CAP Theorem and Consistency Models

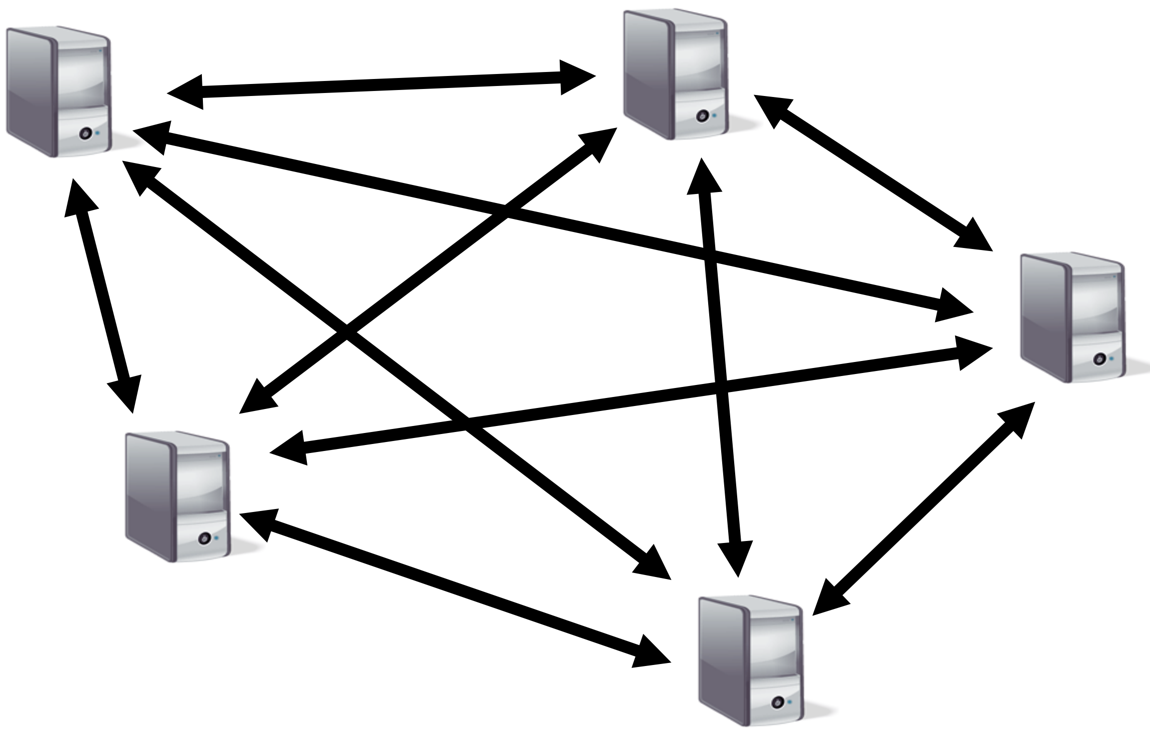
COS 418: Distributed Systems
Lecture 12

Wyatt Lloyd

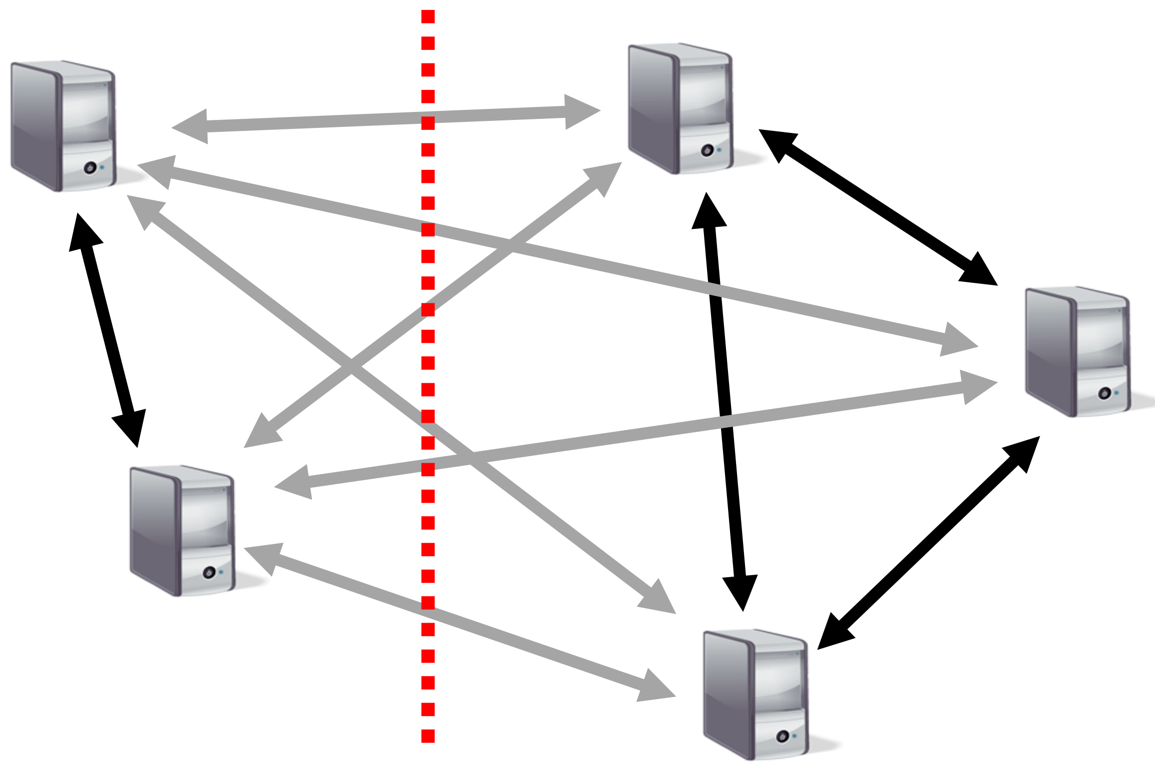
Outline

1. Network Partitions
2. Linearizability
3. CAP Theorem
4. Consistency Hierarchy

Network Partitions Divide Systems



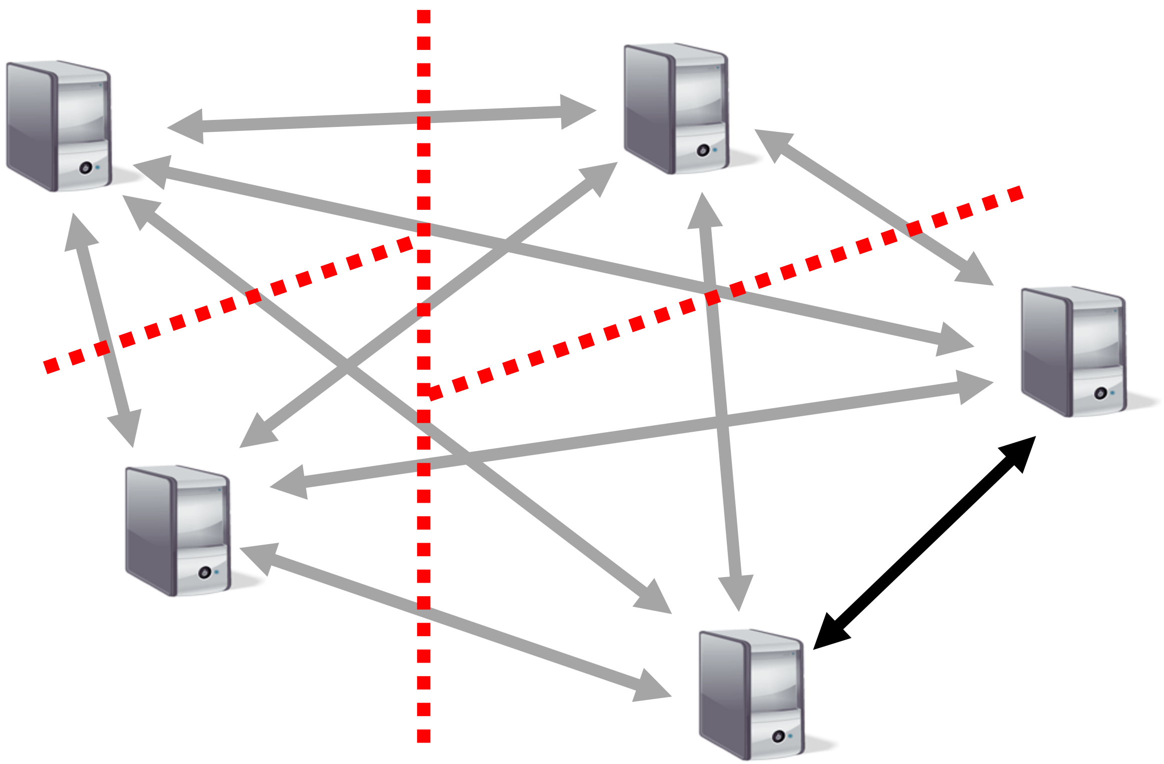
Network Partitions Divide Systems



How Can We Handle Partitions?

- Atomic Multicast?
- Bayou?
- Viewstamped Replication?
- Chord?
- Paxos?
- Dynamo?
- RAFT?

How About This Set of Partitions?



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:
  1. Consistency
  2. Availability
  3. Partition-Tolerance

- Partition Tolerance => Partitions Can Happen
- Availability => All Sides of Partition Continue
- Consistency => Replicas Act Like Single Machine
  - Specifically, Linearizability

Outline

1. Network Partitions
2. Linearizability
3. CAP Theorem
4. Consistency Hierarchy

Linearizability [Herlihy and Wing 1990]

- All replicas execute operations in some total order
- That total order preserves the real-time ordering between operations
  - If operation A completes before operation B begins, then A is ordered before B in real-time
  - If neither A nor B completes before the other begins, then there is no real-time order
    - (But there must be *some* total order)

Real-Time Ordering Examples

Linearizability == “Appears to be a Single Machine”

- Single machine processes requests one by one in the order it receives them
  - Will receive requests ordered by real-time in that order
  - Will receive all requests in some order

- Atomic Multicast, Viewstamped Replication, Paxos, and RAFT provide Linearizability

Linearizability is Ideal?

- Hides the complexity of the underlying distributed system from applications!
  - Easier to write applications
  - Easier to write correct applications

- But, performance trade-offs, e.g., CAP

Outline

1. Network Partitions

2. Linearizability

3. CAP Theorem

4. Consistency Hierarchy

CAP Conjecture [Brewer 00]

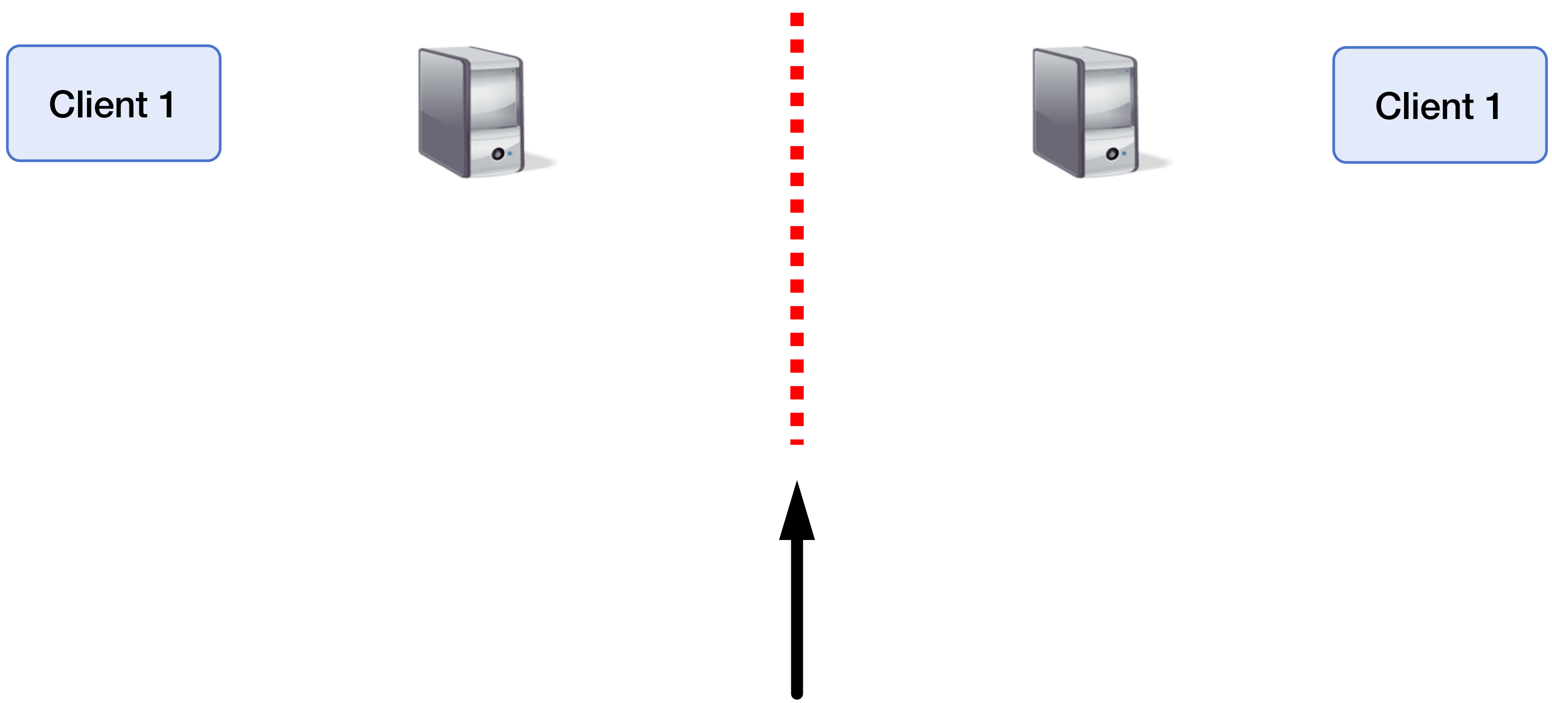
- From keynote lecture by Eric Brewer (2000)
  - History: Eric started Inktomi, early Internet search site based around “commodity” clusters of computers
  - Using CAP to justify “BASE” model: Basically Available, Soft-state services with Eventual consistency
- Popular interpretation: 2-out-of-3
  - Consistency (Linearizability)
  - Availability
  - Partition Tolerance: Arbitrary crash/network failures

CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm *A* provides all of CAP

CAP Theorem [Gilbert Lynch 02]

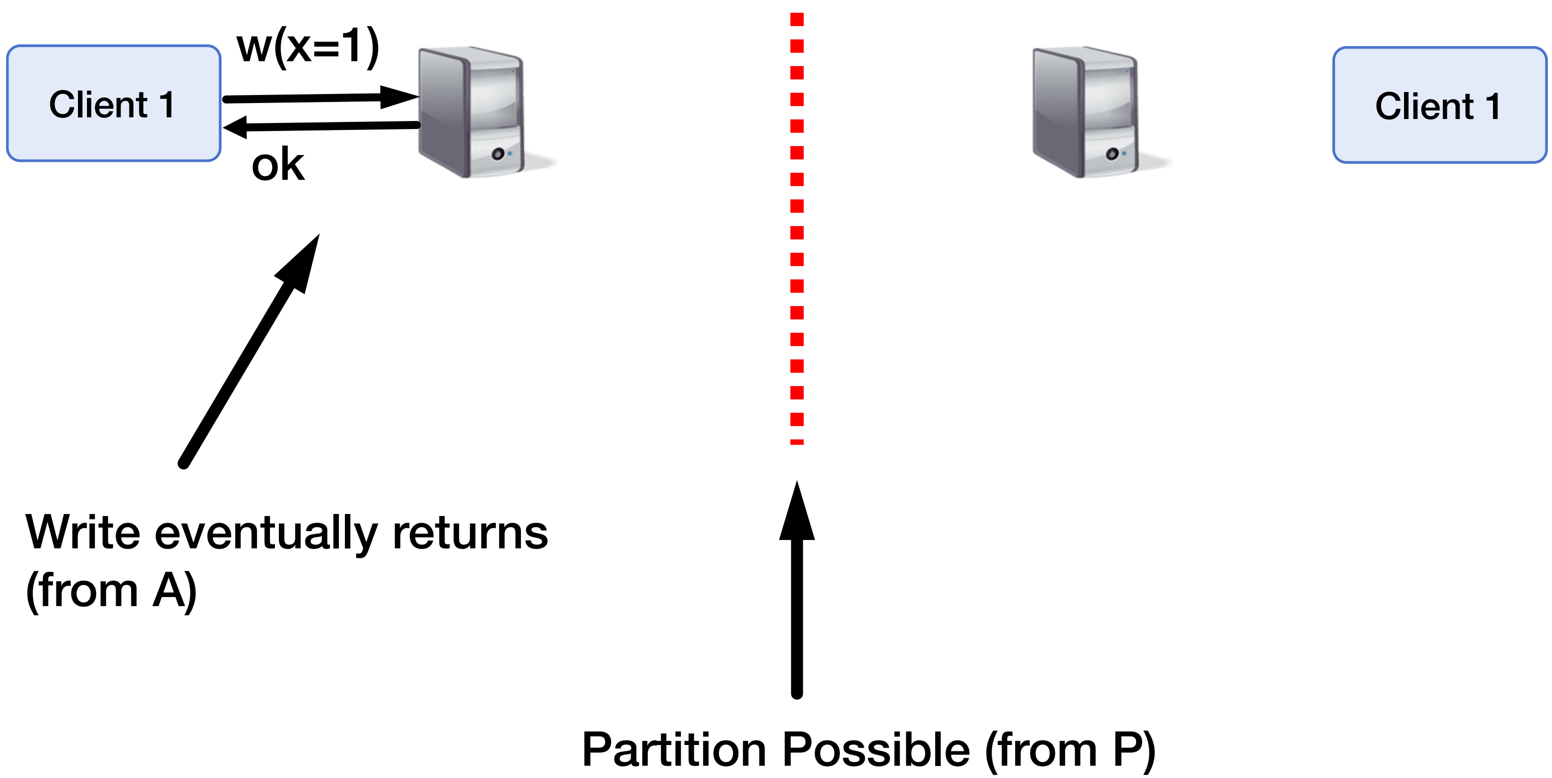
Assume to contradict that Algorithm *A* provides all of CAP



Partition Possible (from P)

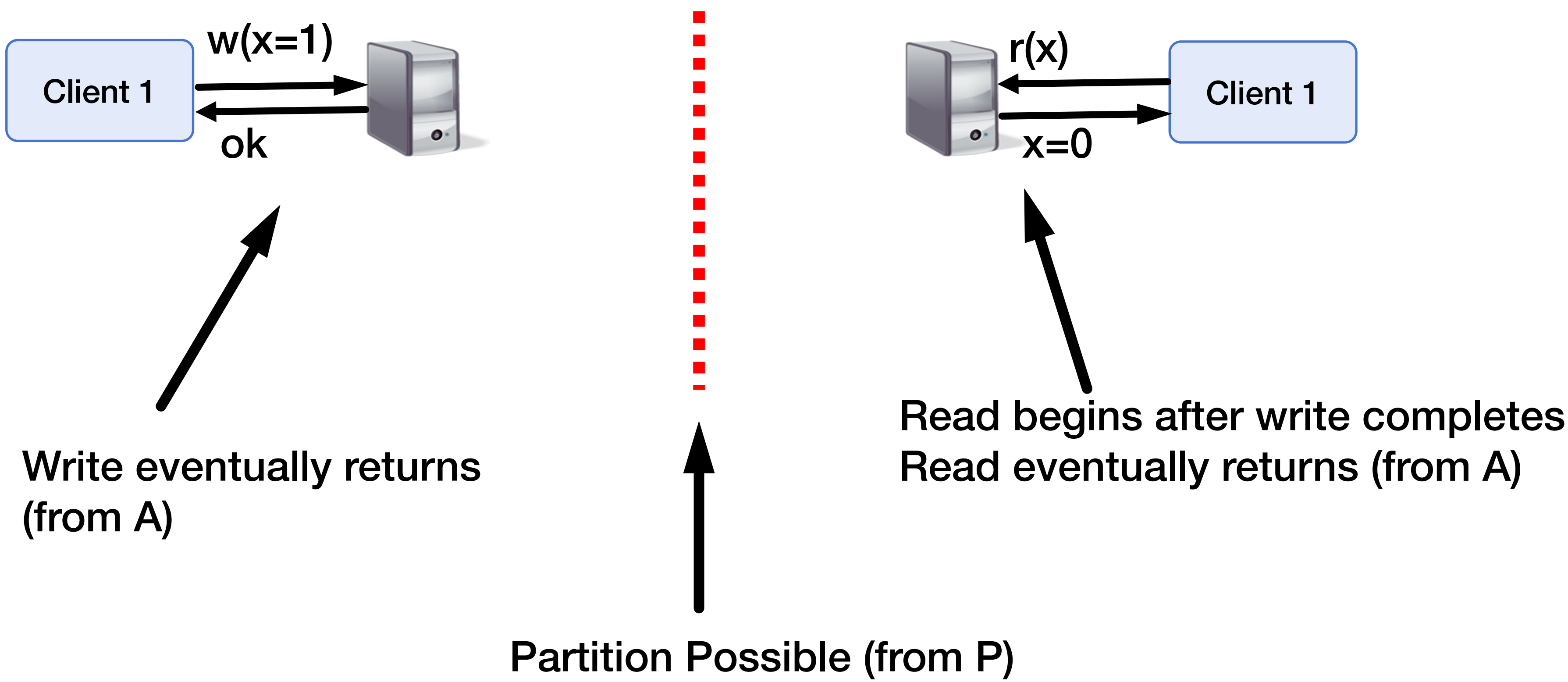
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm *A* provides all of CAP



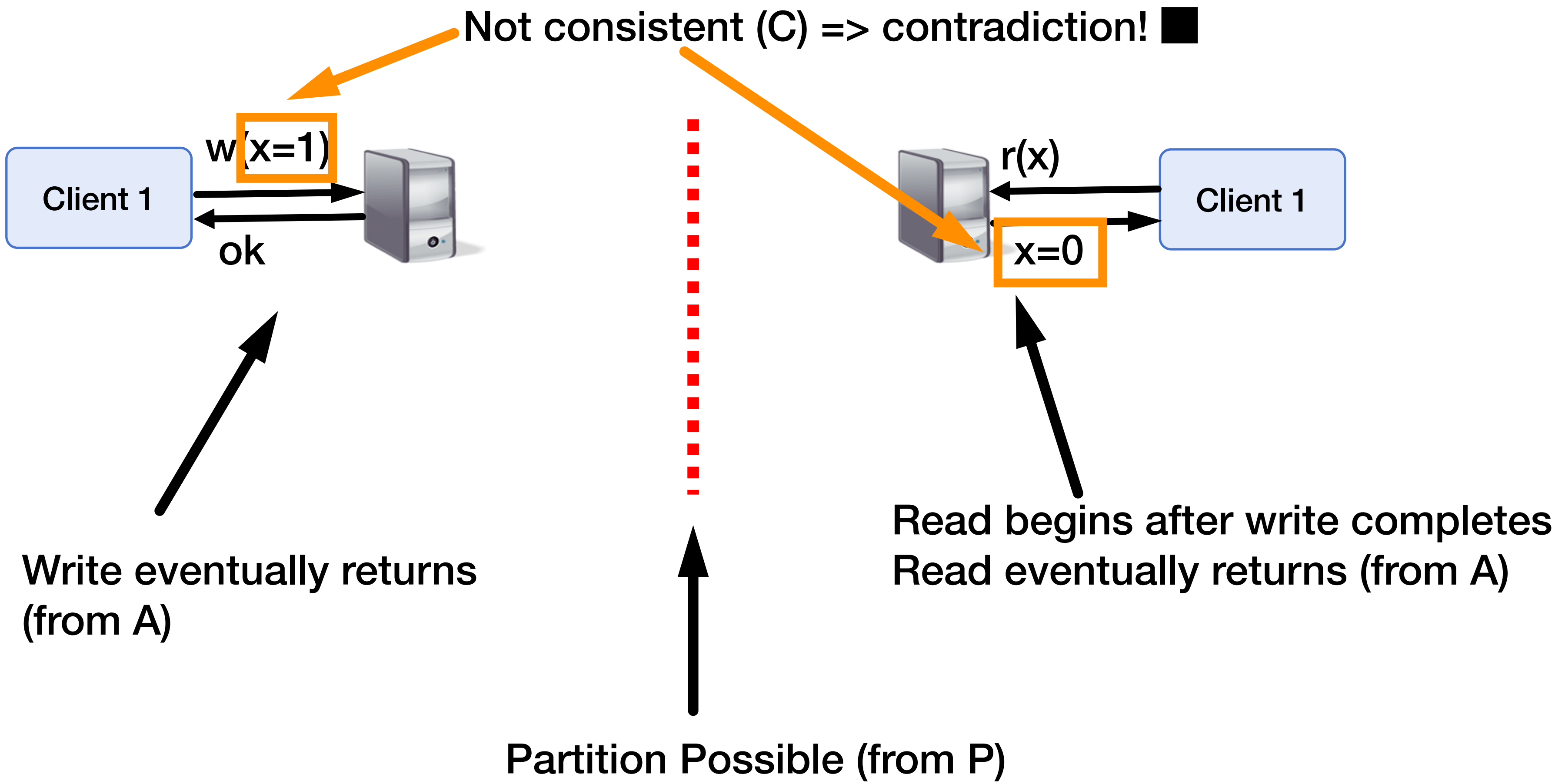
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm *A* provides all of CAP



CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm *A* provides all of CAP



CAP Interpretation Part 1

- Cannot “choose” no partitions
  - 2-out-of-3 interpretation doesn’t make sense
  - Instead, availability OR consistency?

- i.e., 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)

Outline

1. Network Partitions
2. Linearizability
3. CAP Theorem
4. Consistency Hierarchy

Consistency Models

- Contract between a distributed system and the applications that run on it
- A consistency model is a set of guarantees made by the distributed system
- e.g., Linearizability
  - Guarantees a total order of operations
  - Guarantees the real-time ordering is respected

Stronger vs Weaker Consistency

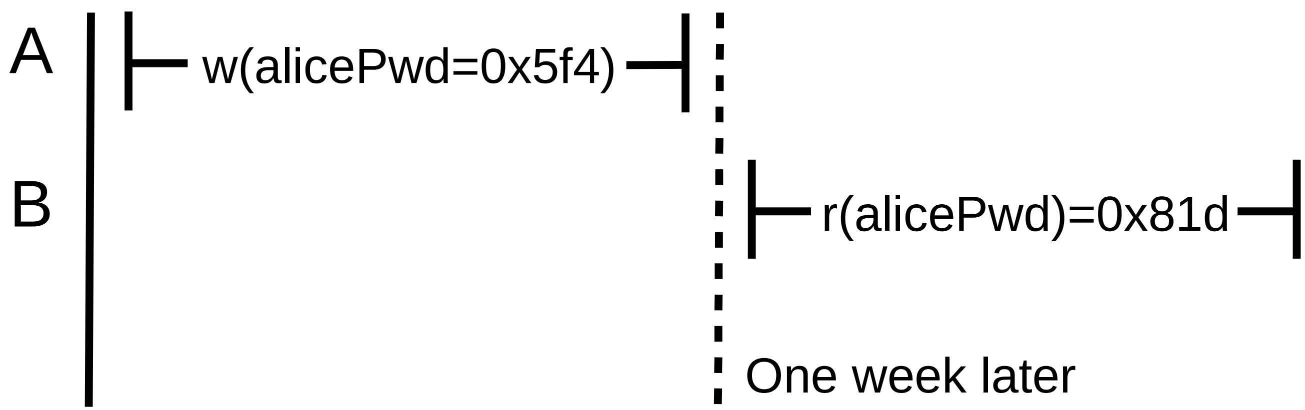
- Stronger consistency models
  - \+ Easier to write applications
  - \- More guarantees for the system to ensure
    - Results in performance tradeoffs
- Weaker consistency models
  - \- Harder to write applications
  - \+ Fewer guarantees for the system to ensure

Strictly Stronger Consistency

- A consistency model *A* is strictly stronger than *B* if it allows a strict subset of the behaviors of B
  - Guarantees are strictly stronger

- Linearizability is strictly stronger than Sequential Consistency
  - Linearizability: $\exists$ total order + real-time ordering
  - Sequential: $\exists$ total order + process ordering
    - Process ordering $\subseteq$ Real-time ordering

Sequential But Not Linearizable



Consistency Hierarchy

Linearizability — e.g., RAFT

↓

Sequential Consistency

↓

Causal+ Consistency — e.g., Bayou

↓

Eventual Consistency — e.g., Dynamo

Causal+ Consistency

- Partially orders all operations, does not totally order them
    - Does not look like a single machine

- Guarantees
    - For each process, $\exists$ an order of all writes + that process's reads
    - Order respects the happens-before ($\rightarrow$) ordering of operations
    - + replicas converge to the same state
        - Skip details, makes it stronger than eventual consistency

Causal+ But Not Sequential

P_A ├ w(x=1) ┤ ├ r(y)=0 ┤

P_B ├ w(y=1) ┤ ├ r(x)=0 ┤

√ Casual+

Happens Before Order

w(x=1) → r(y)=0

w(y=1) → r(x)=0

P_A Order: w(x=1), r(y=0), w(y=1)

P_B Order: w(y=1), r(x=0), w(x=1)

X Sequential

Process Ordering

w(x=1) → r(y)=0

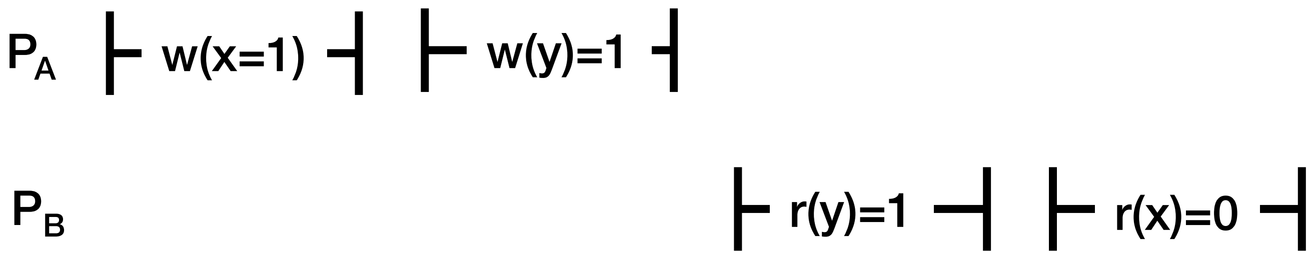
w(y=1) → r(x)=0

No Total Order

w(x=1) → r(y)=0

w(y=1) → r(x)=0

Eventual But Not Causal+

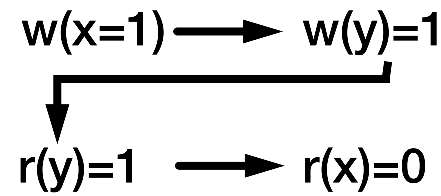


√ Eventual

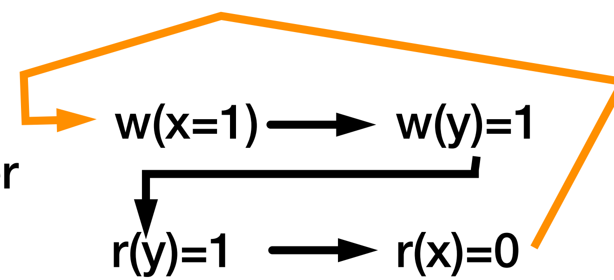
As long as P_B eventually would see r(x)=1 this is fine

X Causal+

Happens Before Ordering



No Order for P_B



Consistency Hierarchy

Linearizability — e.g., RAFT

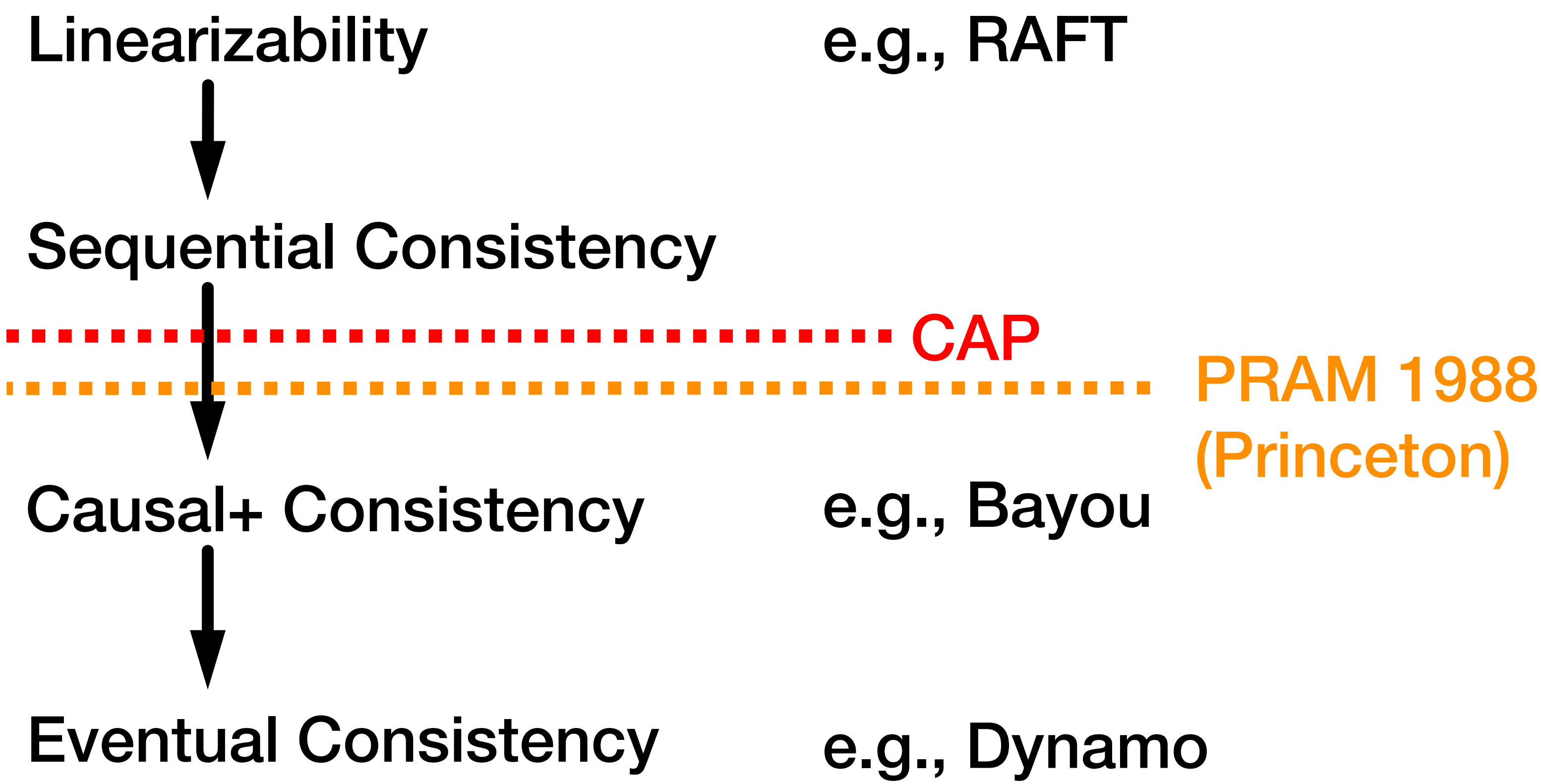
CAP

Sequential Consistency

Causal+ Consistency — e.g., Bayou

Eventual Consistency — e.g., Dynamo

Consistency Hierarchy



PRAM [Lipton Sandberg 88] [Attiya Welch 94]

- *d* is the worst-case delay in the network over all pairs of processes
- Sequentially consistent system
- read time + write time ≥ *d*
- Fundamental tradeoff between consistency and latency!

Outline

1. Network Partitions
2. Linearizability
3. CAP Theorem
4. Consistency Hierarchy