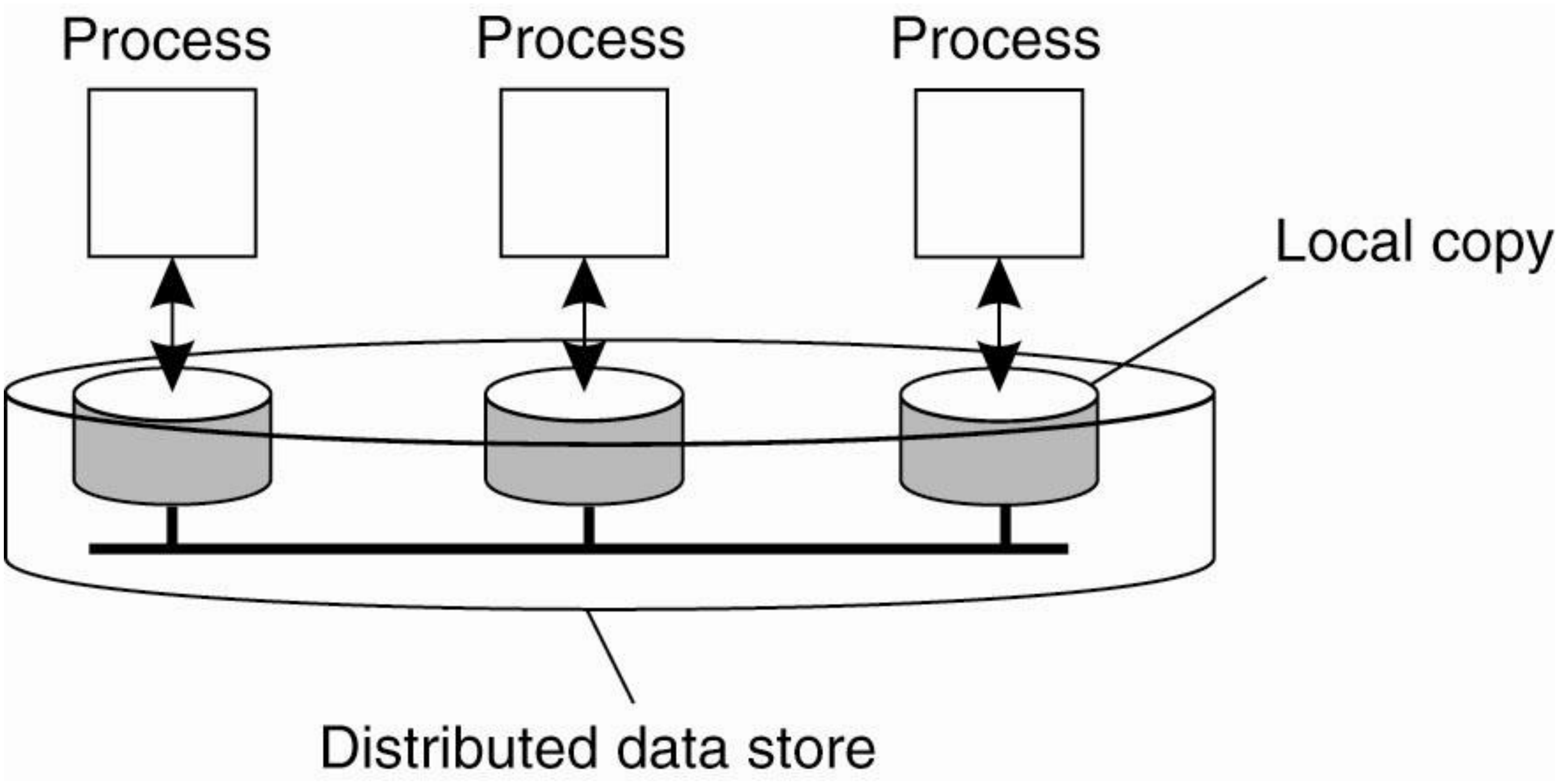
MYE017 Distributed Systems

Kostas Magoutis
magoutis@cse.uoi.gr
http://www.cse.uoi.gr/~magoutis

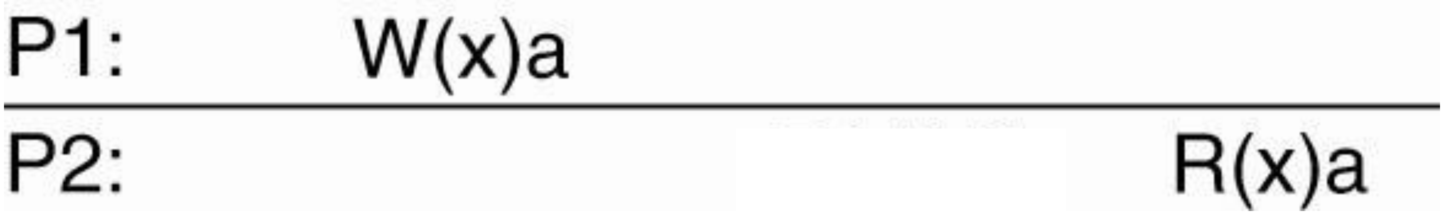
Data-centric consistency models



The general organization of a logical data store, physically distributed and replicated across multiple processes

Strict consistency

Any read on a data item x returns the value of the most recent write to x

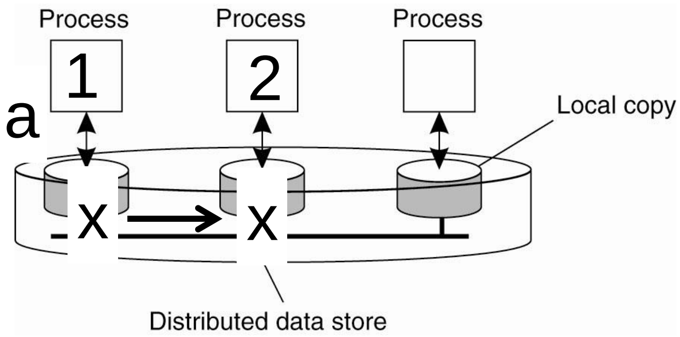


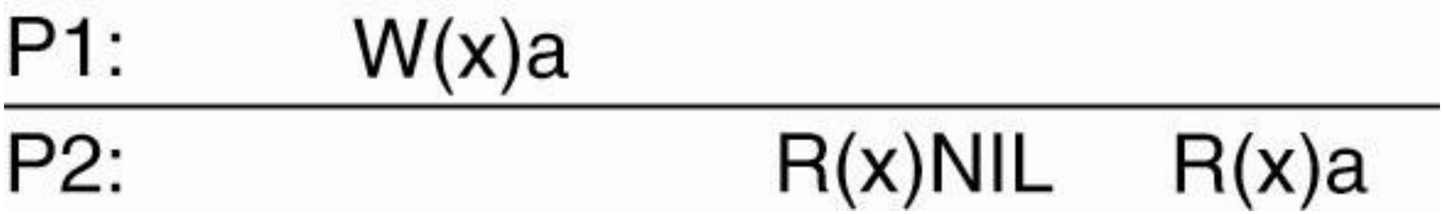
P1: W(x)a

P2: R(x)a

Behavior of two processes operating on the same data item. The horizontal axis is time.

Weaker consistency





```
P1:      W(x)a
P2:                        R(x)NIL     R(x)a
```

Behavior of two processes operating on the same data item. The horizontal axis is time.

Sequential consistency

A data store is *sequentially consistent (SC)* when:

The result of <u>any execution</u> on the data store is the same as if the read and write operations by all processes

- Were executed in some sequential order *on a single copy* of the store
- The operations in this sequence appear in the order specified by each individual process' program

An execution:

E1:	P1:	W(x)a				
E2:	P2:		W(x)b			
E3:	P3:			R(x)b		R(x)a
E4:	P4:				R(x)b	R(x)a

A sequential order: (*history H*)

$W_1(x)b$

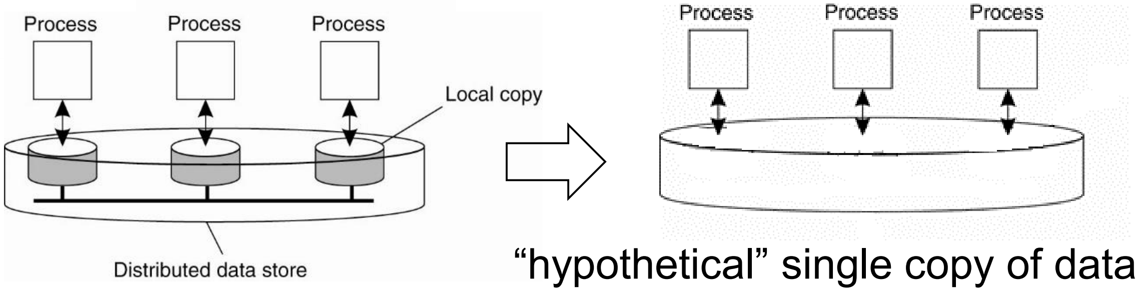
$R_3(x)b$

$R_4(x)b$

$W_2(x)a$

$R_3(x)a$

$R_4(x)a$

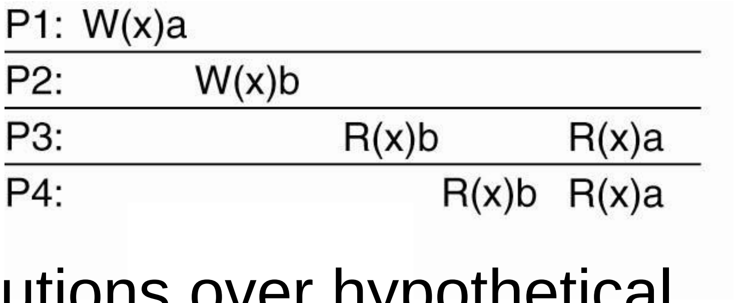


"hypothetical" single copy of data store

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

More formally

- E_i : Sequence of read or write operations executed by process P_i over data store S
  - E.g. $E_3 = R_3(x)b\ R_3(x)a$



P1: W(x)a

P2: W(x)b

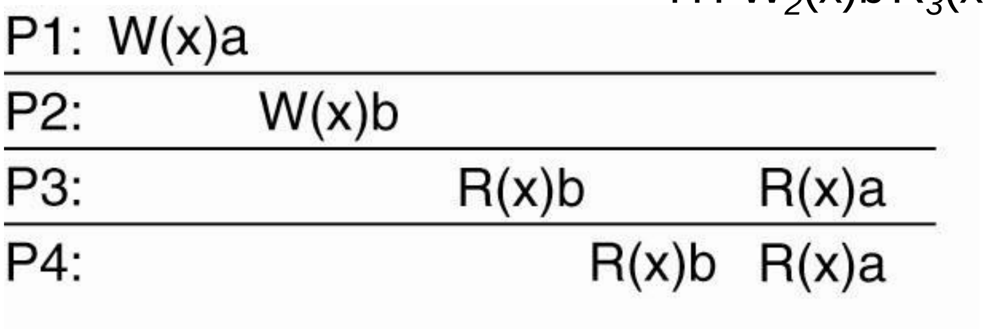
P3: R(x)b R(x)a

P4: R(x)b R(x)a

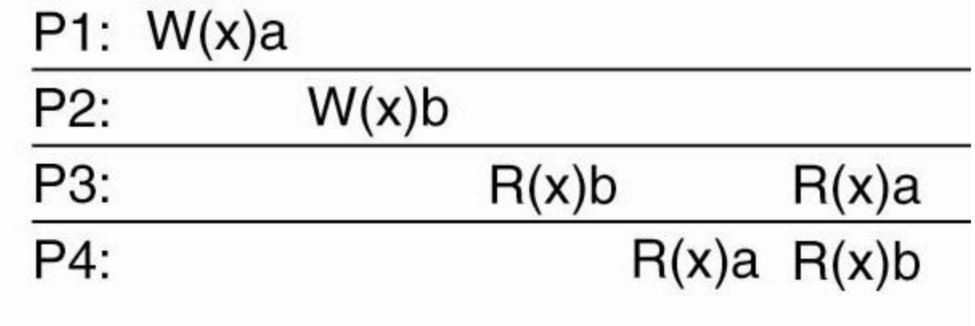
- History H : sequence of op executions over hypothetical centralized data store S′
  - H is an interleaving of E_i i=1, .., n
- All acceptable histories H must respect
  - The order of operations in individual executions
  - Data coherence (read last value written)

Sequential consistency

H : $W_2(x)b\ R_3(x)b\ R_4(x)b\ W_1(x)a\ R_3(x)a\ R_4(x)a$


P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)b R(x)a

(a)


P1: W(x)a
P2: W(x)b
P3: R(x)b R(x)a
P4: R(x)a R(x)b

(b)

(a) A sequentially consistent data store.

(b) A data store that is not sequentially consistent.

Sequential consistency

Data type: 4-location byte-valued read/write snapshot register

location	value
1	0
2	0
3	0
4	0

A multi-location read-write memory has

- a set of locations (or addresses)
- operations such as
    - read(*a*)
    - write(*a*, *w*)
    - snapshot()
- snapshot() returns a set of values, one for each location

Sequential consistency

Two replicas at sites X and Y, clients located at T and U

a legal history

(write(1, 5), "OK")

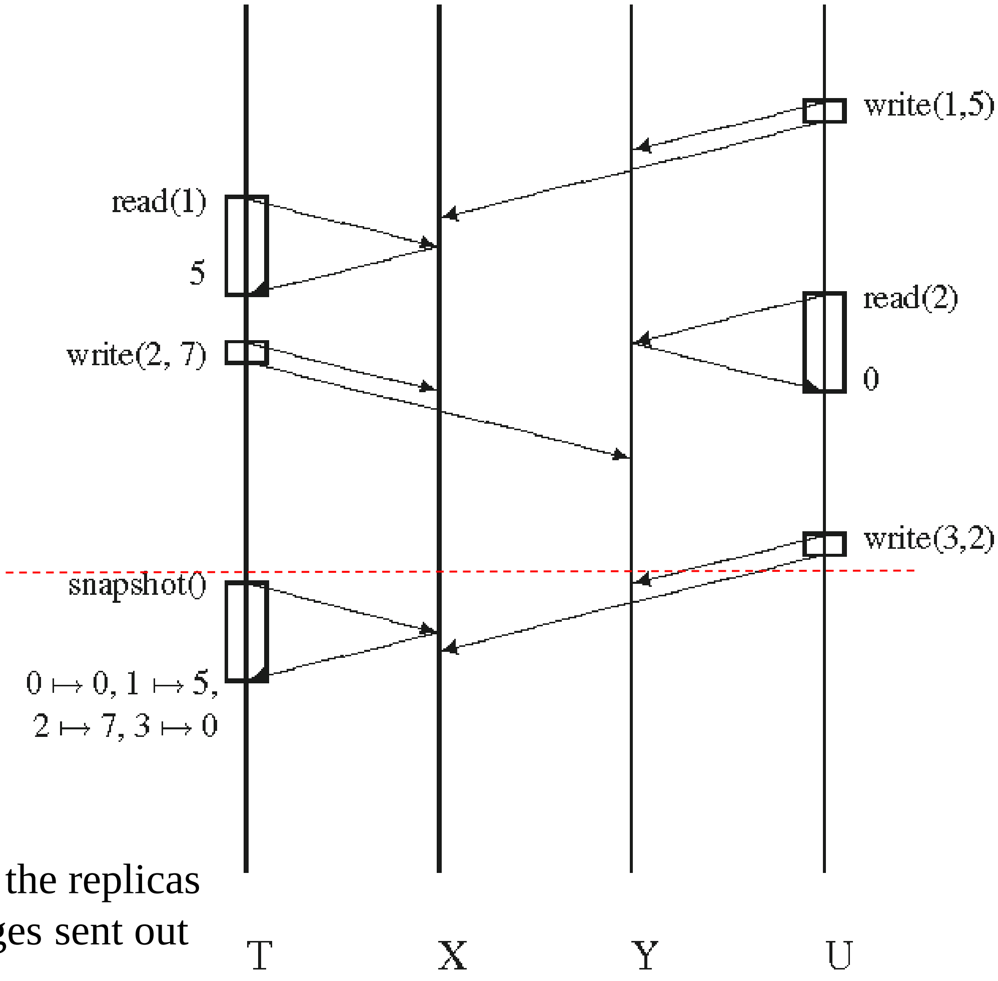
(read(1), 5)

(read(2), 0)

(write(2, 7), "OK")

(snapshot(), $(0 \mapsto 0, 1 \mapsto 5, 2 \mapsto 7, 3 \mapsto 0)$)

(write(3, 2), "OK")



Implementation rules:

- each read or snapshot is done on one replica
- each write is done on both replicas
- different writes are done in the same order at the replicas
- a write returns to the client as soon as messages sent out

From "Consistency Models for Replicated Data", A. D. Fekete, Krithi Mamamritham

B. Charron-Bost, F. Pedone, and A. Schiper (Eds.): Replication, LNCS 5959, pp. 1–17, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Linearizability

A data store is *linearizable* when:

The result of any execution on the data store is the same as if the (read/write) operations by all processes

- Were executed in some sequential order on a single copy of the store
- The operations of each individual process appear in this sequence in the order specified by its program

Additionally

- If the duration of $OP_1(x)$ is entirely before the duration of $OP_2(y)$ (in same or different clients) then $OP_1(x)$ must precede $OP_2(y)$ in this seq. order

Linearizable execution

Implementation rules:

- each read or snapshot is done on one replica
- each write is done on both replicas
- different writes are done in the same order at the replicas
- <u>a write doesn’t return to the client until acked</u>

a legal history

(write(1, 5), “OK”)

(read(1), 5)

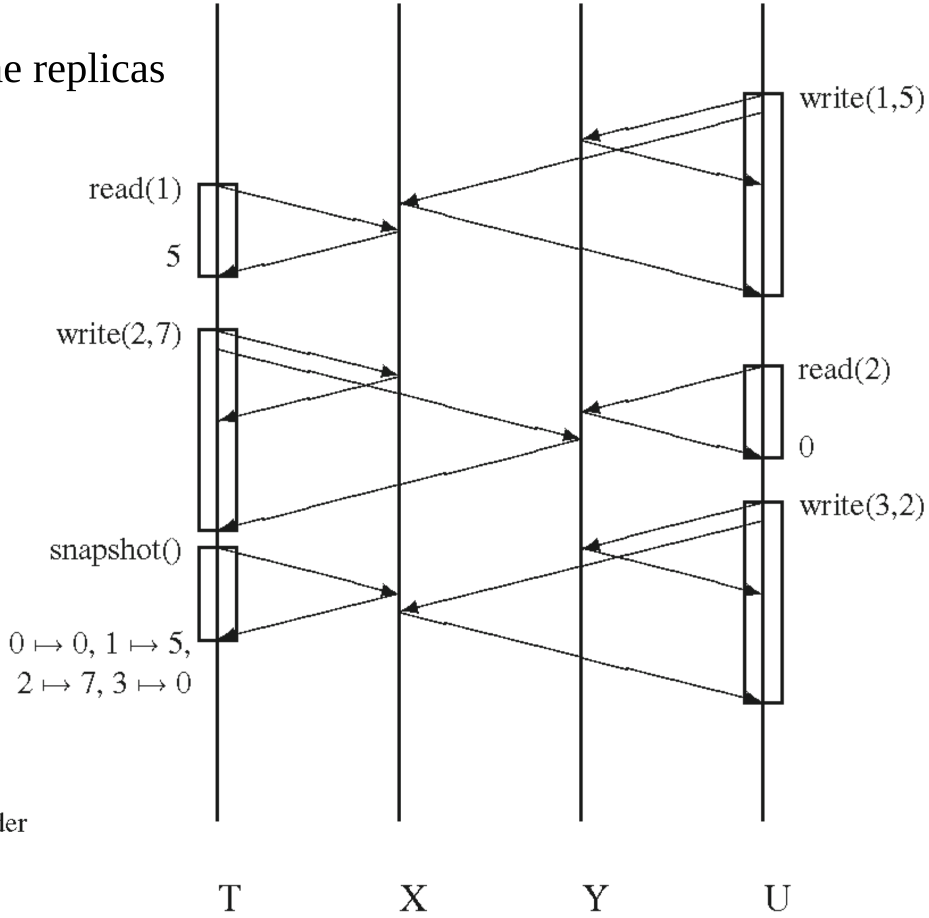
(read(2), 0)

(write(2, 7), “OK”)

(snapshot(), $(0 \mapsto 0, 1 \mapsto 5, 2 \mapsto 7, 3 \mapsto 0)$)

(write(3, 2), “OK”)

the order of operations as they occur in the sequence must not contradict any order information visible to an observer of the system execution.

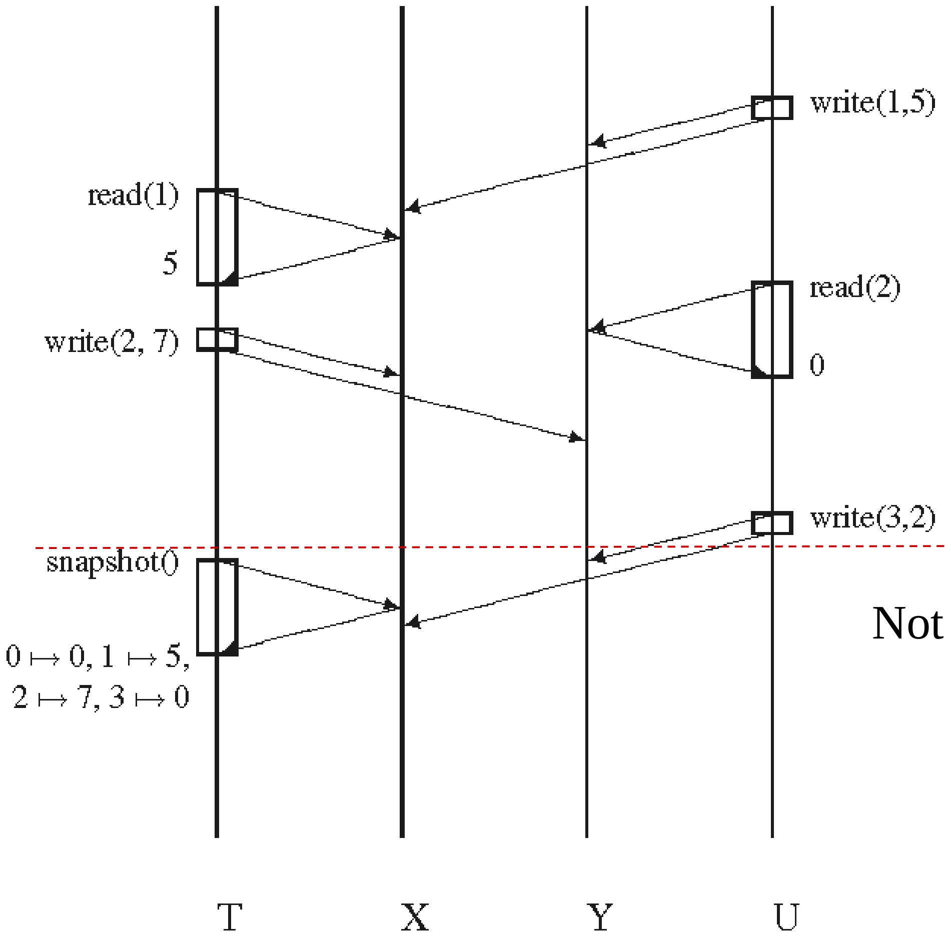


From “Consistency Models for Replicated Data”, A. D. Fekete, Krithi Mamamritham

B. Charron-Bost, F. Pedone, and A. Schiper (Eds.): Replication, LNCS 5959, pp. 1–17, 2010.
© Springer-Verlag Berlin Heidelberg 2010

SC but not linearizable

4-location byte-valued snapshot memory



a legal history

(write(1, 5), “OK”)

(read(1), 5)

(read(2), 0)

(write(2, 7), “OK”)

(snapshot(), (0 $\mapsto$ 0, 1 $\mapsto$ 5, 2 $\mapsto$ 7, 3 $\mapsto$ 0)

(write(3, 2), “OK”)

Not linearizable!!

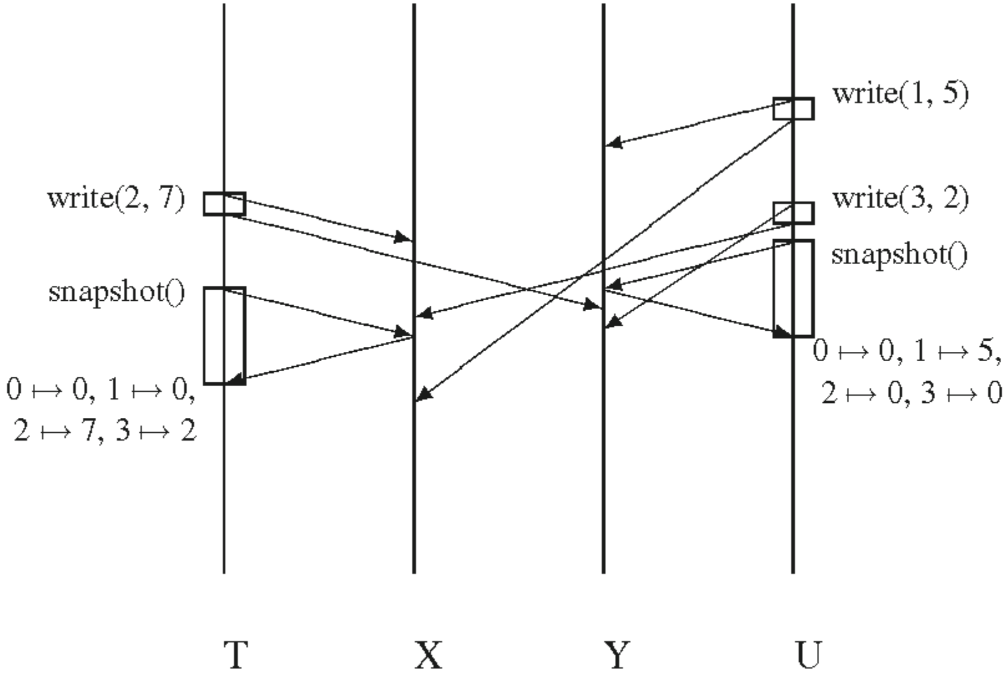
From “Consistency Models for Replicated Data”, A. D. Fekete, Krithi Mamamritham

B. Charron-Bost, F. Pedone, and A. Schiper (Eds.): Replication, LNCS 5959, pp. 1–17, 2010.
© Springer-Verlag Berlin Heidelberg 2010

Weak consistency

Implementation rules:

- each read or snapshot is done on one replica
- each write is done on both replicas
- ~~different writes are done in the same order at the replicas~~
- a write returns to the client as soon as messages sent out



Cannot find a legal history that would satisfy either linearizability or SC conditions