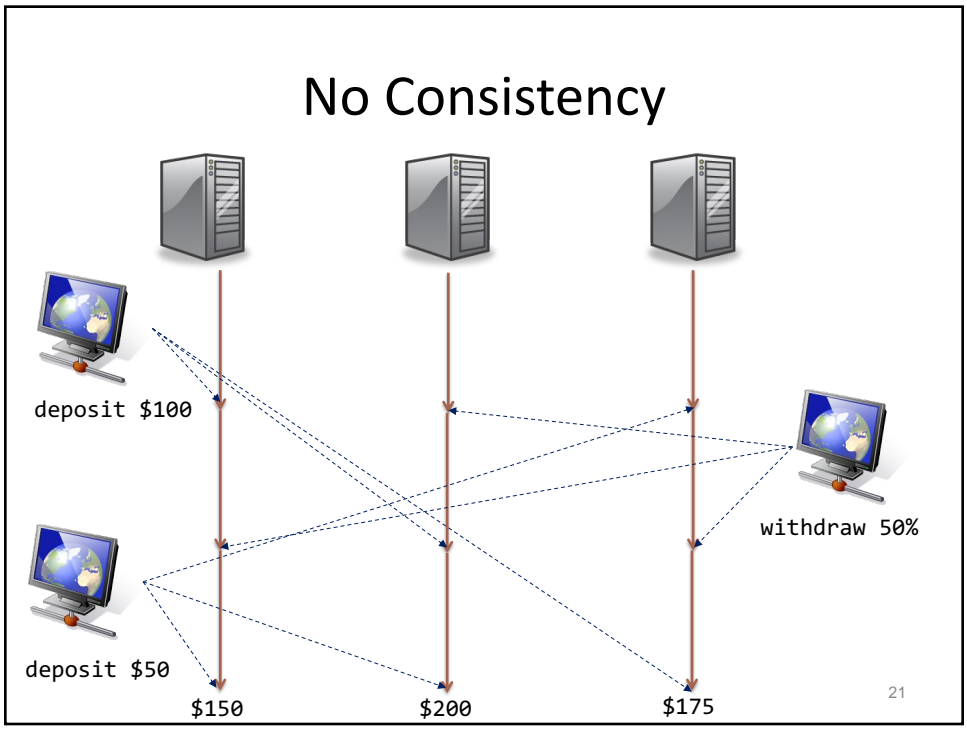
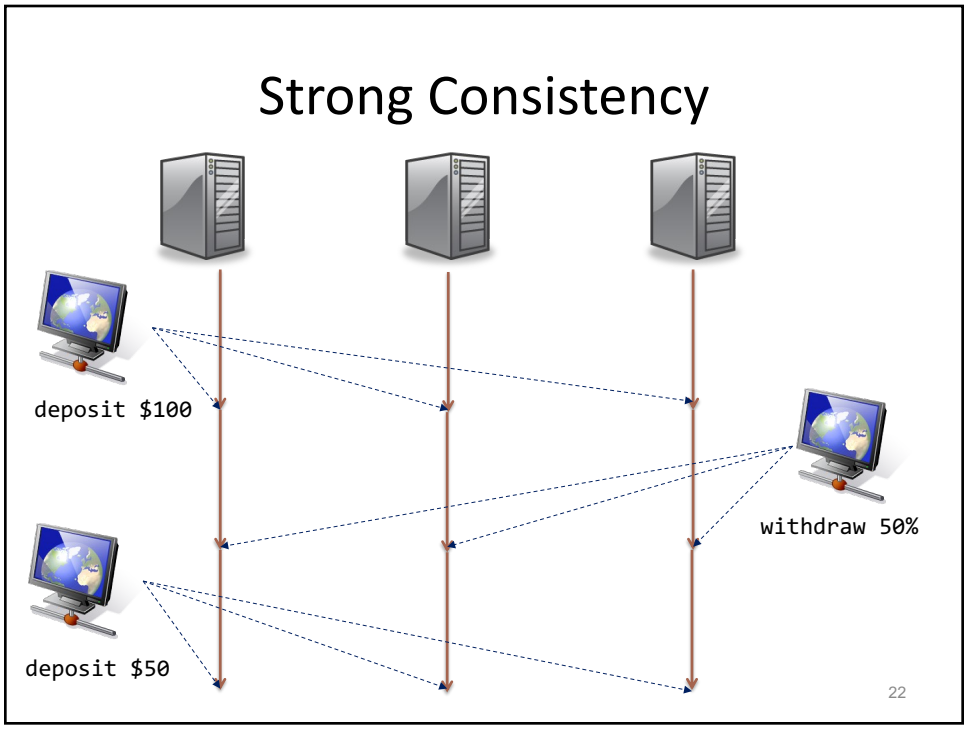
EVENTUAL CONSISTENCY

Strong Consistency

- As soon as one client successfully completes a write, all clients reading from the database must be able to see the value just written (*Recency*)
- Replicated database
- Must not read stale copy of data

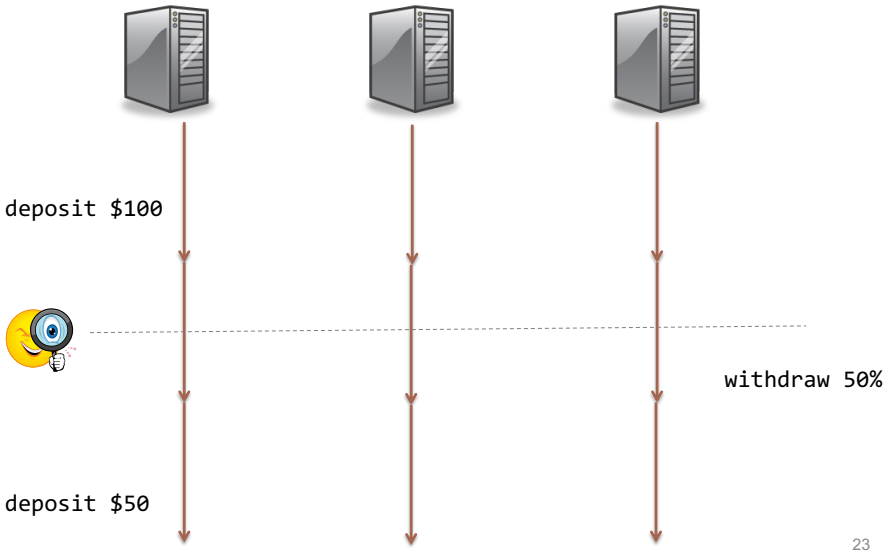

No Consistency
deposit $100
withdraw 50%
deposit $50
$150
$200
$175
21

21


Strong Consistency
deposit $100
withdraw 50%
deposit $50
22

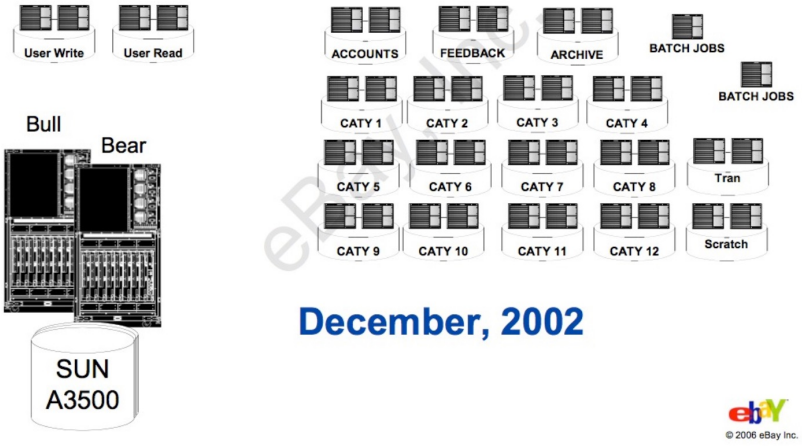
22

Strong Consistency



23

23



24

24

BASE: Basically Available, Soft-state, Eventually consistent

- "BASE is diametrically opposed to ACID. Where ACID is pessimistic and forces consistency at the end of every operation, BASE is optimistic and accepts that the database consistency will be in a state of flux. Although this sounds impossible to cope with, in reality it is quite manageable and leads be obtained with ACID."
  - "BASE: An Acid Alternative," Dan Pritchett, eBay

25

25

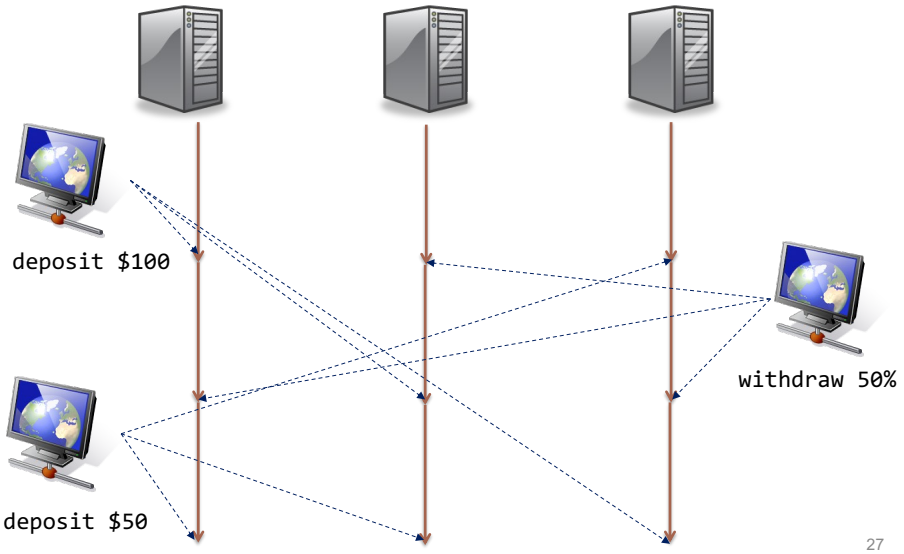
Relaxed Consistency

- Best effort consistency
  - Ex: Grapevine email transport
  - Replicas may unintentionally fail to converge
  - Ex: Update operations are not deterministic
  - Ex: Updates performed in different orders on replicas

26

26

Relaxed Consistency



27

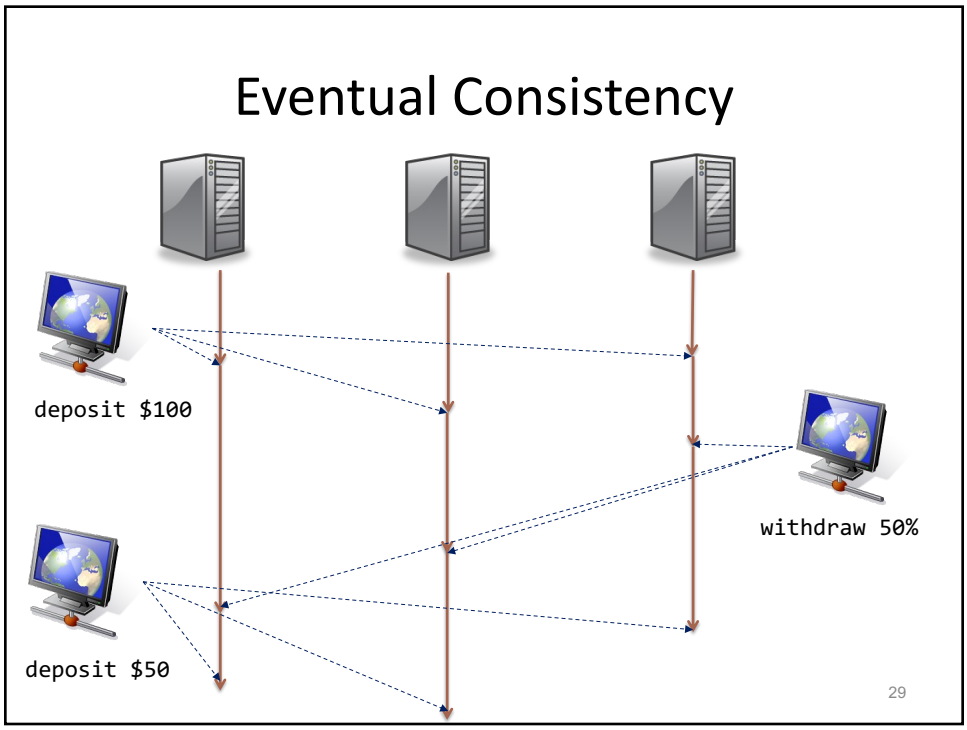
27

Strong and Eventual Consistency

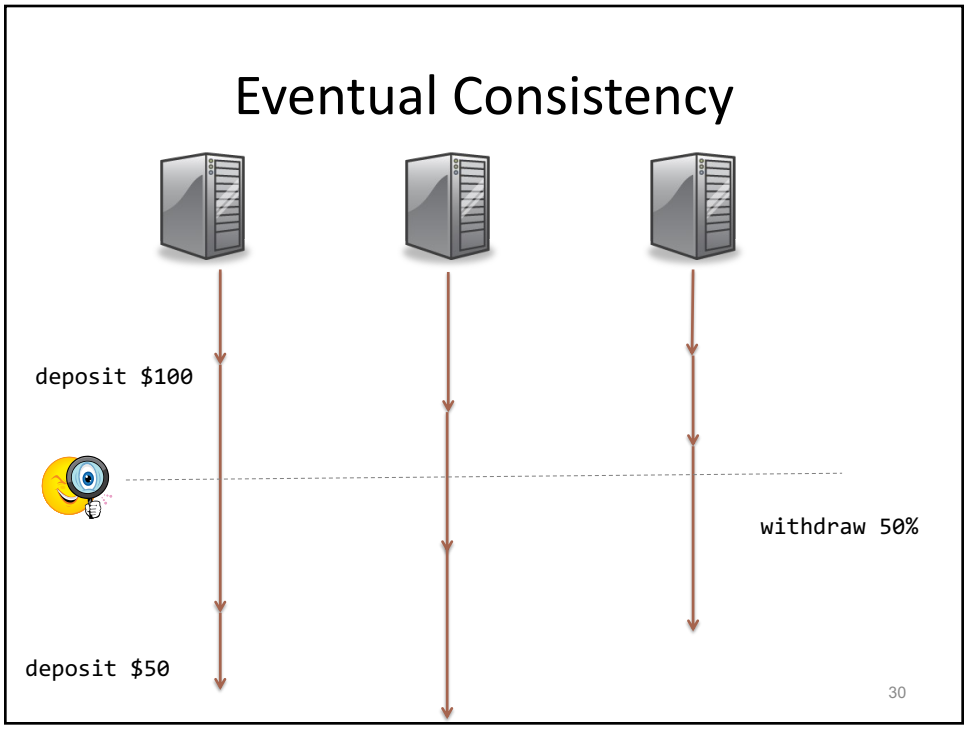
- **Strong Consistency:** Effect of an update is visible by any operation that follows it
  - No stale reads
- **Eventual Consistency:**
  - **Consistent ordering:** Updates are done in same order on all replicas
  - **Total propagation:** Updates are performed on all replicas *eventually*

28

28


Eventual Consistency
deposit $100
withdraw 50%
deposit $50
29

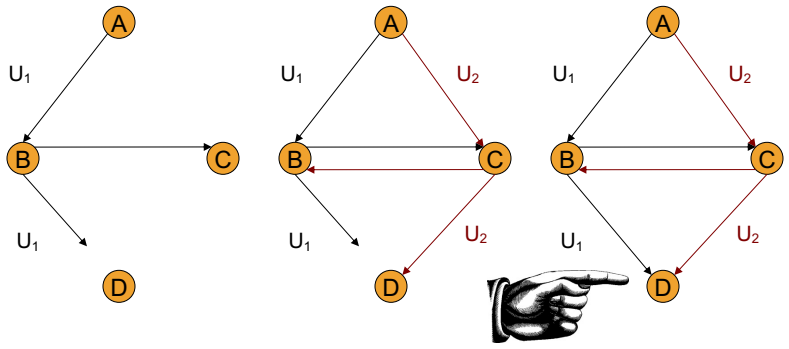
29


Eventual Consistency
deposit $100
withdraw 50%
deposit $50
30

30

Eventual Consistency

- Eventual consistency
  - Ordering of updates important

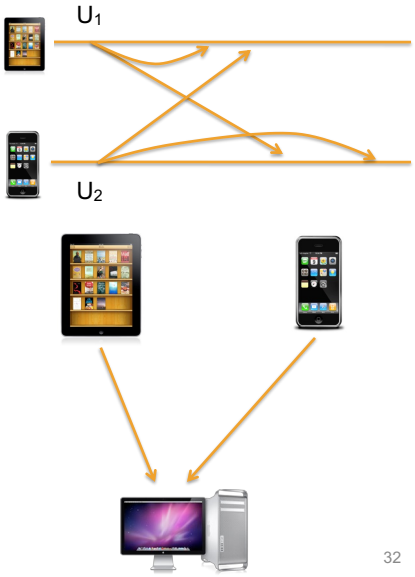


31

31

Out-of-order updates?

- Delay delivery
  - Ordered multicast
  - Delay even for local replica!
  - Not practical for disconnected
- Tentative update
  - Resolve with other updates later
    - Undo, perform missing updates, redo
  - Information about non-conflicting updates
    - API to specify



32

32

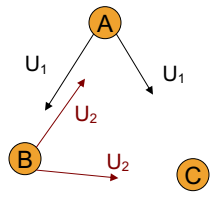
Causal Consistency

- Strong & Eventual Consistency:
  - **Total order** on all updates
  - Causal: Weaken the required ordering
- What if two updates unrelated?
  - E.g. different DB records, or …
  - No **causal** relationship
- Advantage: No need to order "concurrent" updates
  - Unless application **requires** total order

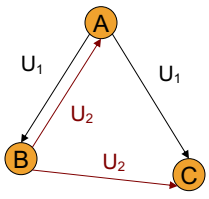
33

33

Causal Consistency



$U_1 \parallel U_2$



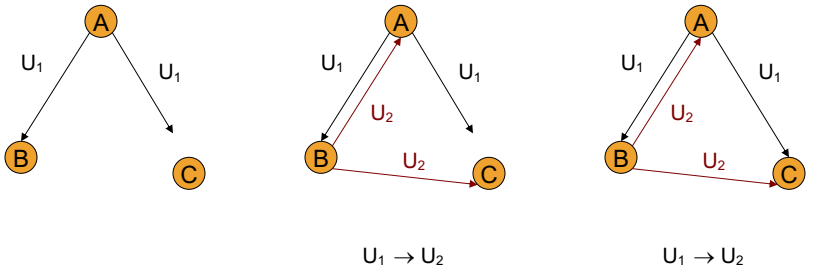
$U_1 \parallel U_2$

Does order of delivery matter at C?
Strong consistency: ✓
Causal consistency: ✗

34

34

Causal Consistency



$U_1 \rightarrow U_2$ $U_1 \rightarrow U_2$

35

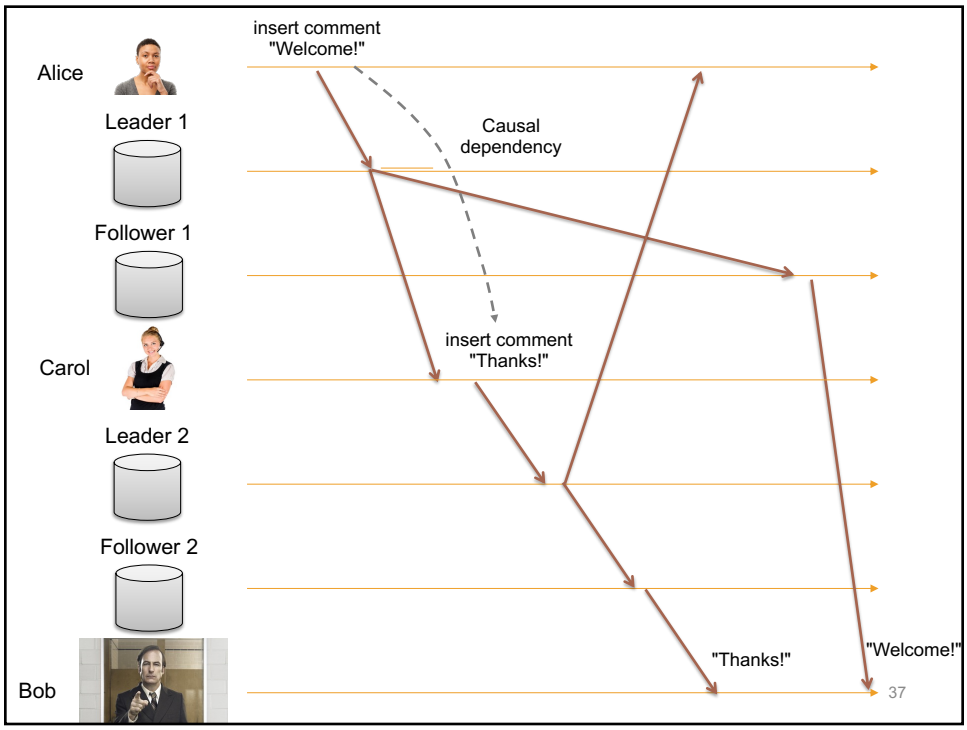
35

Consistent Prefix Reads

- Issue: Violation of causal dependency

36

36



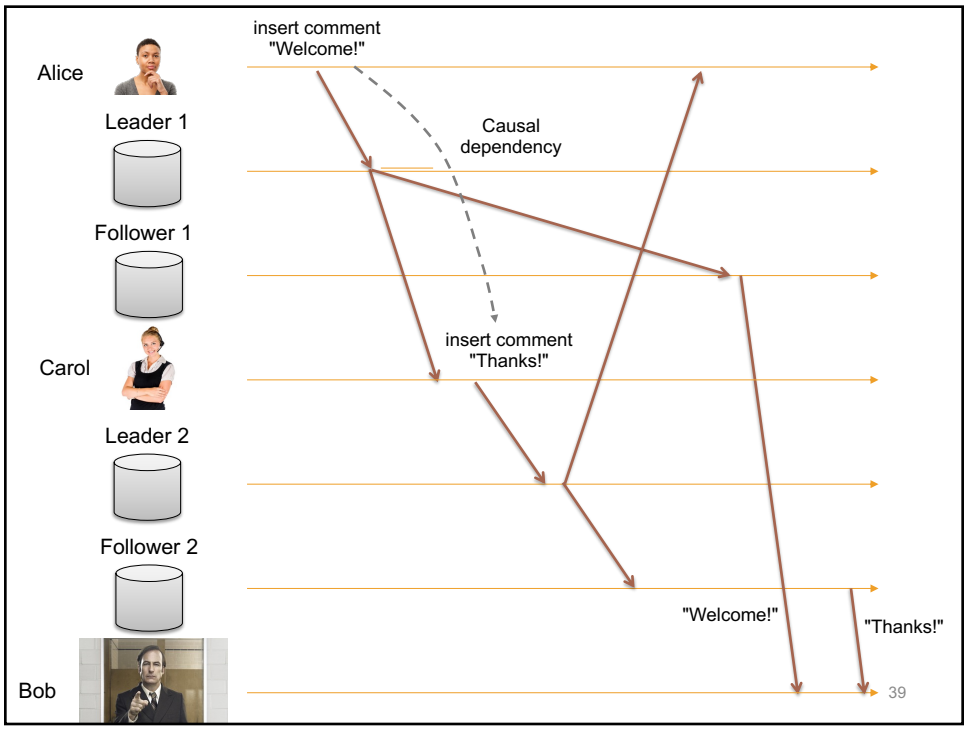
37

Consistent Prefix Reads

- Issue: Violation of causal dependency
- CPR: Causally related writes are read in that order
  - "Welcome!" written before "Thanks!"
- Guarantee: Causally consistent version of DB
  - May not be current version of master...
  - …but it was a version *at some point*
  - …related to Snapshot Isolation

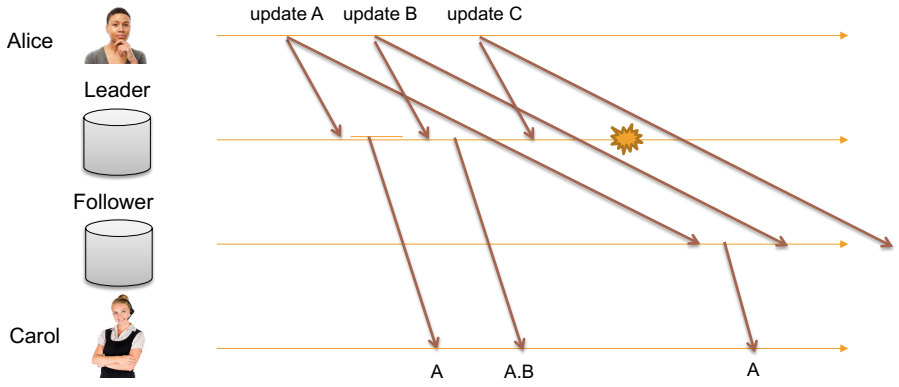
38

38



39

Consistent Prefix Reads



40

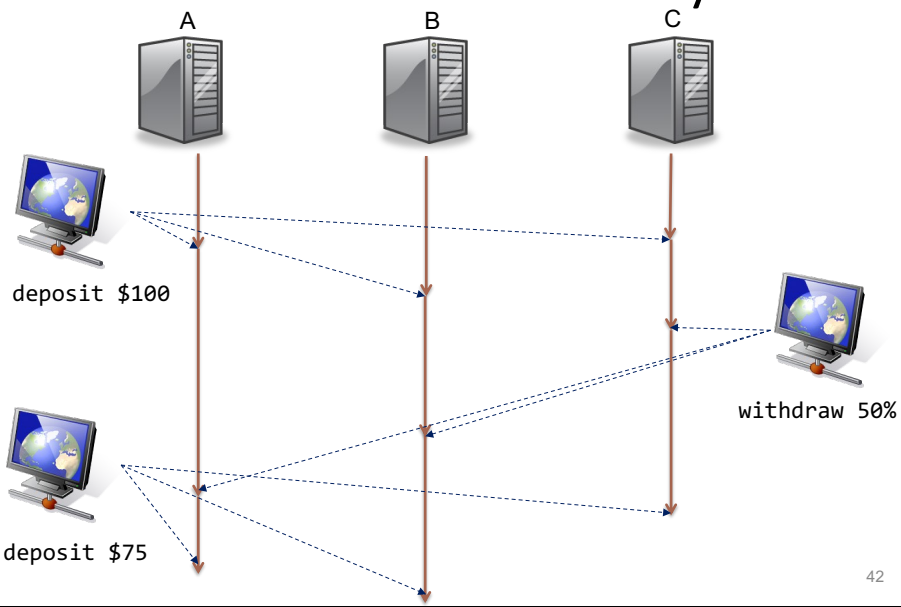
Consistent Prefix Reads

- Issue: Violation of causal dependency
- CPR: Causally related writes are read in that order
  - "Welcome!" written before "Thanks!"
- Guarantee: Causally consistent version of DB
- No bound on staleness from any replica
- …only global ordering

41

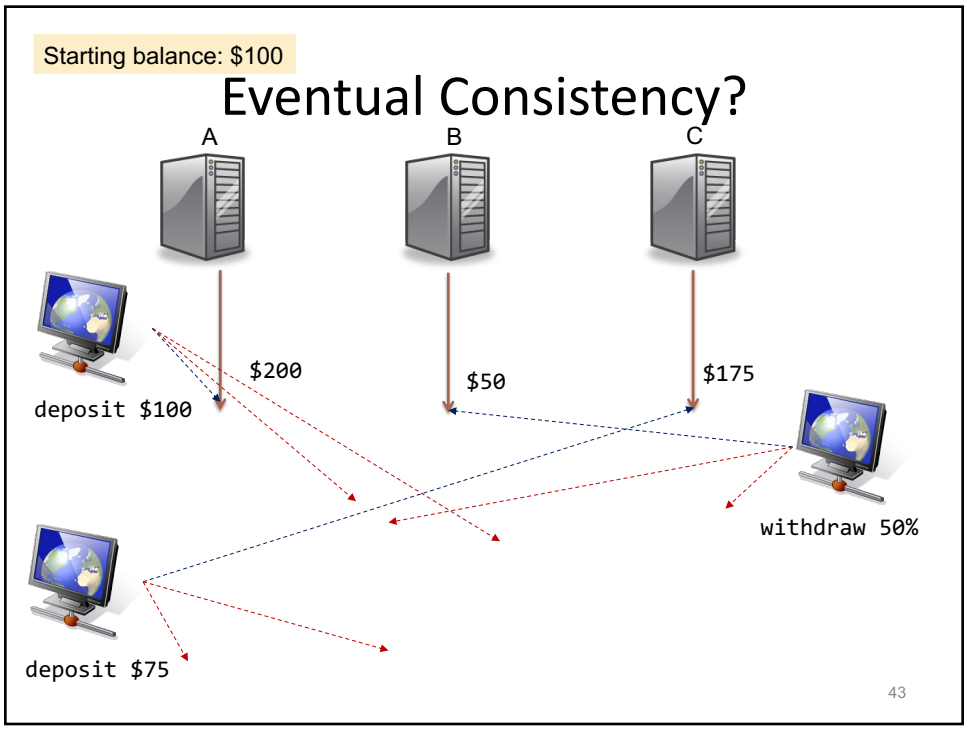
41

Eventual Consistency

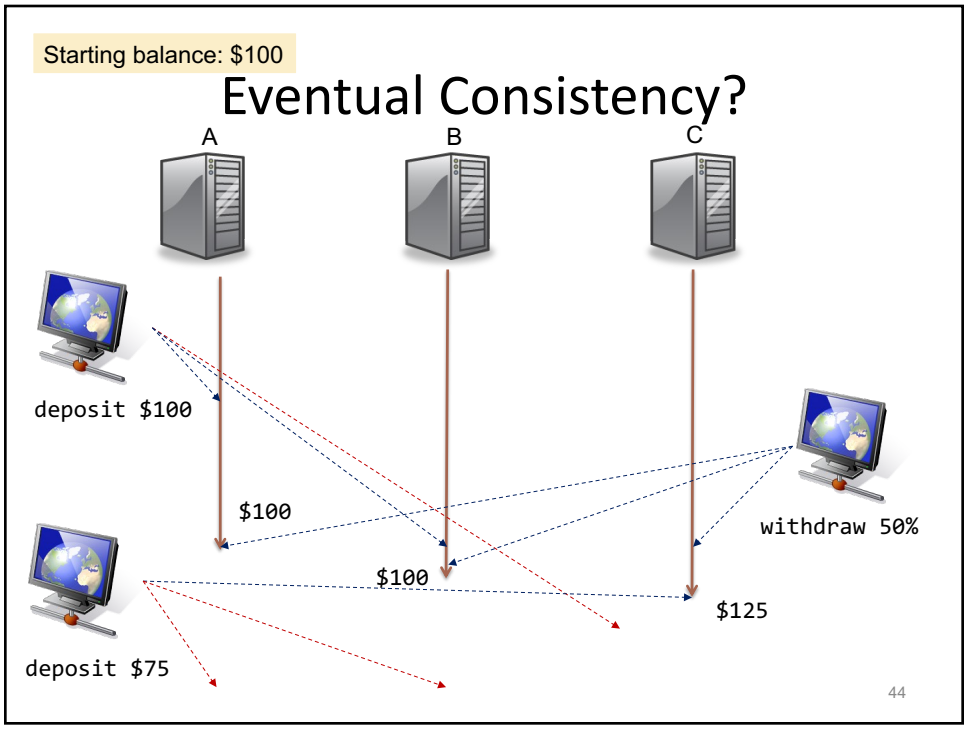


42

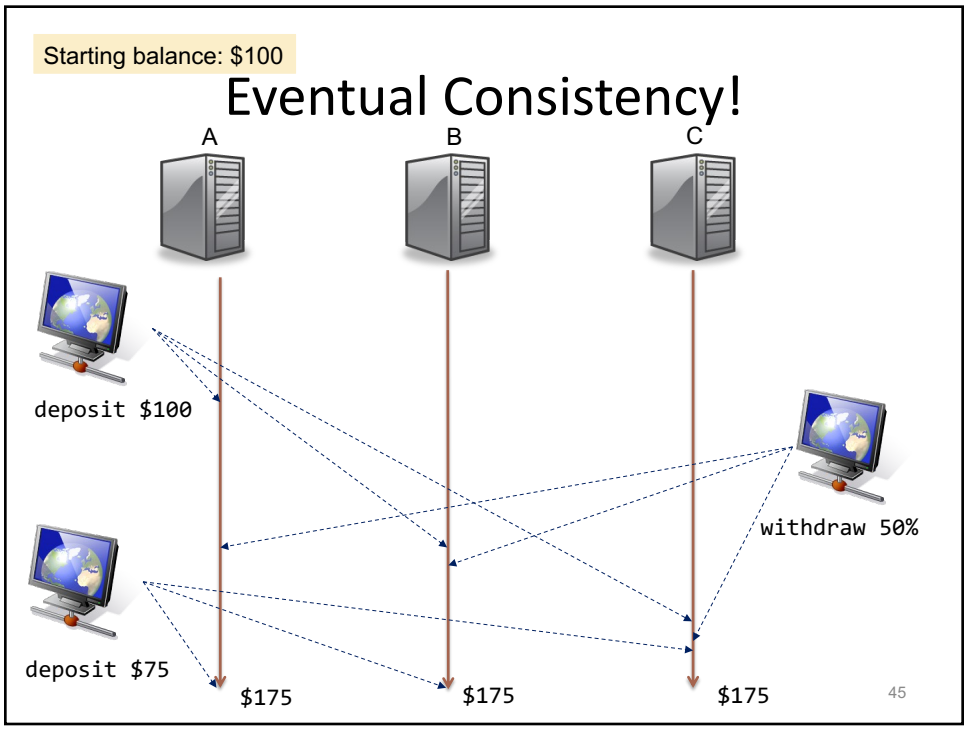
42


Starting balance: $100
Eventual Consistency?
A
B
C
$200
$50
$175
deposit $100
withdraw 50%
deposit $75
43

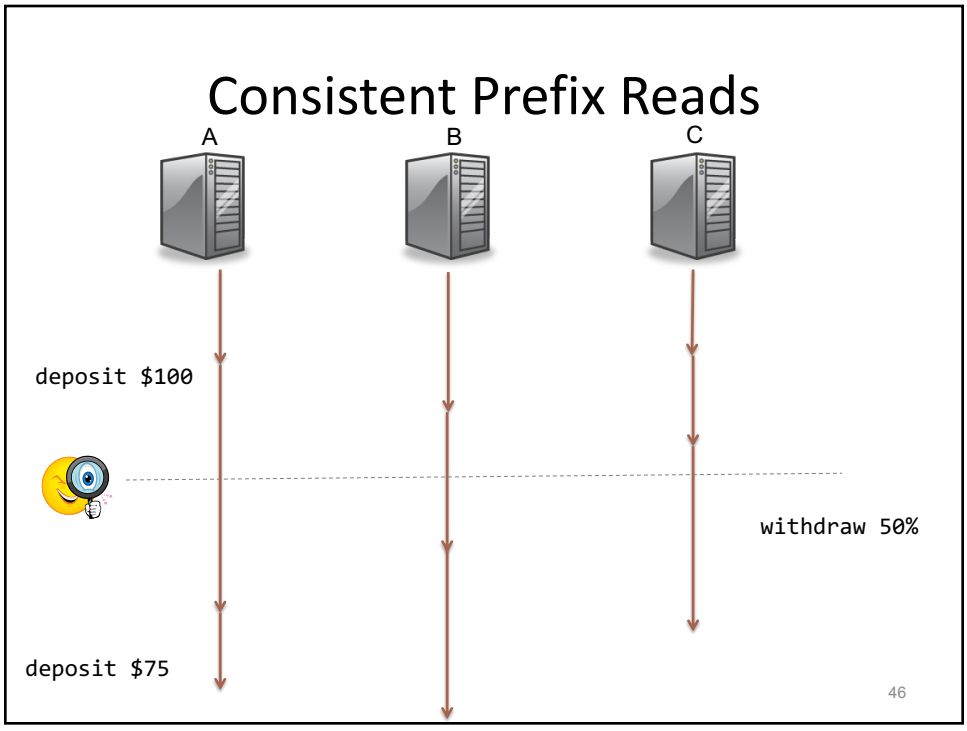
43


Starting balance: $100
Eventual Consistency?
A
B
C
deposit $100
$100
withdraw 50%
$100
$125
deposit $75
44

44


Starting balance: $100
Eventual Consistency!
A
B
C
deposit $100
withdraw 50%
deposit $75
$175
$175
$175
45

45


Consistent Prefix Reads
A
B
C
deposit $100
withdraw 50%
deposit $75
46

46