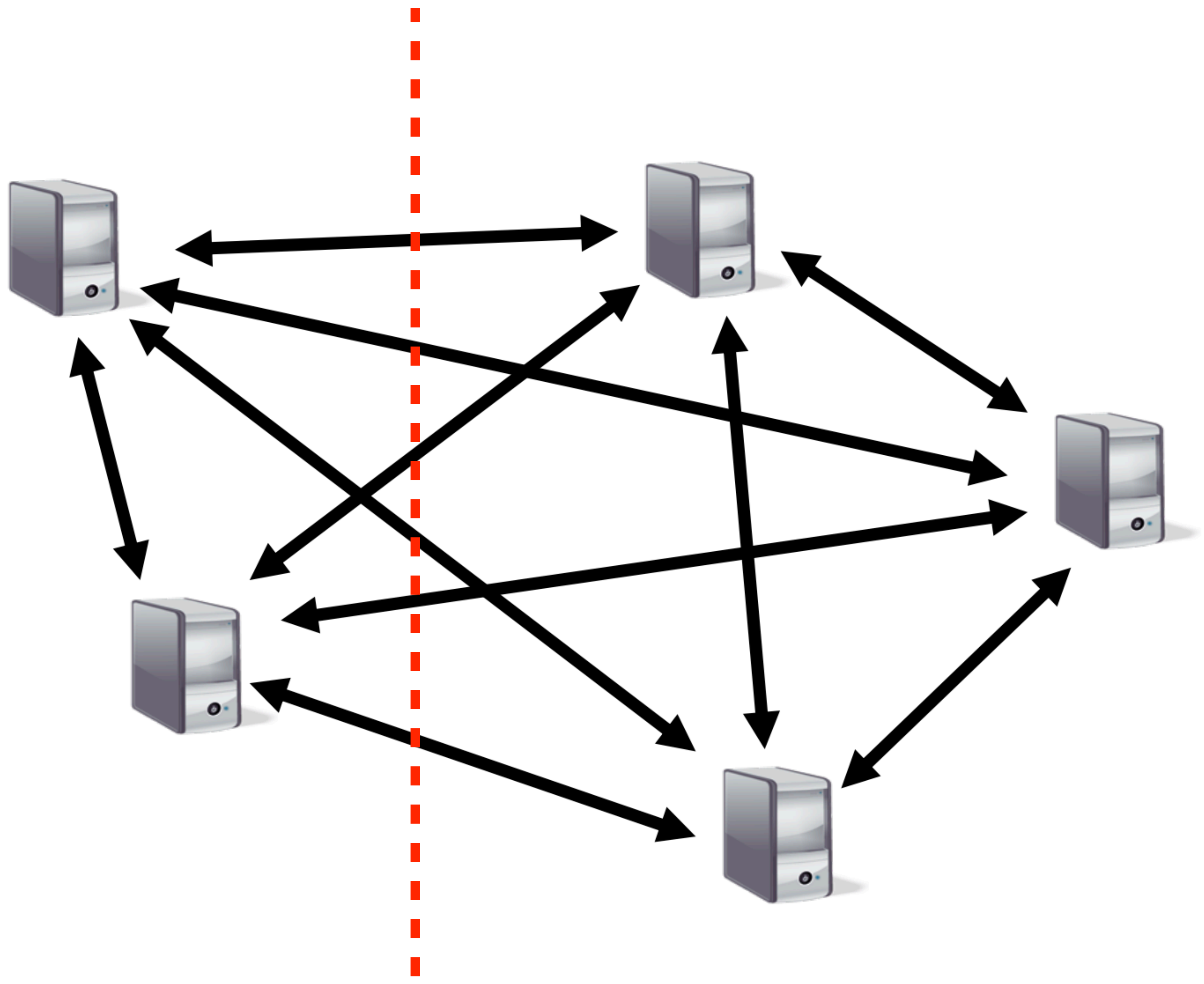
- Agreeing on order of updates to replicated DB.
  - One solution is primary/secondaries replication
  - There are several replicas, one is primary.
  - Reads and writes are accepted only by primary, which establishes an order for all operations before forwarding them to secondaries.
  - Multiple variants exist, but they all reduce to one core consensus question: how to choose the primary? A.k.a. leader election.



IMPOSSIBILITY #2

- Reaching an agreement, when there's a partition

NETWORK PARTITIONS DIVIDE SYSTEMS



FUNDAMENTAL TRADEOFF?

- Replicas appear to be a single machine, but lose availability during a network partition
- or
- All replicas remain available during a network partition but do not appear to be a single machine

CAP THEOREM PREVIEW

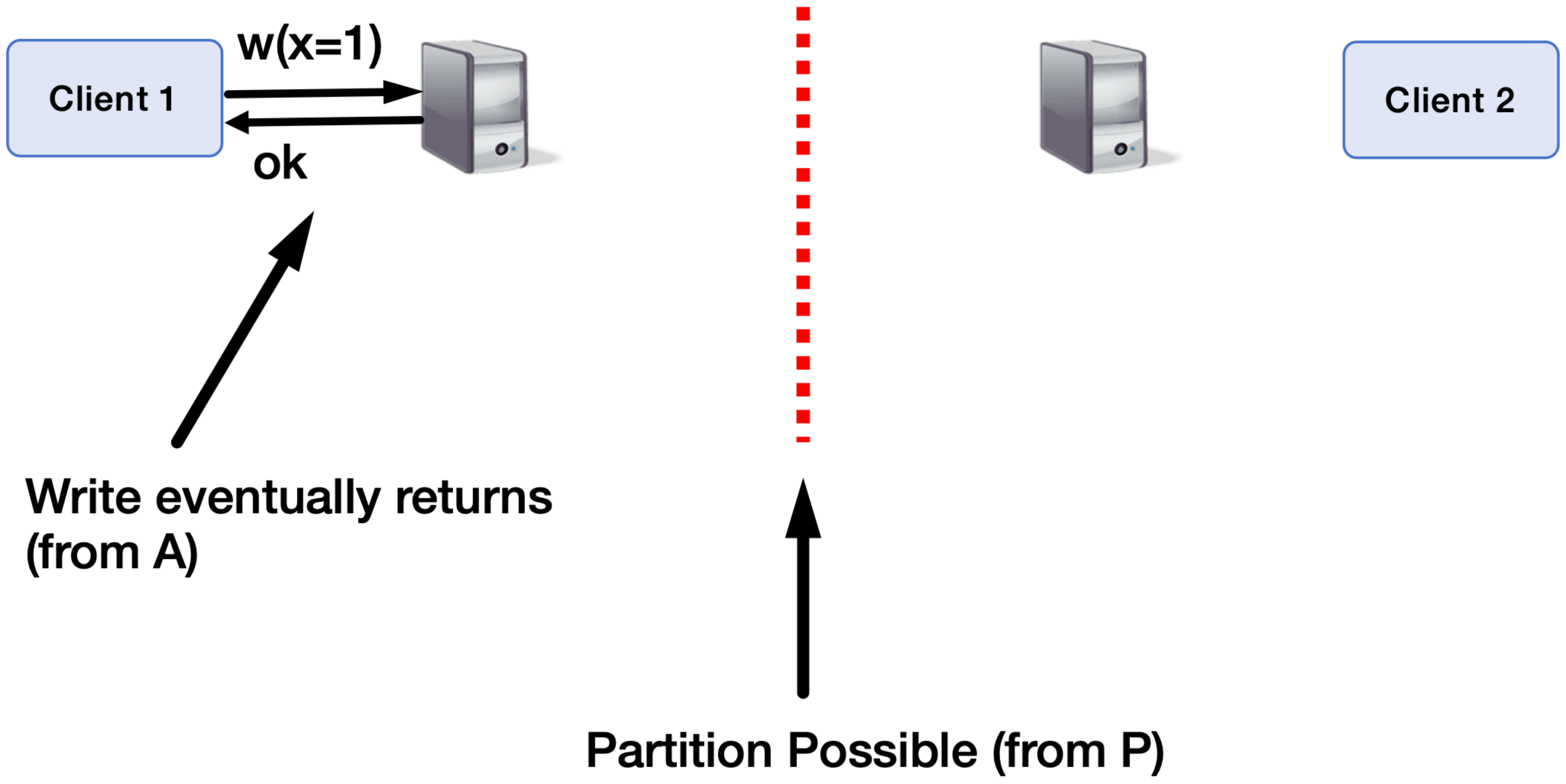
- You cannot achieve all three of:
  - Consistency
  - Availability
  - Partition-Tolerance
- Partition Tolerance => Partitions Can Happen
- Availability => All Sides of Partition Continue
- Consistency => Replicas Act Like Single Machine
  - Specifically, Linearizability

CAP THEOREM

– Assume to contradict that Algorithm A provides all of CAP

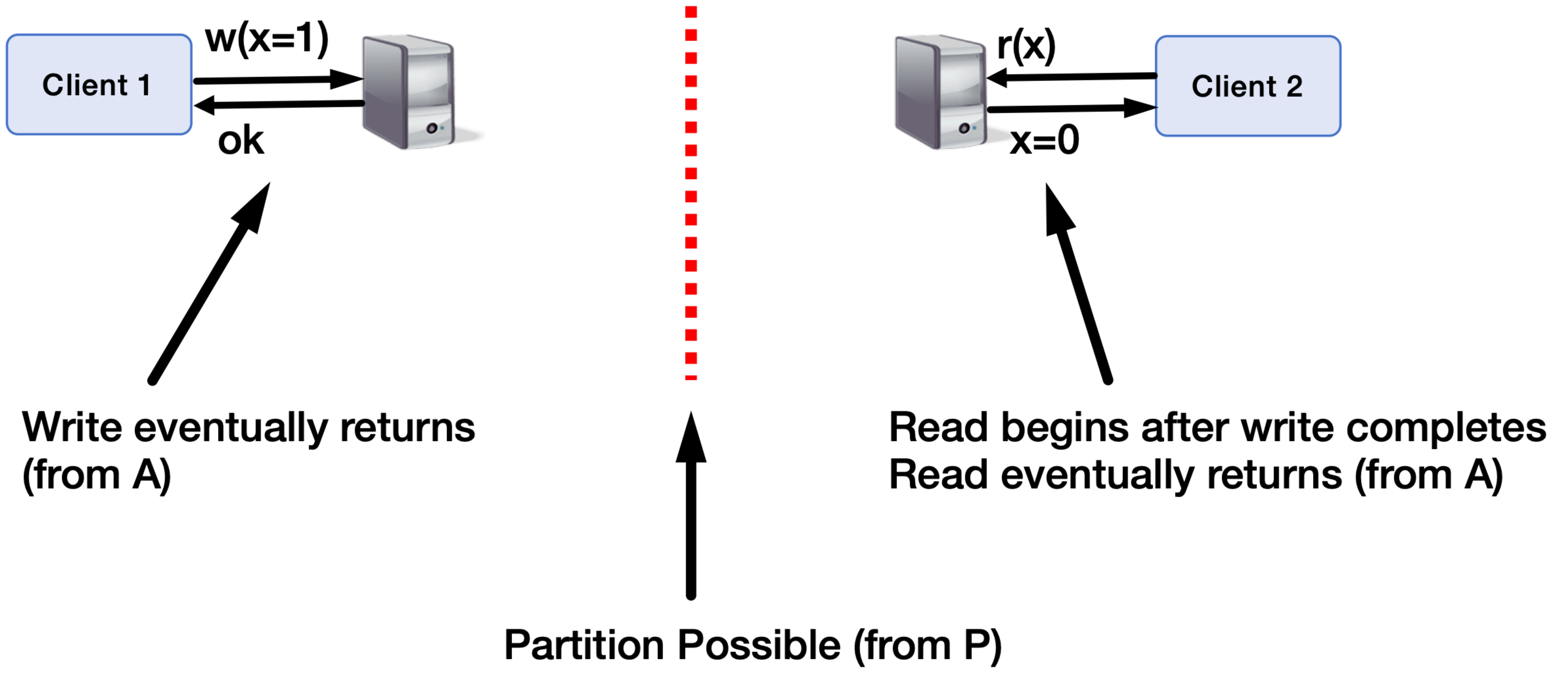
CAP THEOREM

– Assume to contradict that Algorithm A provides all of CAP



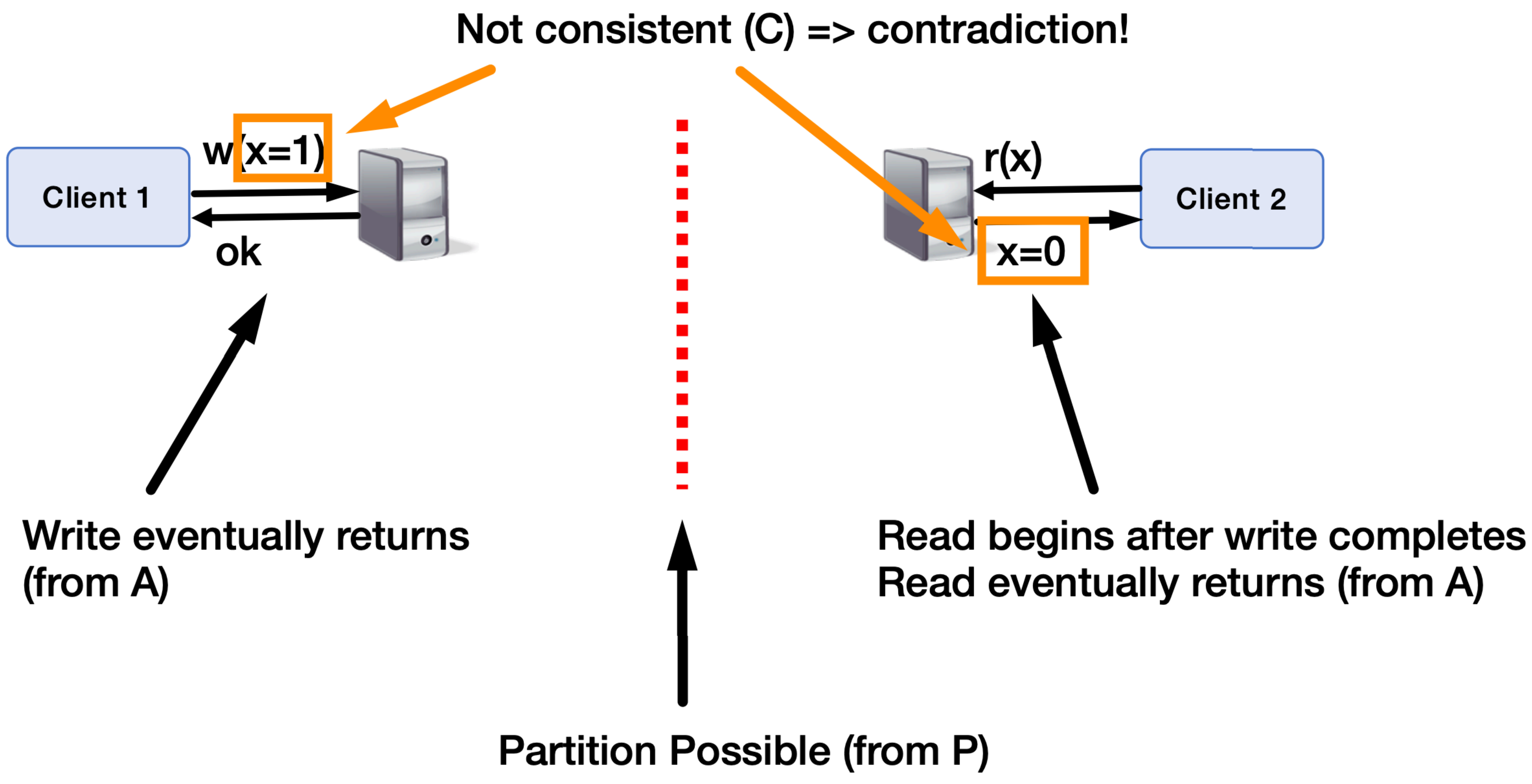
CAP THEOREM

– Assume to contradict that Algorithm A provides all of CAP



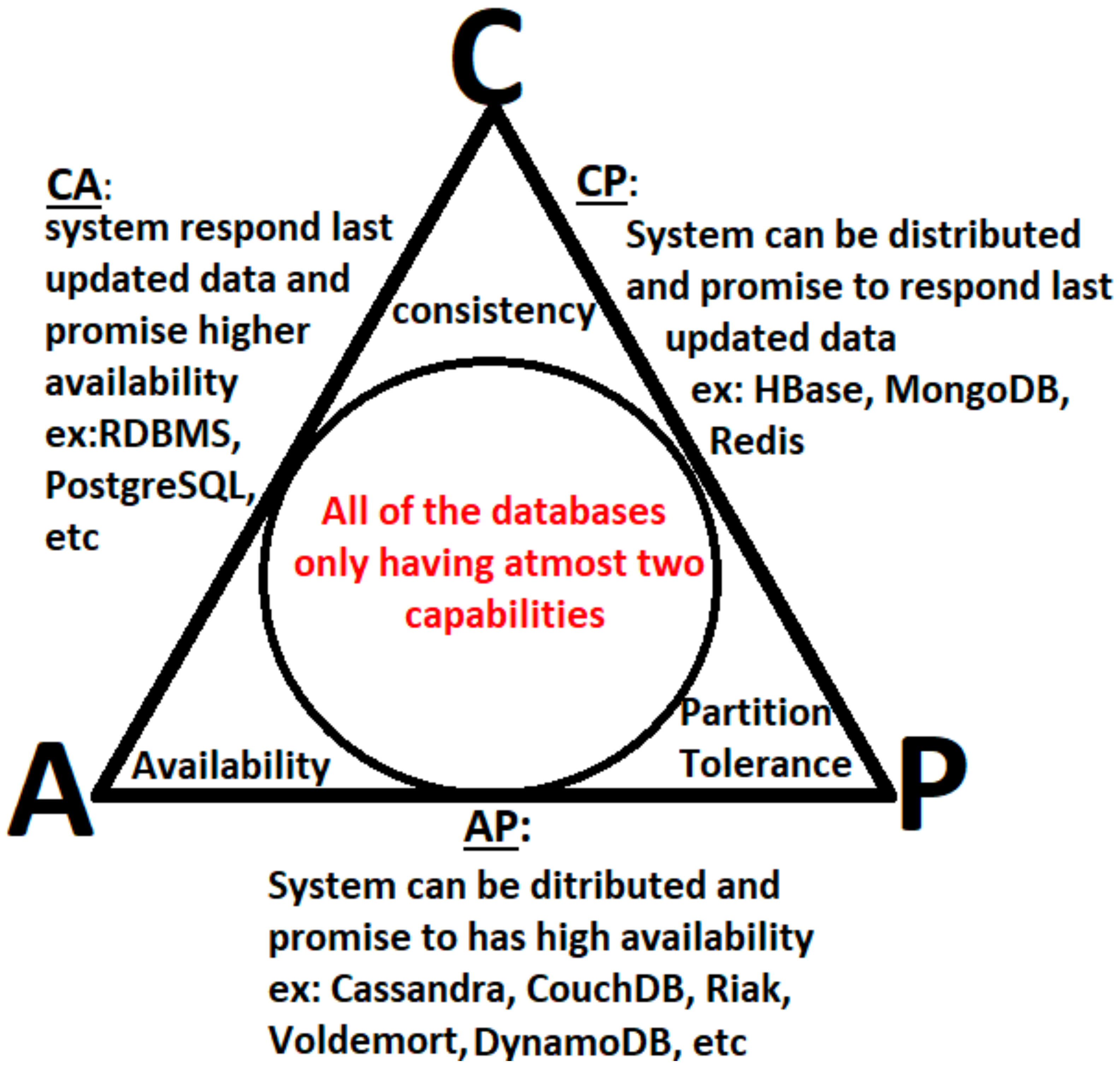
CAP THEOREM

– Assume to contradict that Algorithm A provides all of CAP



CAP CONJECTURE

– Popular interpretation: 2-out-of-3



CAP INTERPRETATION PART 1

- Cannot “choose” no partitions
  - 2-out-of-3 interpretation doesn’t make sense
  - Instead, availability OR consistency?
- That is: Fundamental tradeoff between availability and consistency
  - When designing system must choose one or the other, both are not possible

CAP INTERPRETATION PART 2

- It is a theorem, with a proof, that you understand!
- Cannot “beat” CAP Theorem
- Can engineer systems to make partitions extremely rare, however, and then just take the rare hit to availability (or consistency)

TAKEAWAYS

- Impossibility results are very useful
  - Avoids wasting effort trying to achieve impossible
  - Tells us the best-possible systems we can build!
- Today: two "impossiblilities"
  - FLP: async systems, infinite time
  - CAP: consistency, availability, partition-tolerance
- Next class: **Two-phase Commit (2PC)**

University of Virginia

ACKNOWLEDGEMENT

THIS COURSE IS DEVELOPED HEAVILY BASED ON COURSE MATERIALS SHARED BY PROF. INDRANIL GUPTA, PROF. ROBERT MORRIS, PROF. MICHAEL FREEDMAN, PROF. KYLE JAMIESON, PROF. WYATT LLOYD AND PROF. ROXANA GEAMBASU. MANY APPRECIATIONS FOR GENEROUSLY SHARING THEIR MATERIALS AND TEACHING INSIGHTS.