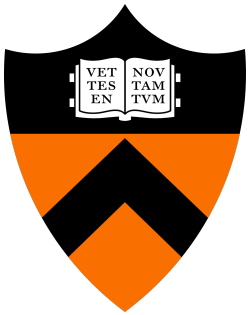
# Consistency Models




COS 418/518: Distributed Systems

Lecture 14

Wyatt Lloyd, Mike Freedman

# Consistency Models

- Contract between a distributed system and the applications that run on it

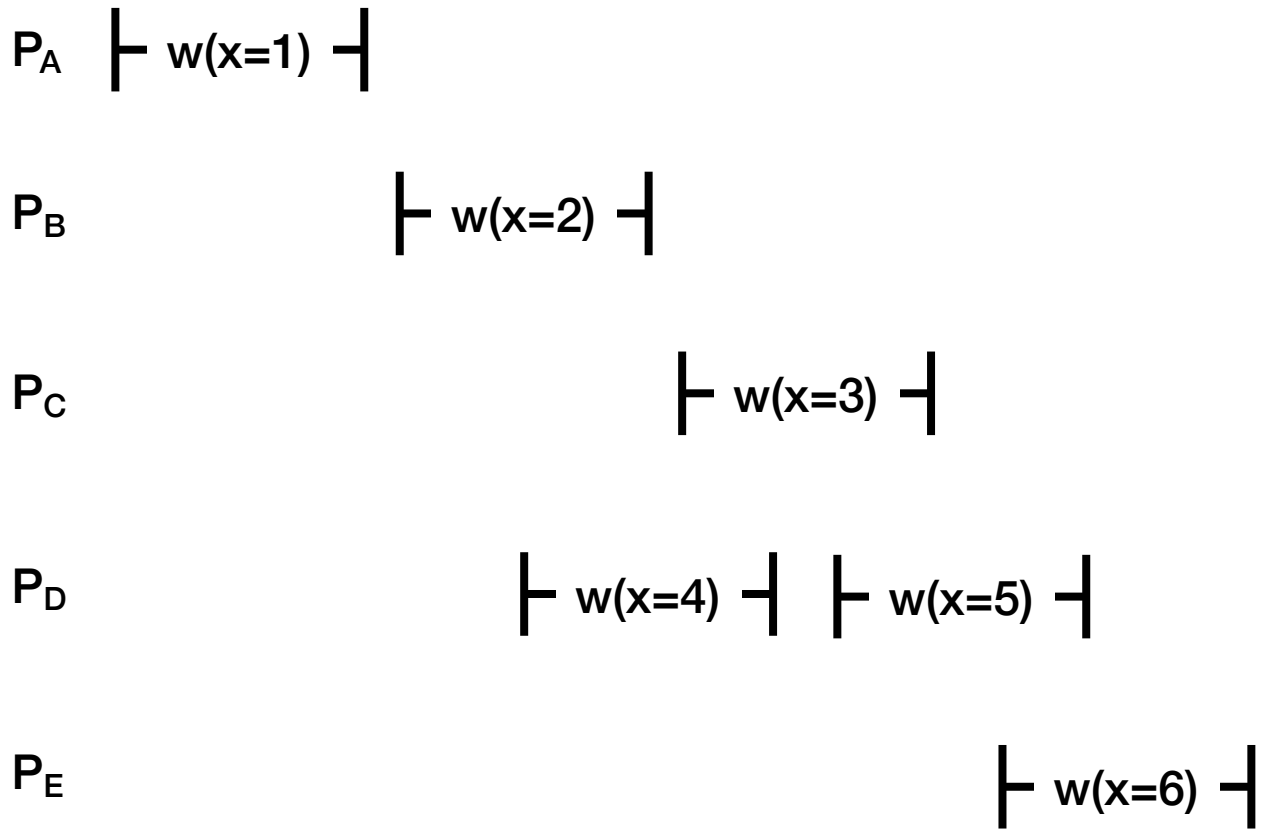
- A consistency model is a set of guarantees made by the distributed system

# Linearizability

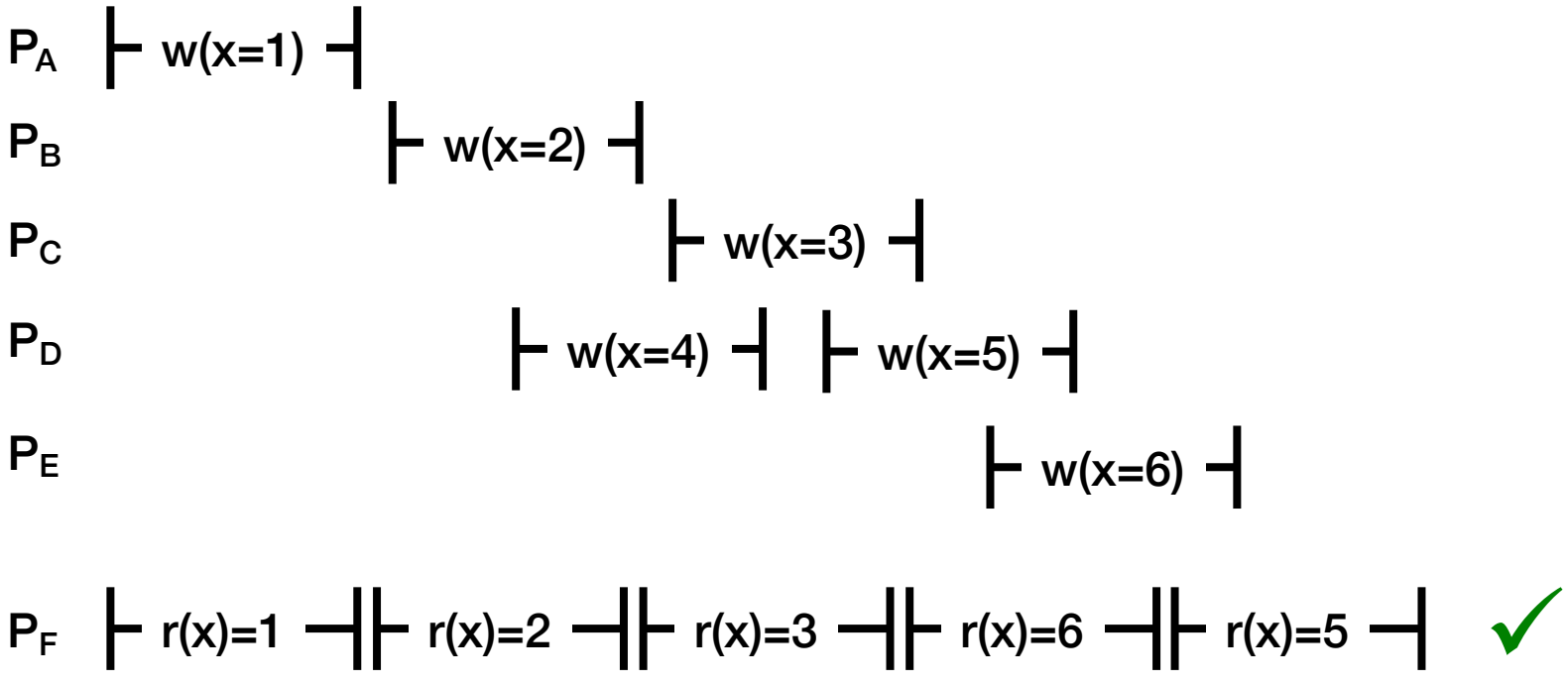
- All replicas execute operations in some total order

- 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)

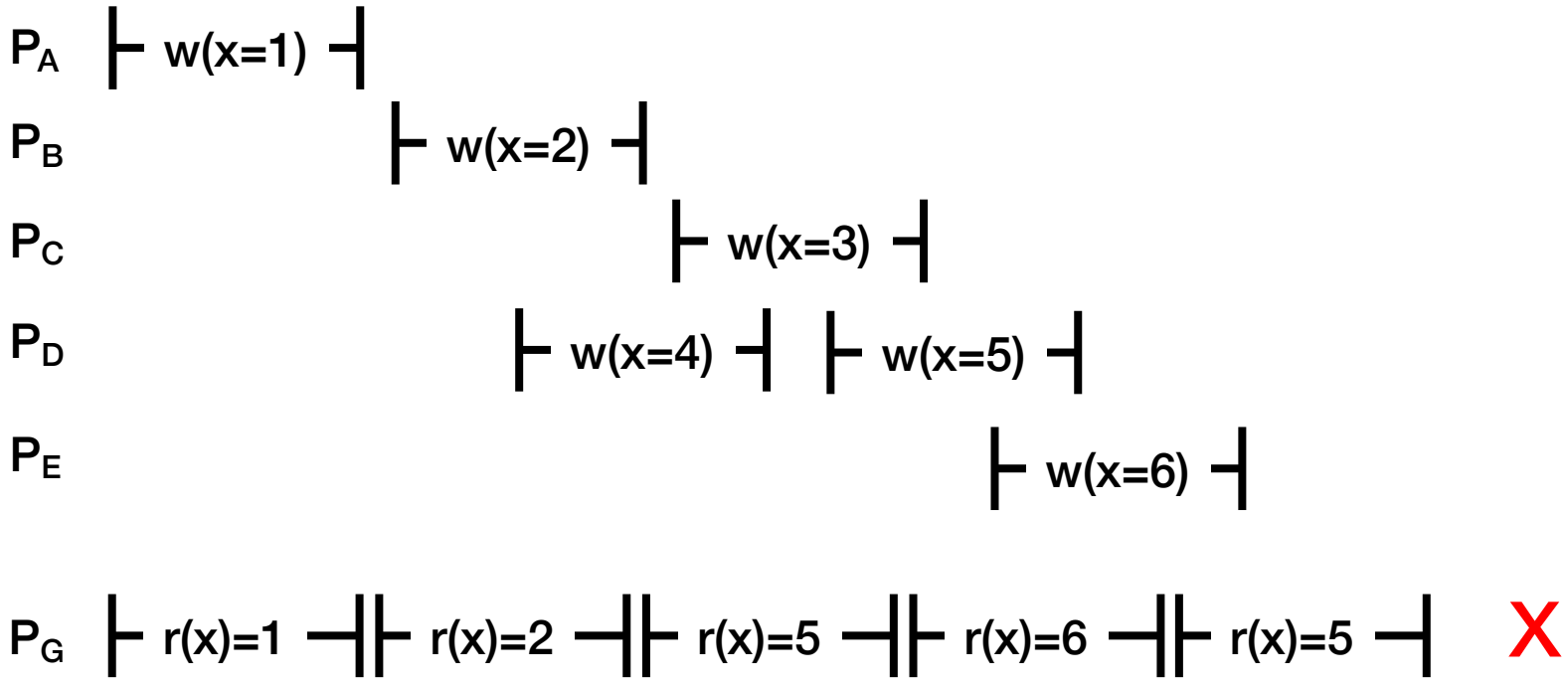
# Real-Time Ordering Examples



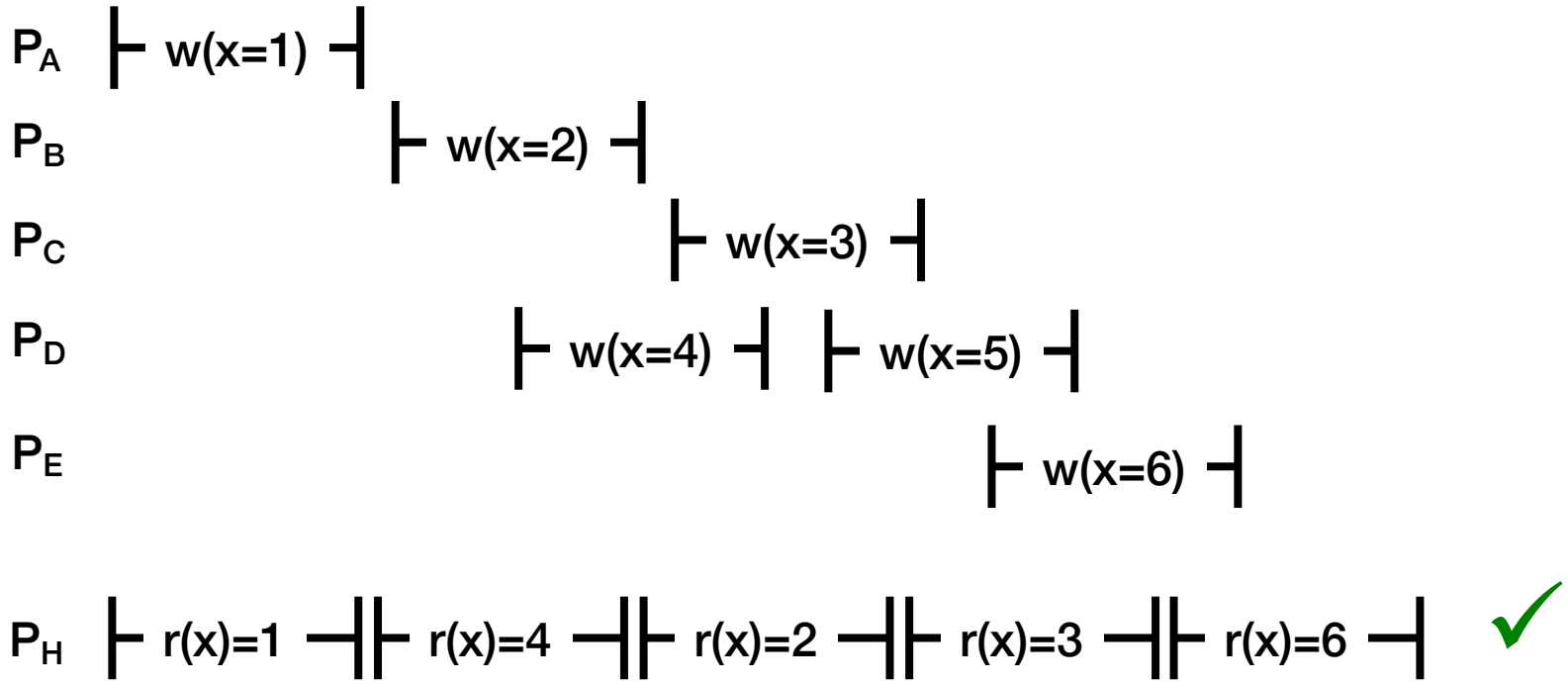
# Linearizable?



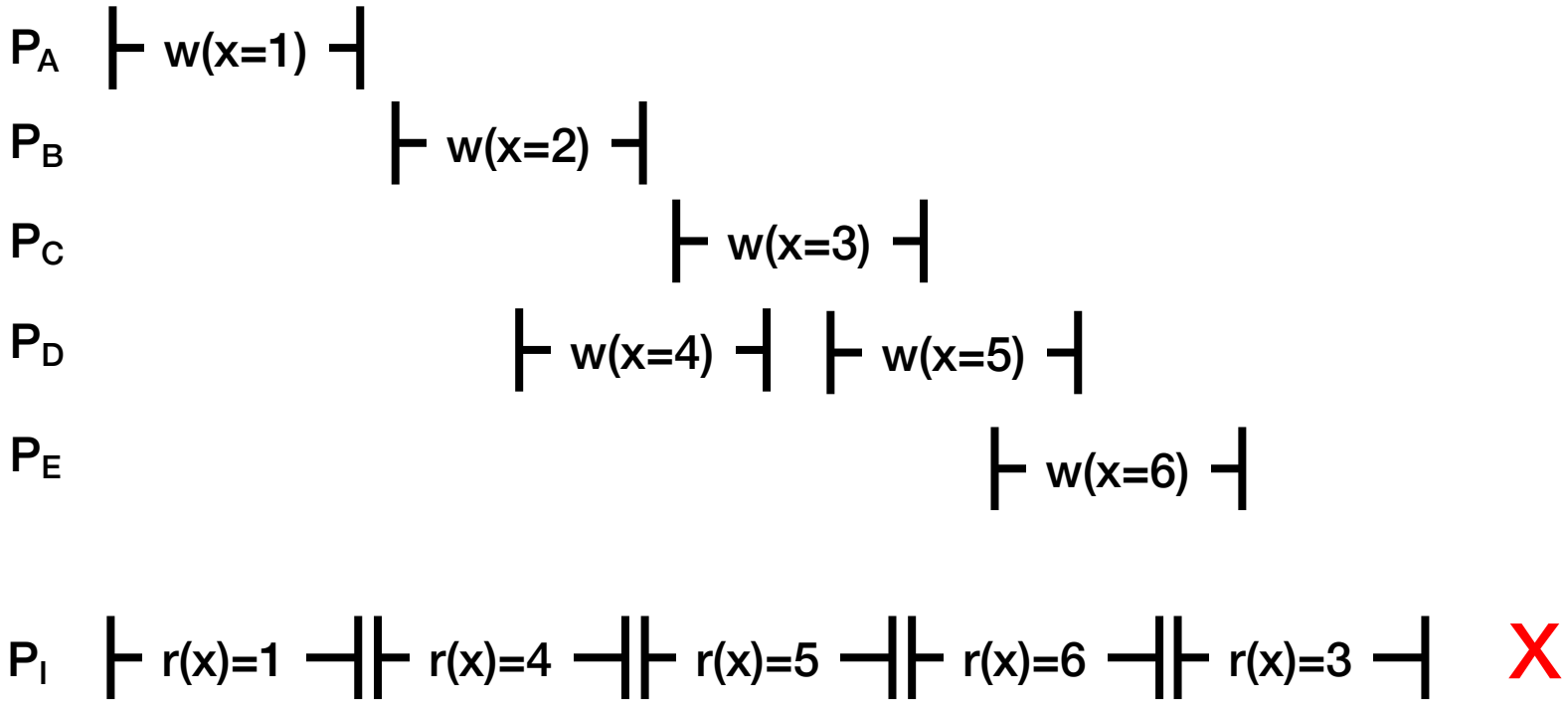
# Linearizable?



# Linearizable?



# Linearizable?



# 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

- Single machine processing incoming requests one at a time also 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

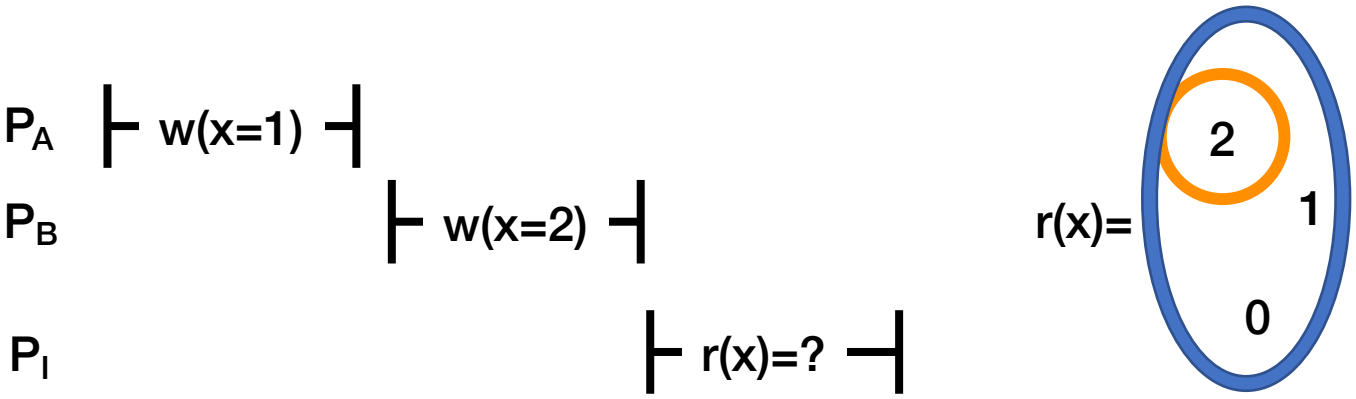
# Stronger vs Weaker Consistency

- 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



# Sequential Consistency

- All replicas execute operations in some total order

- That total order preserves the process ordering between operations
  - If process P issues operation A before operation B, then A is order before B by the process order
  - If operations A and B and done by different processes then there is no process order between them
    - (But there must be *some* total order)

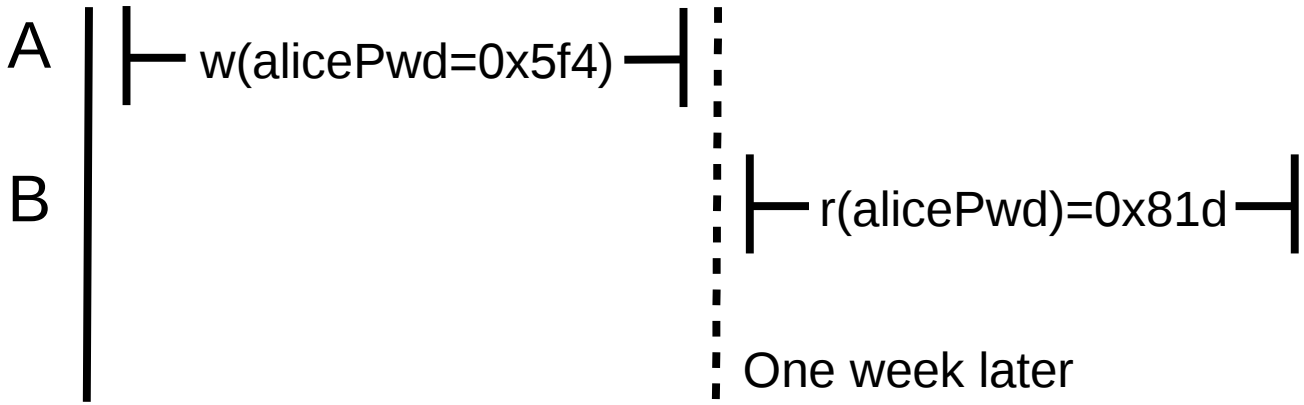
# Sequential Consistency ≈ “Appears to be a Single Machine”

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

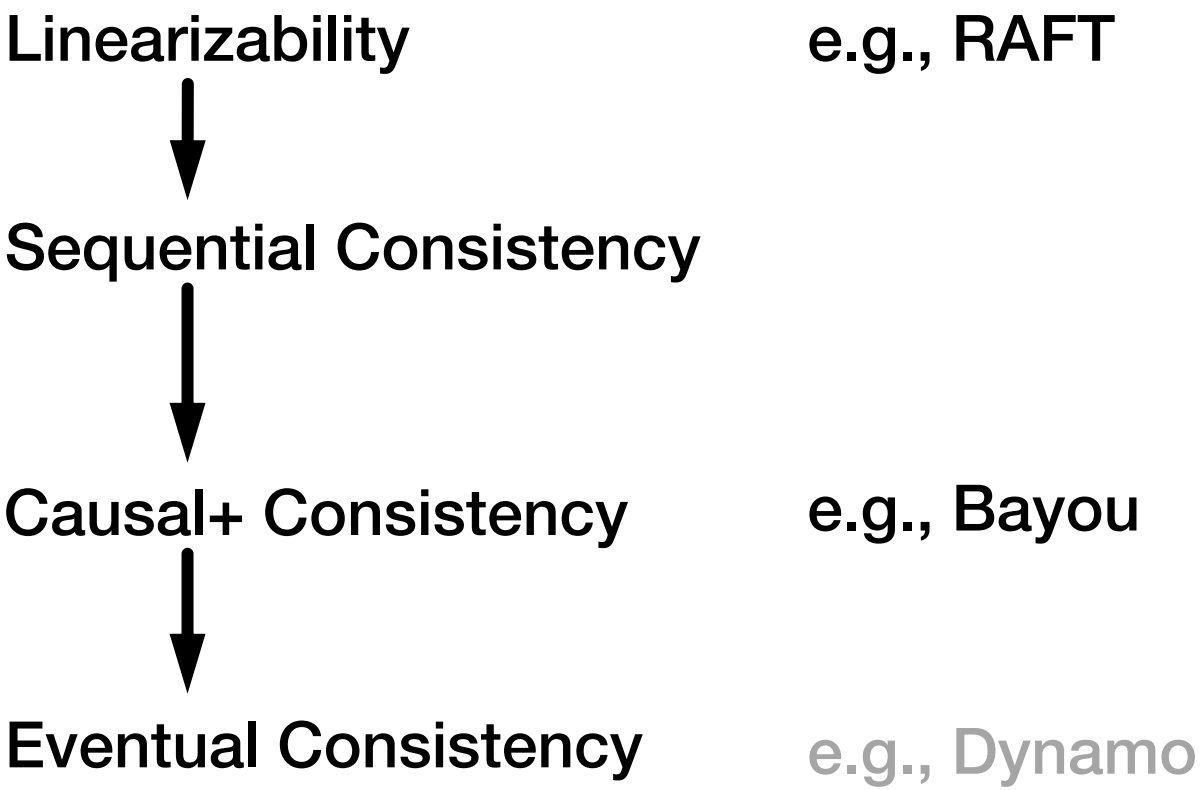
# Linearizability is strictly stronger than Sequential Consistency

- Linearizability: ∃total order + real-time ordering
- Sequential: ∃total order + process ordering
  - Process ordering ⊆ Real-time ordering

# Sequential But Not Linearizable



# Consistency Hierarchy



# Causal+ Consistency

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

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

# Causal Consistency

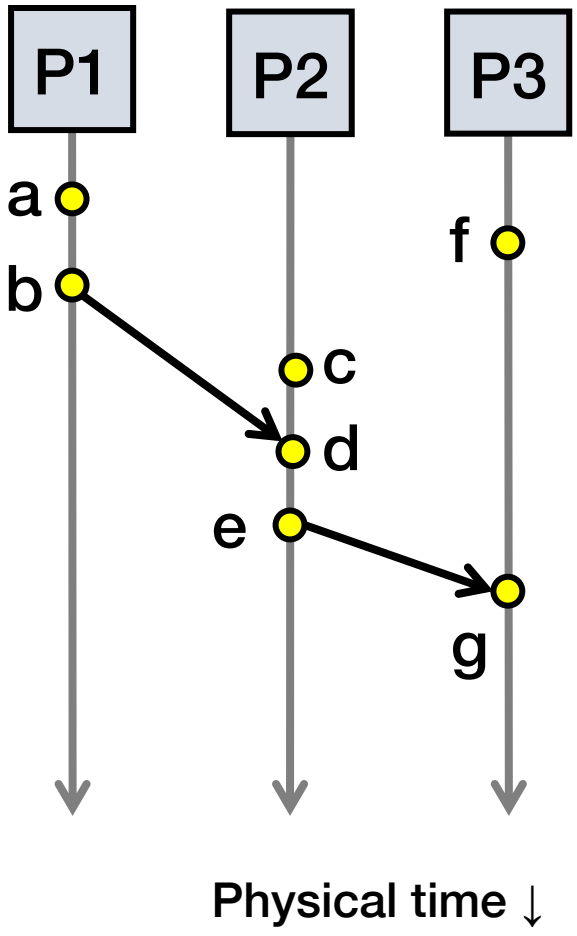
1. Writes that are potentially causally related must be seen by all processes in same order.
2. Concurrent writes may be seen in a different order on different processes.

- Concurrent: Ops not causally related

# Causal Consistency

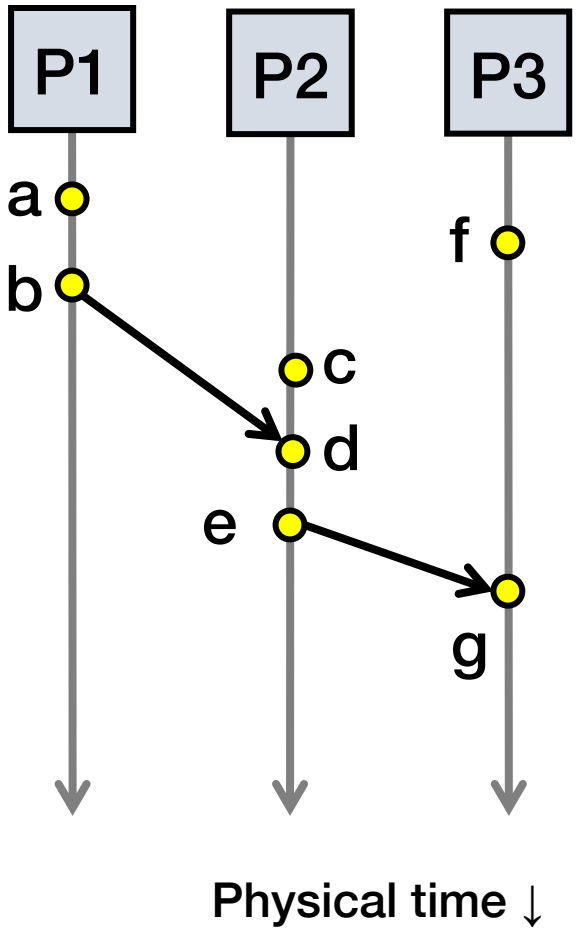
1. Writes that are potentially causally related must be seen by all processes in same order.
2. Concurrent writes may be seen in a different order on different processes.

- Concurrent: Ops not causally related



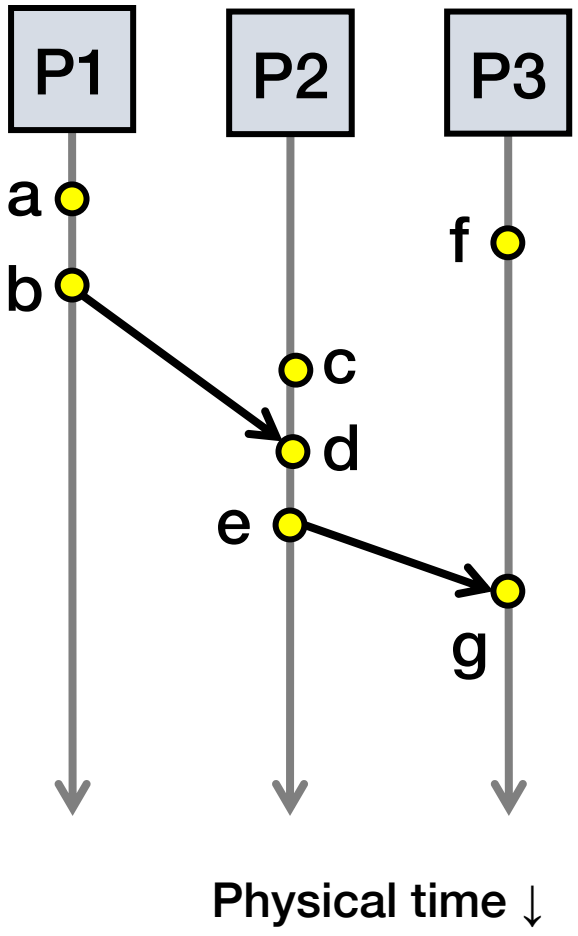
# Causal Consistency

| Operations | Concurrent? |
|---|---|
| a, b | |
| b, f | |
| c, f | |
| e, f | |
| e, g | |
| a, c | |
| a, e | |

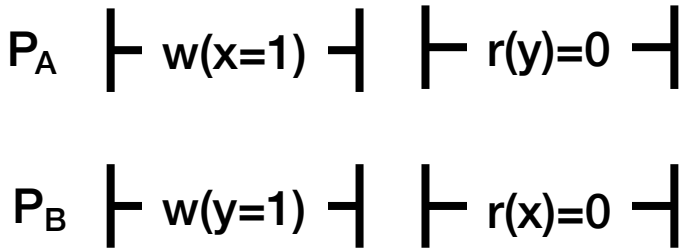


# Causal Consistency

| Operations | Concurrent? |
|---|---|
| a, b | N |
| b, f | Y |
| c, f | Y |
| e, f | Y |
| e, g | N |
| a, c | Y |
| a, e | N |



# Causal+ But Not Sequential




## ✓ 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)

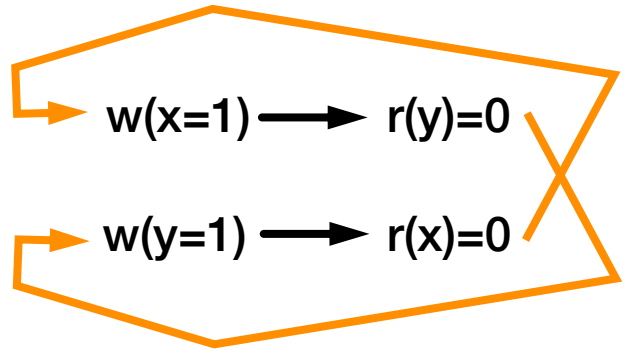
## X Sequential

Process Ordering

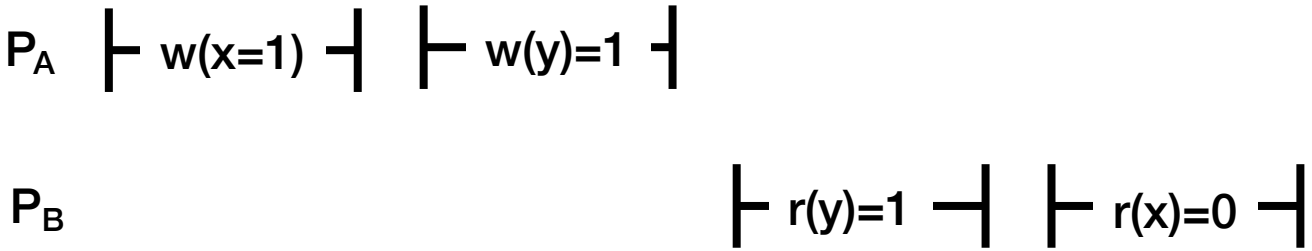
w(x=1) → r(y)=0

w(y=1) → r(x)=0

No Total Order



# Eventual But Not Causal+

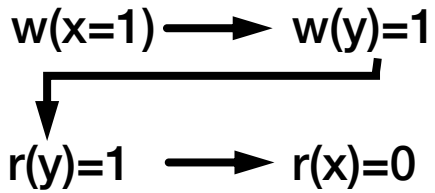



√ Eventual

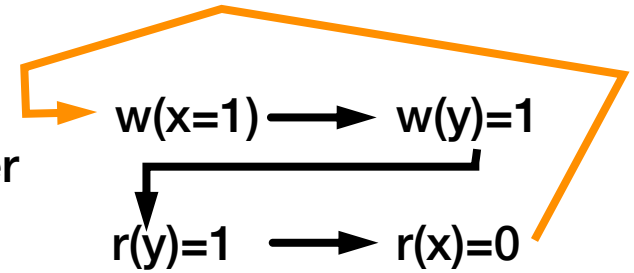
As long as $P_B$ eventually would see r(x)=1 this is fine

X Causal+

Happens Before Ordering




No Order for $P_B$



# Consistency Hierarchy