8. Replication

8. Replication

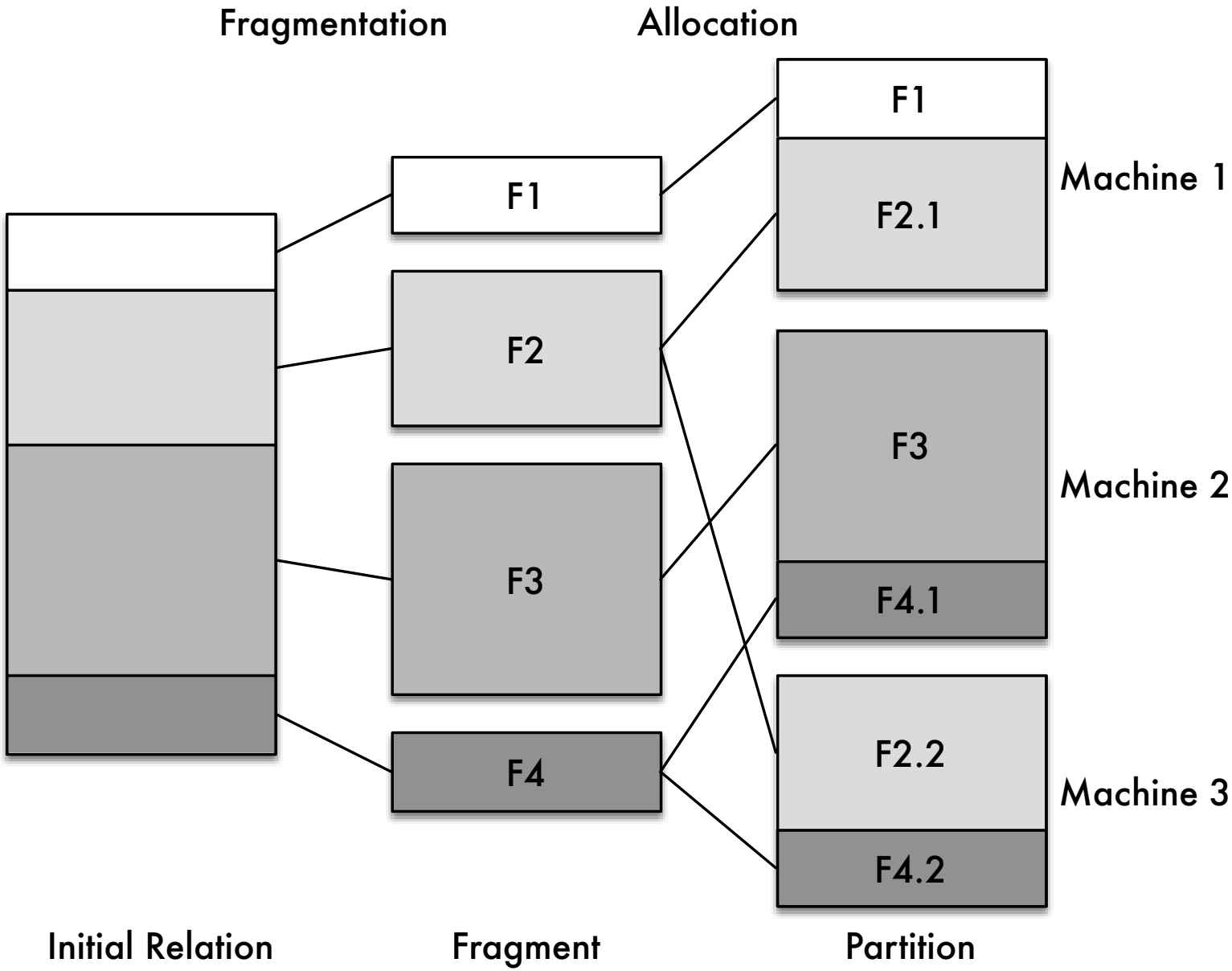
8. Replication

Reasons for replication

Replication is done for following reasons:

- Query acceleration
  - Parallel processing
  - Load balancing
  - Regional proximity of copies
- Increased availability
  - Replica steps in at node failures
  - Access even in partitioned networks
- Data backup
  - Replica instead of backup on magnetic tape

Replication and data distribution



BASE instead of ACID

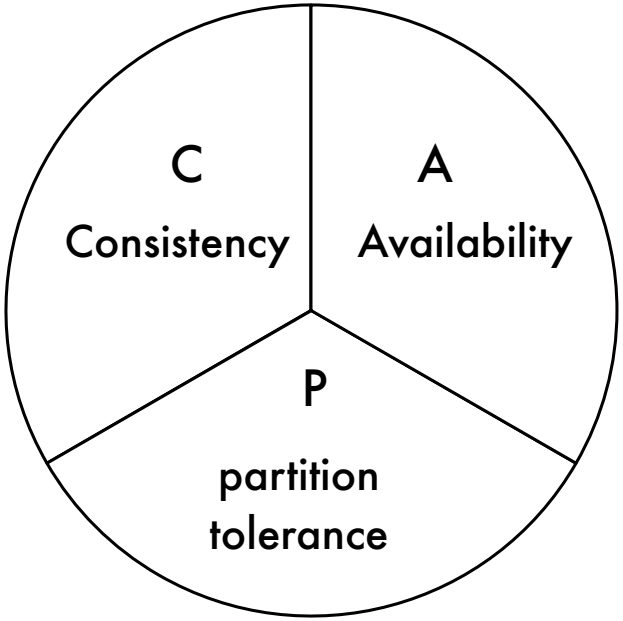
BASE stands for

- Basically available
  - Accessibility has highest priority in cloud applications
- Soft state
  - Short-term inconsistencies in replicated databases are tolerated
- Eventual consistency
  - *Eventual consistency* allows that individual clients can see outdated data but a consistent state will be reached eventually

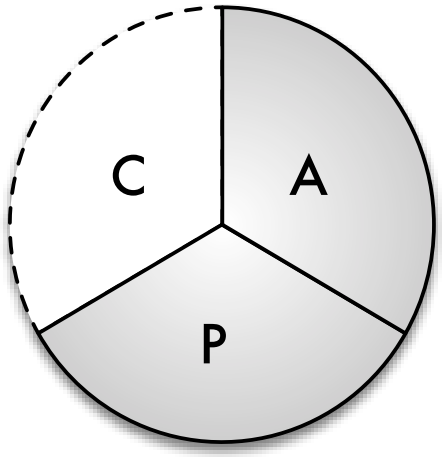
BASE instead of ACID: II

ACID	BASE
Strong consistency	Weak consistency
Focus: Isolation	Focus: Availability
Pessimistic synchronization	Optimistic synchronization
Global commits	Decoupled local commits

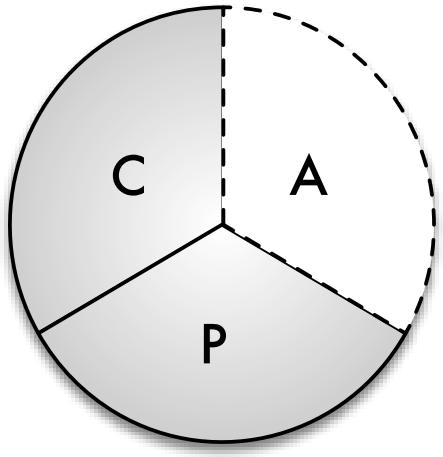
Replication and CAP



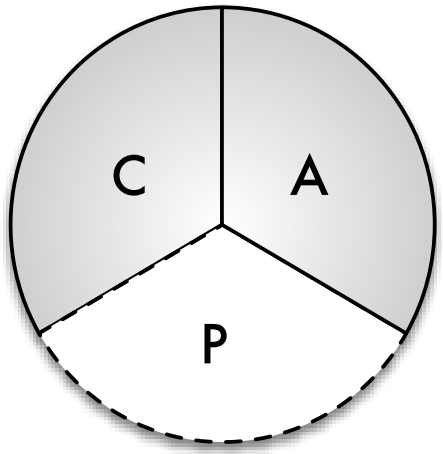
Response to CAP



a) limited consistency



b) limited availability



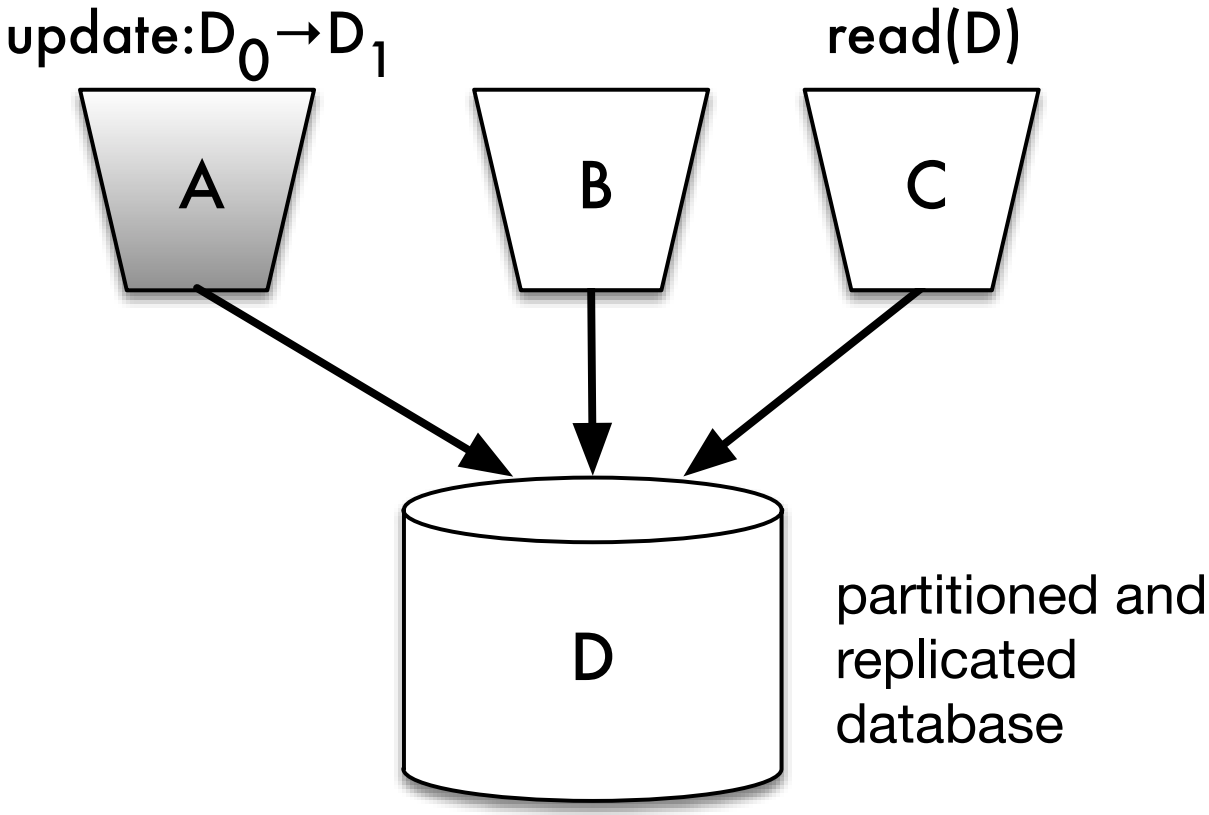
c) limited tolerance for network partitionings

AP Scalable cloud applications: *Eventual Consistency*

CP Distributed DB with pessimistic replication

AC Single-Server-DB, Cluster-DB

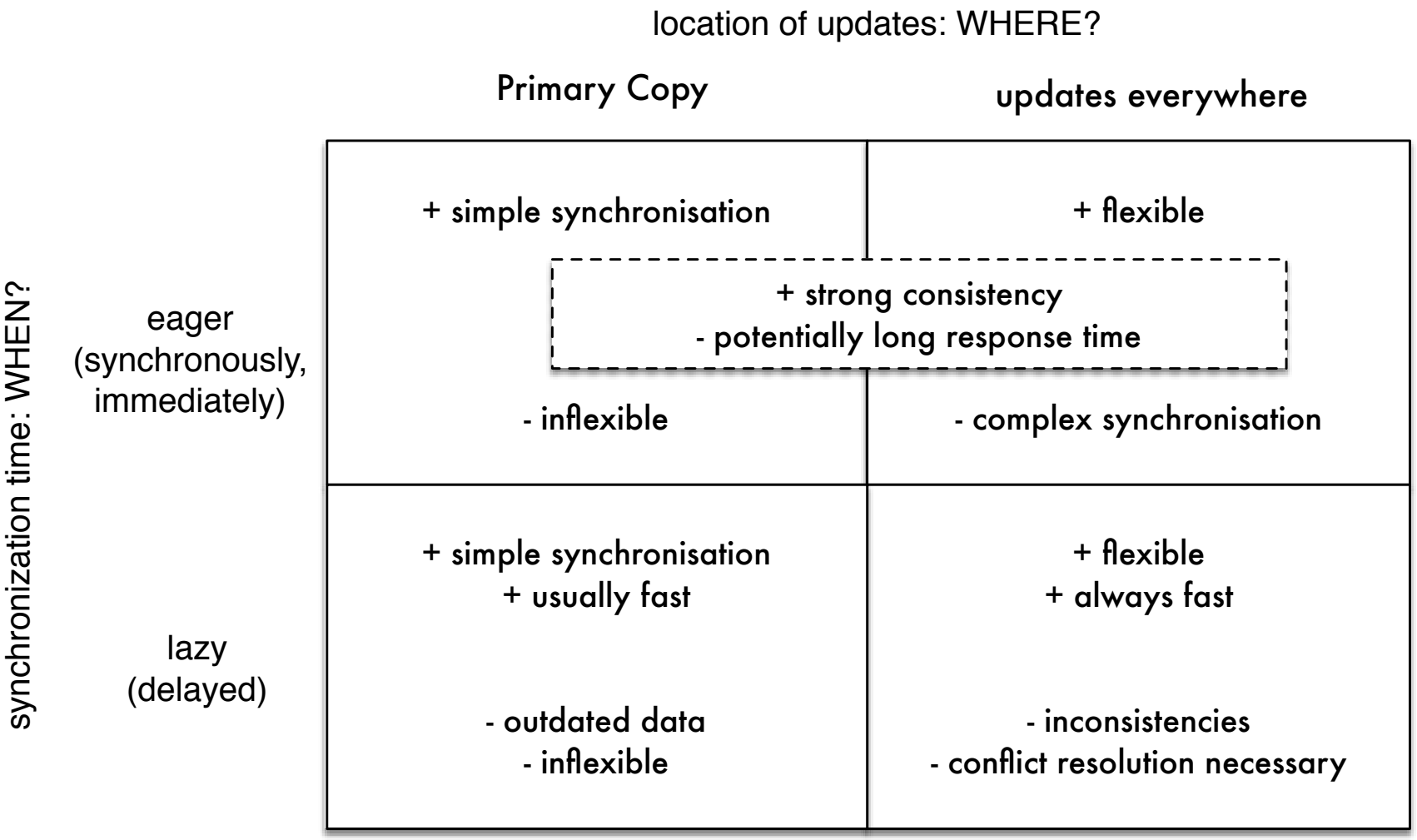
Scenario for Eventual Consistency



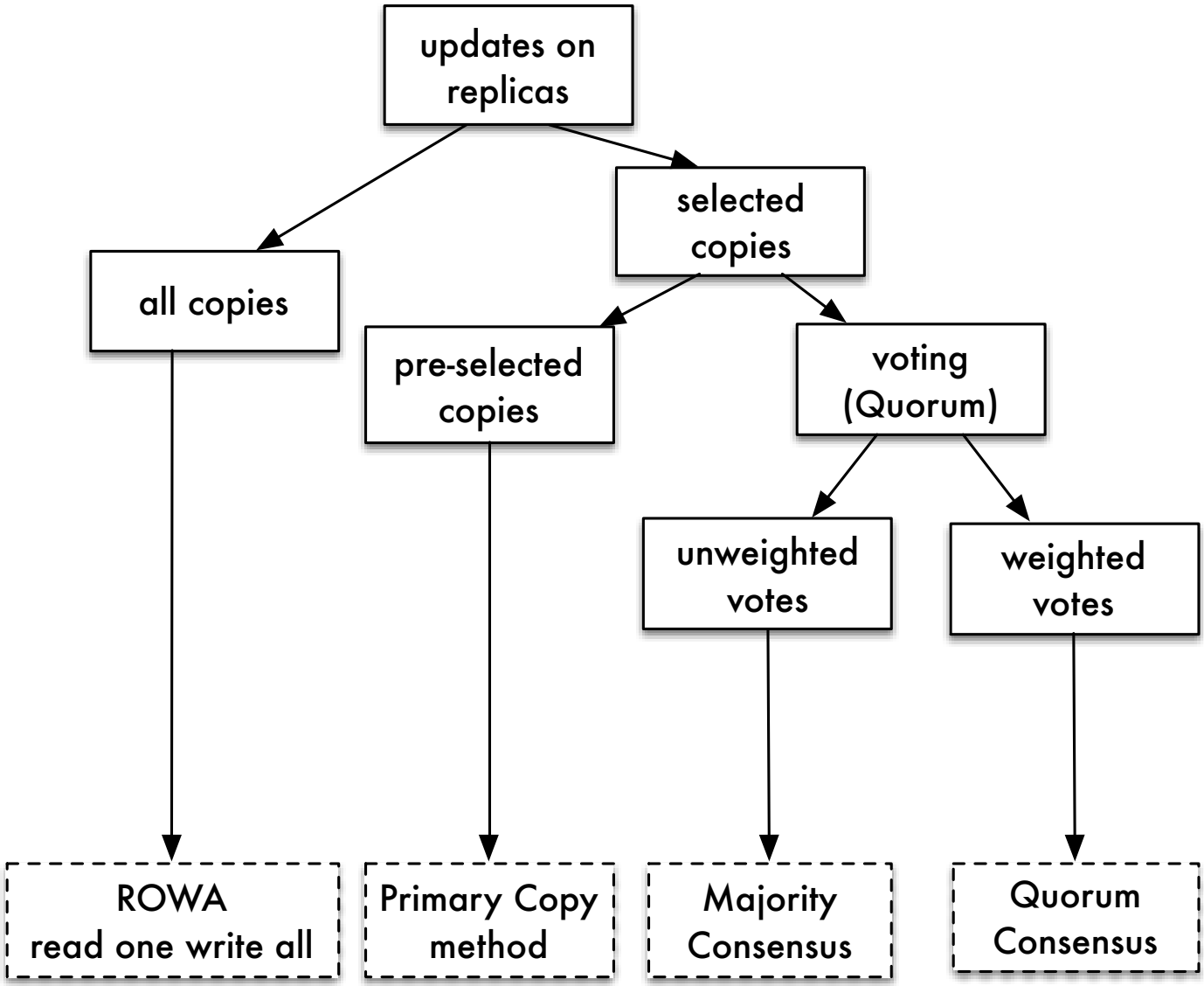
Variants of Eventual Consistency

- *Causal consistency*: If A contacts the node B and informs it of the update, B will then read D_1. However, this does not apply to the node C!
- *Read-your-writes*: After its own update, A will always read D_1. This may for example be realized by using the same nodes to process all subsequent read operations of A, so that the propagation of updates to other replicas is not important
- *Session consistency*: read-your-writes within a session
- *Monotonic reads*: If a process has seen D_k, any subsequent access will never return any D_i with $i < k$
- *Monotonic writes*: Guarantees to serialize the writes of the same process

Classification of replication synchronization



Pessimistic Replication



Transactions on replicas: Correctness

A schedule *s* on a replicated database is *1-copy-serializable* if there is a serial schedule on a non-replicable database that has the same effect as *s* on a replicable data set.

- Correctness condition for pessimistic replication

Transactions on replicas: Protocols I

Replication protocol

- *ROWA-method* (Read One, Write All): Local read and synchronic update of all replicas $\rightarrow$ extremely high complexity ; some nodes might be unavailable
- *ROWAA-method* (Read One, Write All Available)
- *Voting procedure*: Voting procedure or quorum procedure
  - Statistical number of “entitled voters“
  - Dynamical number of “entitled voters“ depends on environmental influences such as lost connections and access behavior

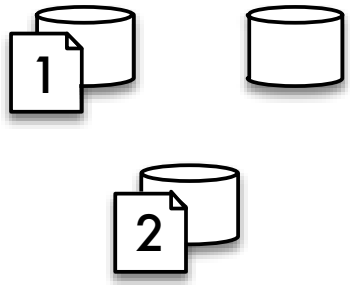
  (Weighting of votes is possible)

Transactions on replicas: Protocols II

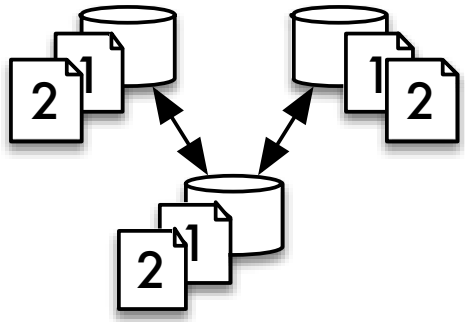
Replication protocol

- *Absolutistic approach:* e.g. primary copy method: A certain computer node updates a replica in any case. Choice is statistical or has a *token*

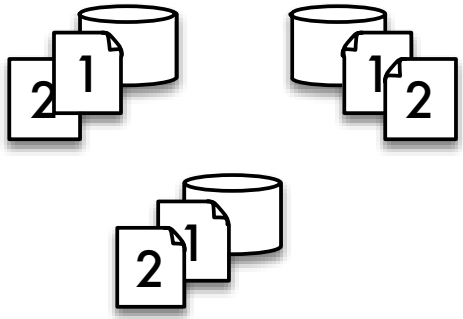
Optimistic Replication



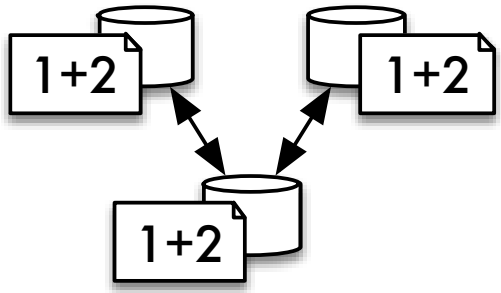
1. Submission



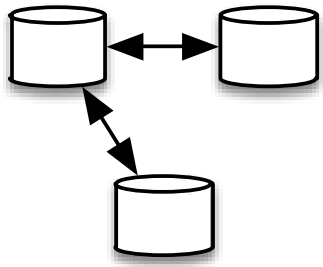
2. Propagation



3. Order Determination

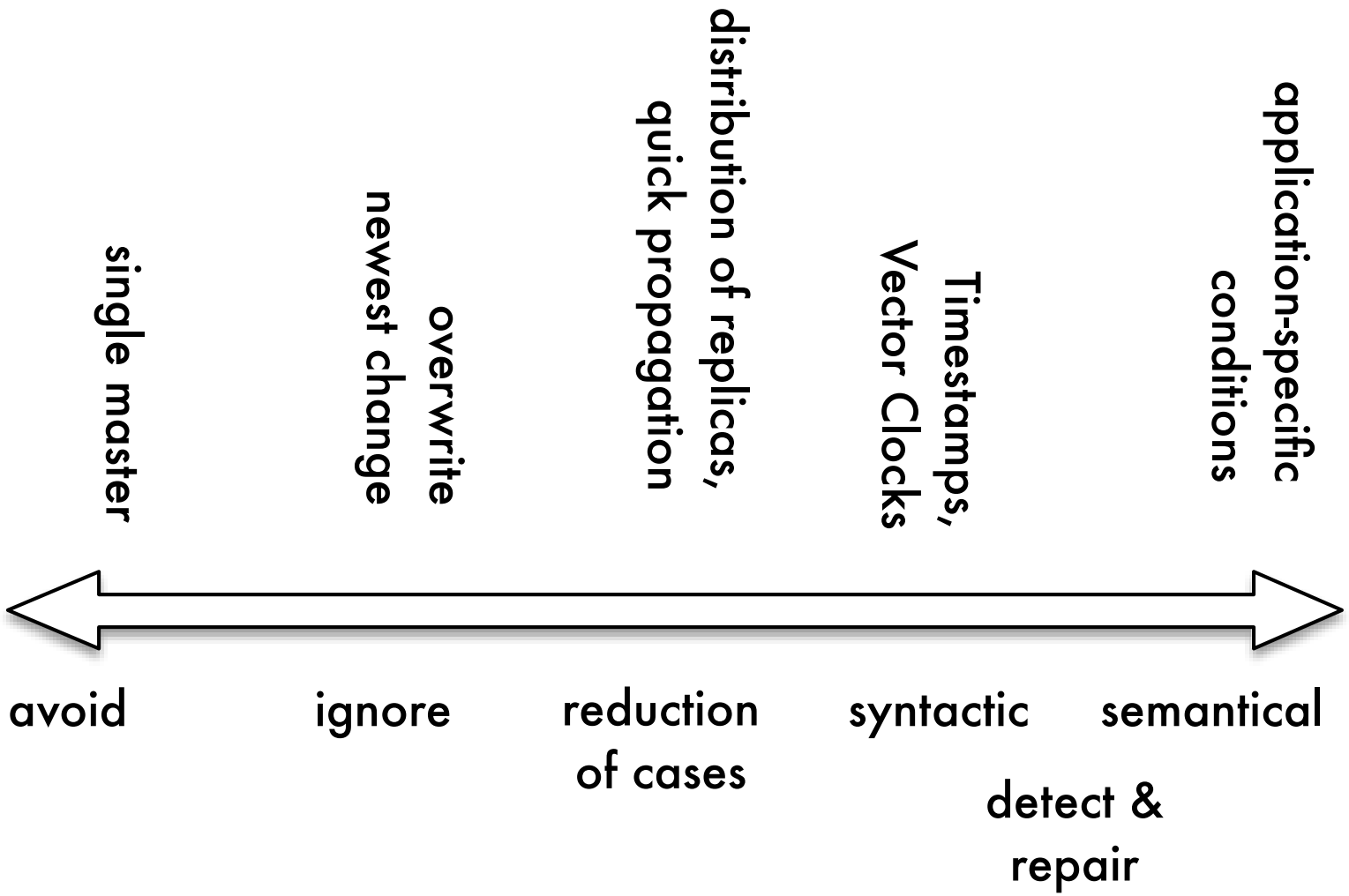


4. Conflict Resolution

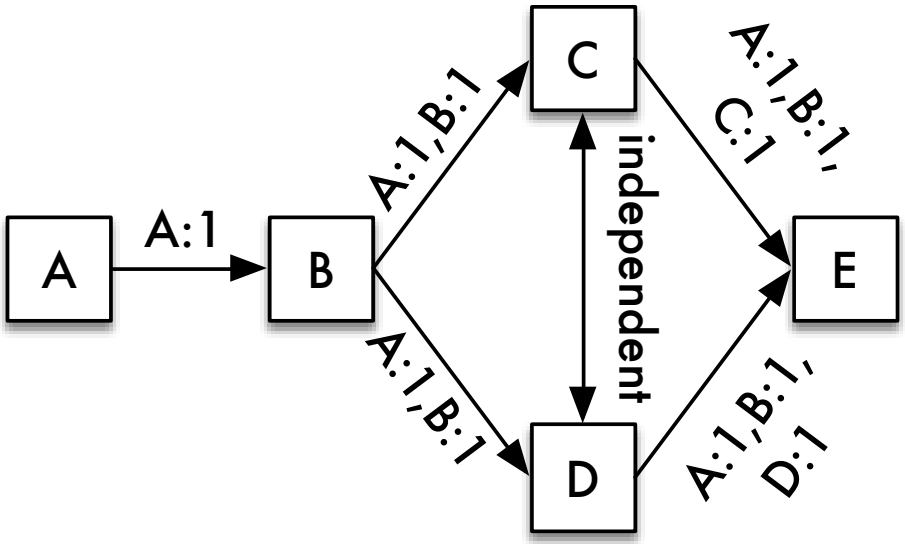


5. Commit

Optimistic conflict strategies



Vector clocks



- A clock per replicated object (timestamp)
  - Same values: Identical version
  - Dominance: Update of the older version
  - Incomparable: Conflict