Consistency examples

COS 418: Distributed Systems
Precept 5

Themis Melissaris

Plan

- Midterm poll
- Consistency examples

Fill out this poll:

http://tinyurl.com/zdeq4lr

Linearizability

Strong consistency = linearizability

- Linearizability (Herlihy and Wang 1991)
  1. All servers execute all ops in *some* identical sequential order
  2. Global ordering preserves each client's own local ordering
  3. Global ordering preserves real-time guarantee
     - All ops receive global time-stamp using a sync'd clock
     - If $ts_{op1}(x) < ts_{op2}(y)$, OP1(x) precedes OP2(y) in sequence
- Once write completes, all later reads (by wall-clock start time) should return value of that write or value of later write.
- Once read returns particular value, all later reads should return that value or value of later write.

Sequential Consistency

Is that valid?

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

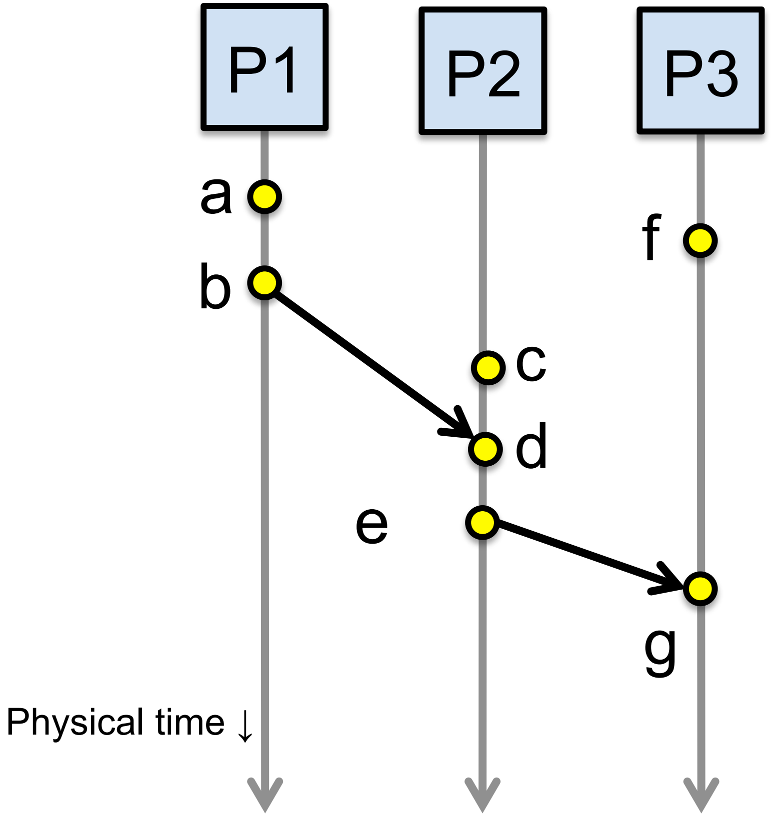
No. Why?

Ordering of the Write operations needs to be preserved

Causal Consistency

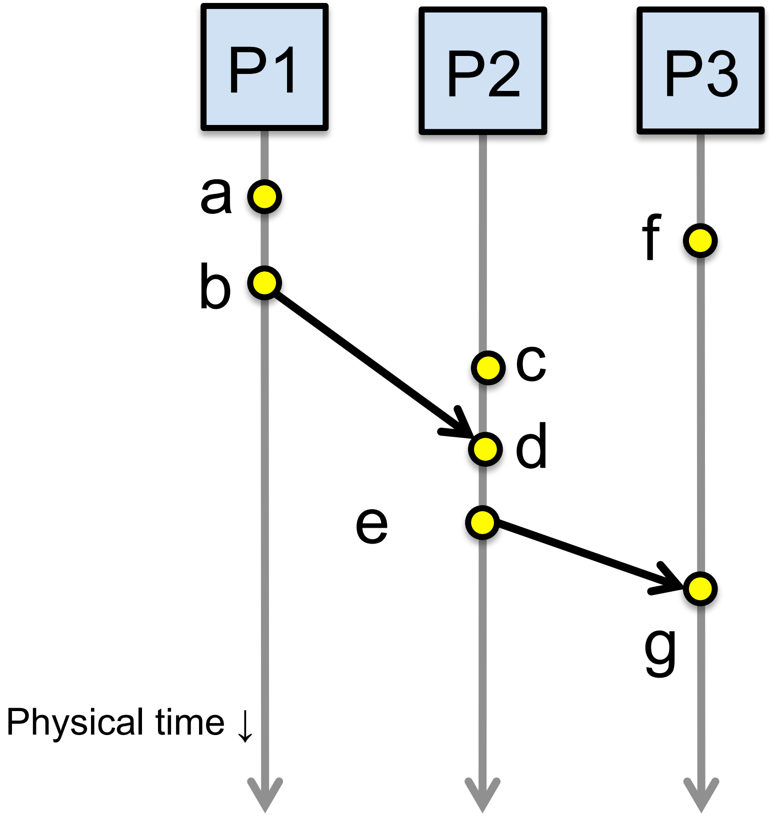
Examples

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



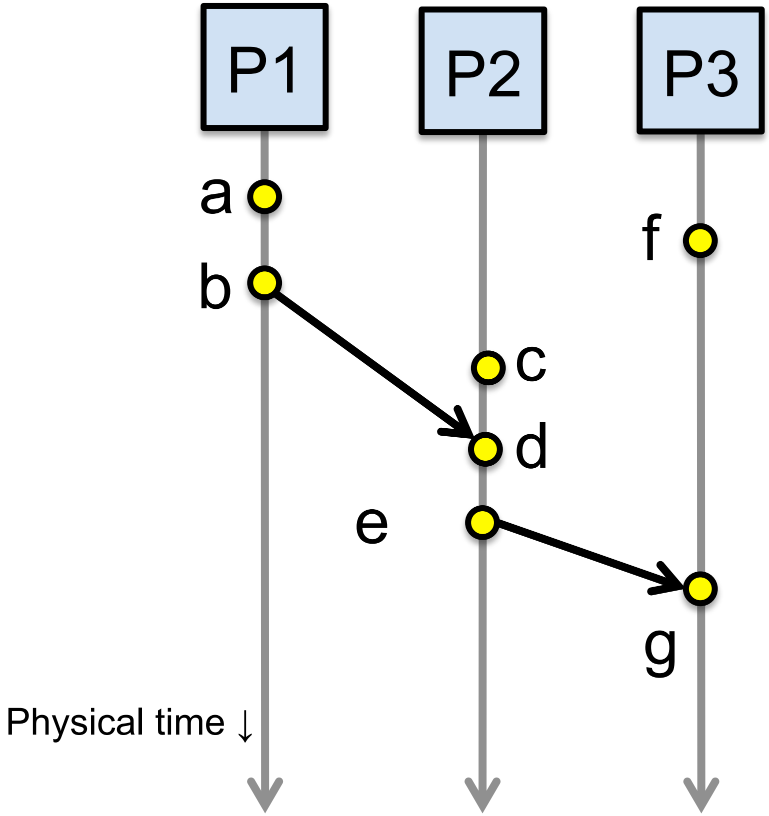
Example

Operations	Concurrent?
a, b	**N**
b, f	
c, f	
e, f	
e, g	
a, c	
a, e	



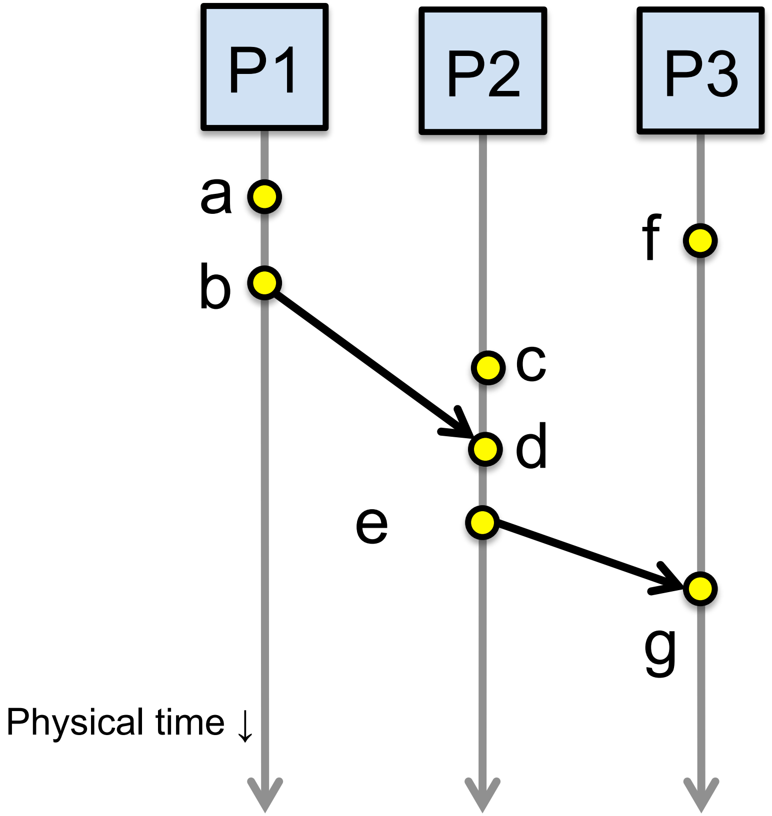
Example

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



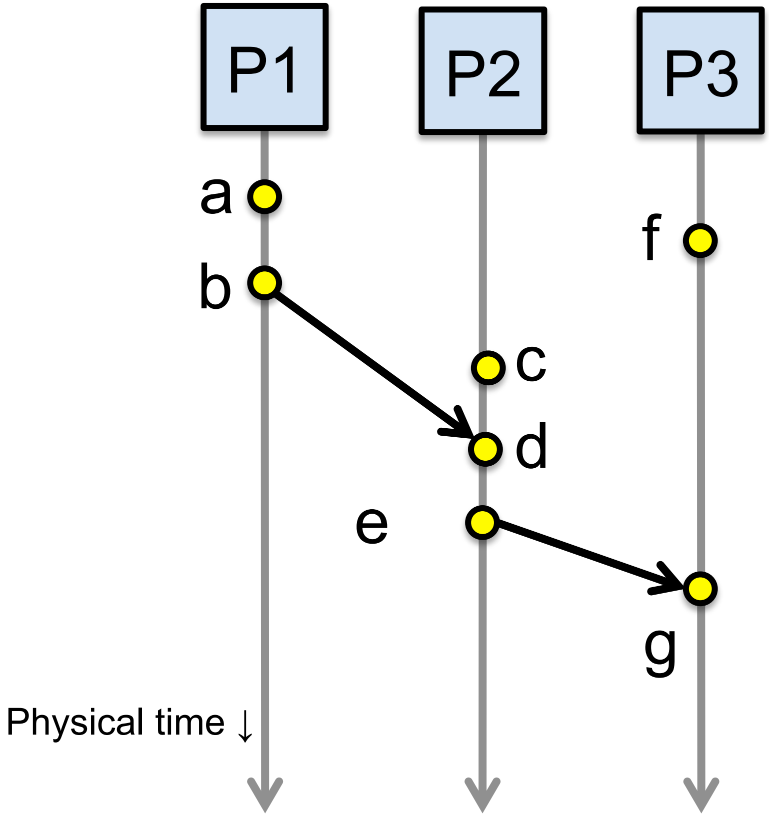
Example

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



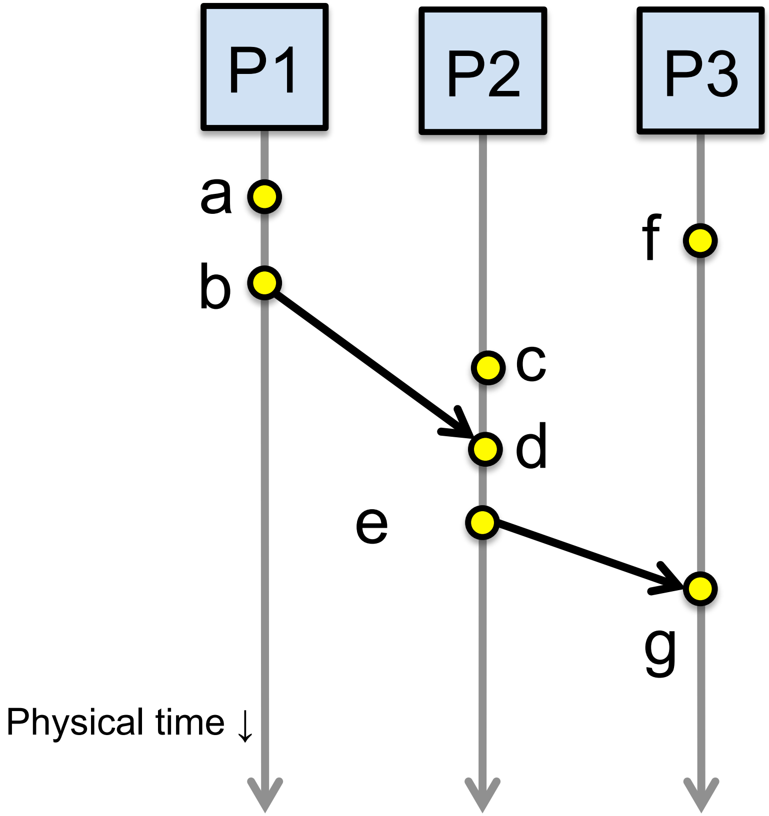
Example

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



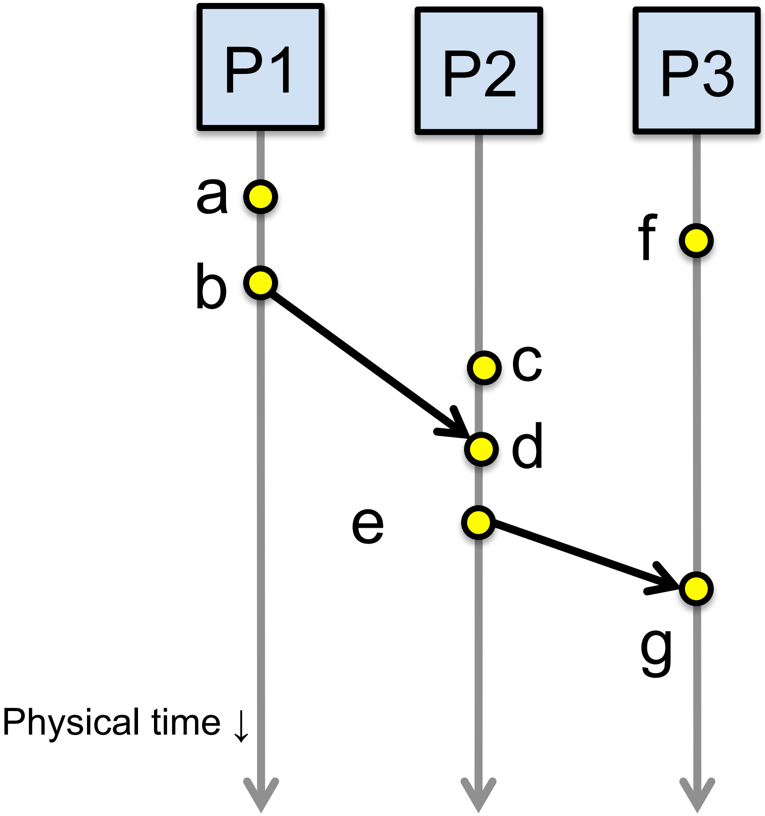
Example

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



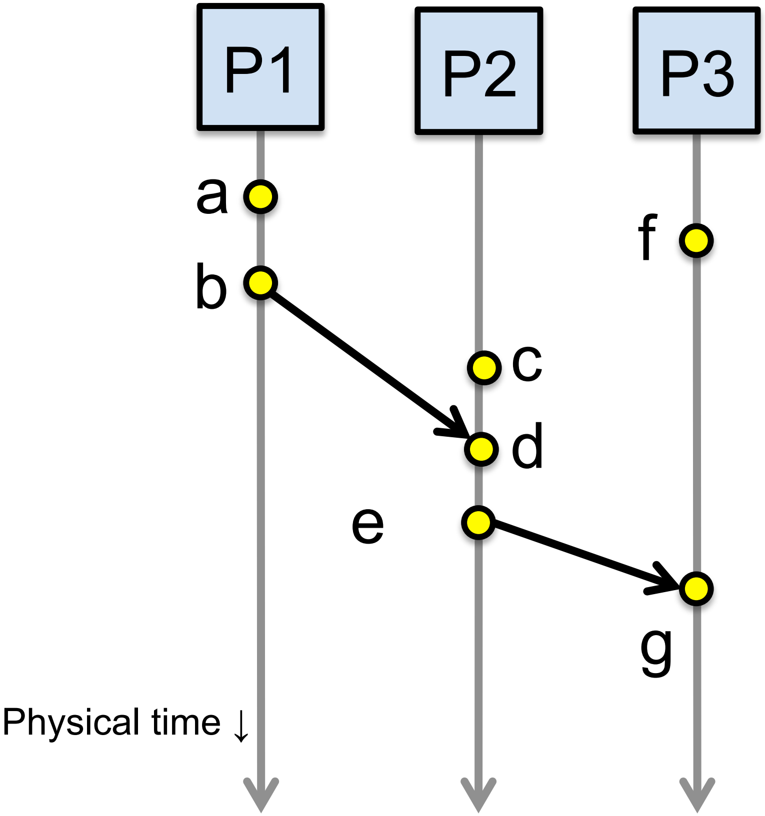
Example

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



Example

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



Is that valid?

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

Is that valid?

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

Valid under causal consistency

Why? *W(x)b* and *W(x)c* are concurrent

So all processes don't (need to) see them in same order

P3 and P4 read the values 'a' and 'b' in order as potentially causally related. No 'causality' for 'c'.

What causal dependencies do we have?

```
Direction of time ---------------->
P1 : w1[x] = 1
P2 : w2[x] = 2
P3 :               r3[x] = 2
P4 :                         r4[x] = 1
```

What causal dependencies do we have?

```
Direction of time ---------------->
P1 : w1[x] = 1
P2 : w2[x] = 2
P3 :                r3[x] = 2
P4 :                          r4[x] = 1
```

Execution causally consistent:

- r3[x] is causally dependent on w2[x]
- r4[x] is causally dependent on w1[x]

Designing a stock market

- What kind of consistency would you use to implement an electronic stock market?

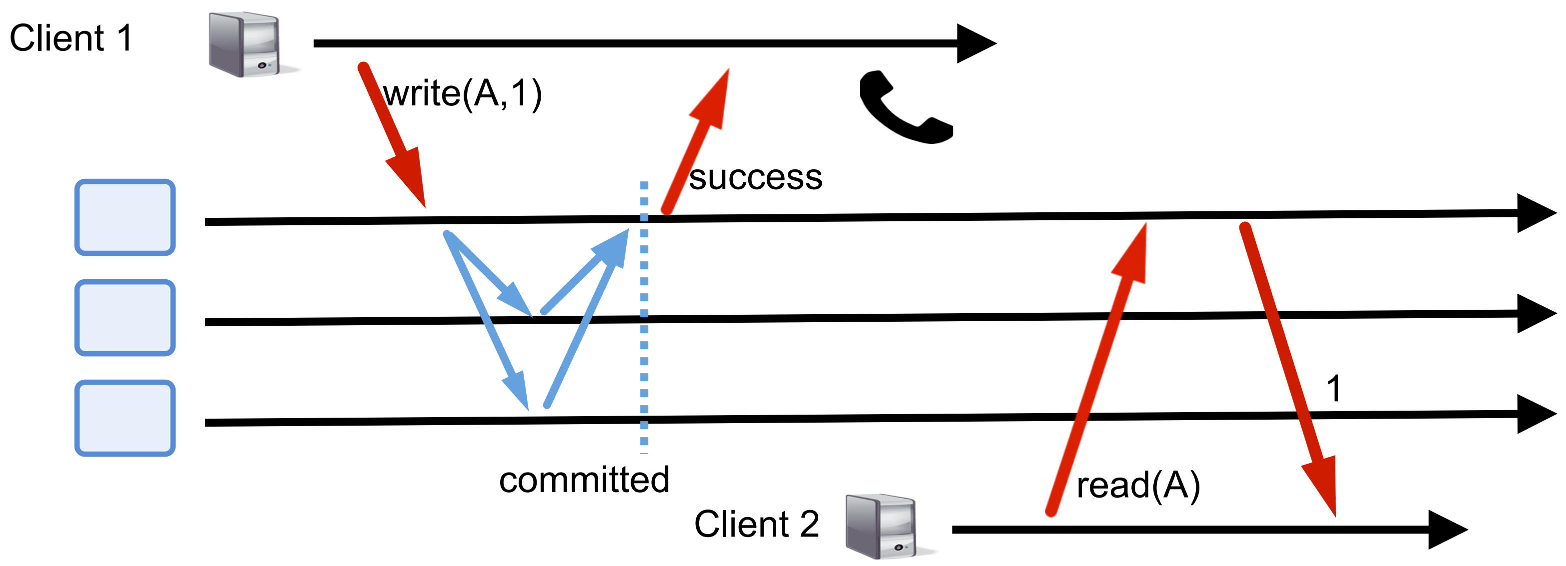
Example: Designing a stock market

- What kind of consistency would you use to implement an electronic stock market?

- Causal Consistency:
  - Reactions to changes in stock values should be consistent.
  - Changes in stocks that are independent can be seen in different orders.

Further examples

Simplifications



- In the following example, simplified read, write and reduced system complexity (no replicas).
- Assignment as write, print as read.
- Intended to demonstrate how linearizability, sequential consistency are violated.
- More realistic view, as above.

Example

Client 1

$X_1 = X_1 + 1$

$Y_1 = Y_1 + 1$

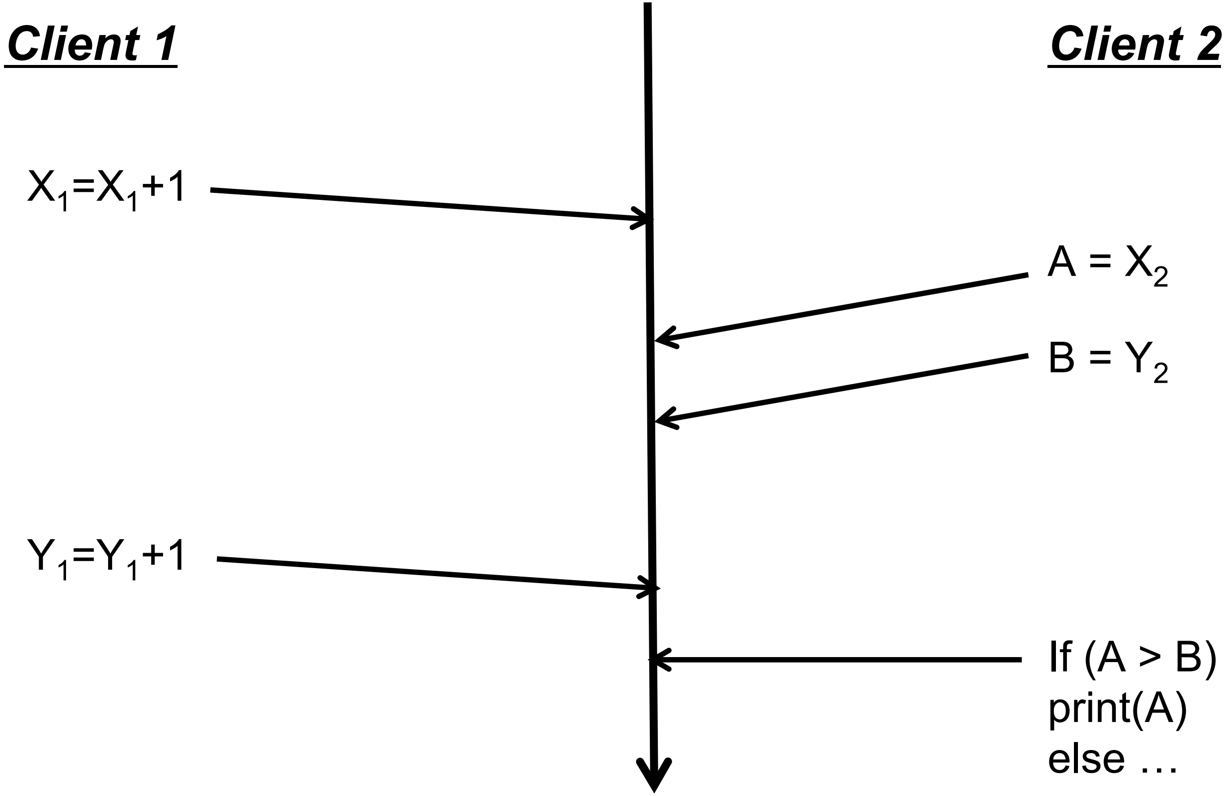
Client 2

$A = X_2$

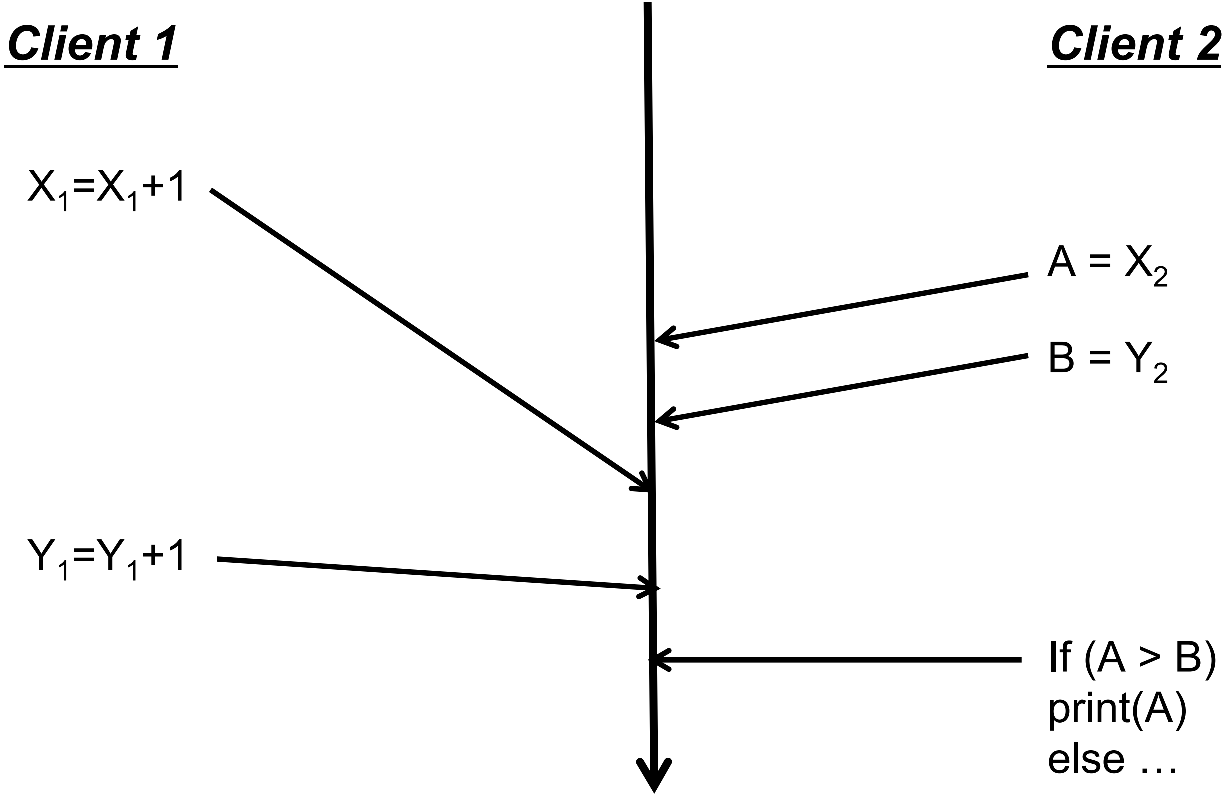
$B = Y_2$

If (A > B)
print(A)
else …

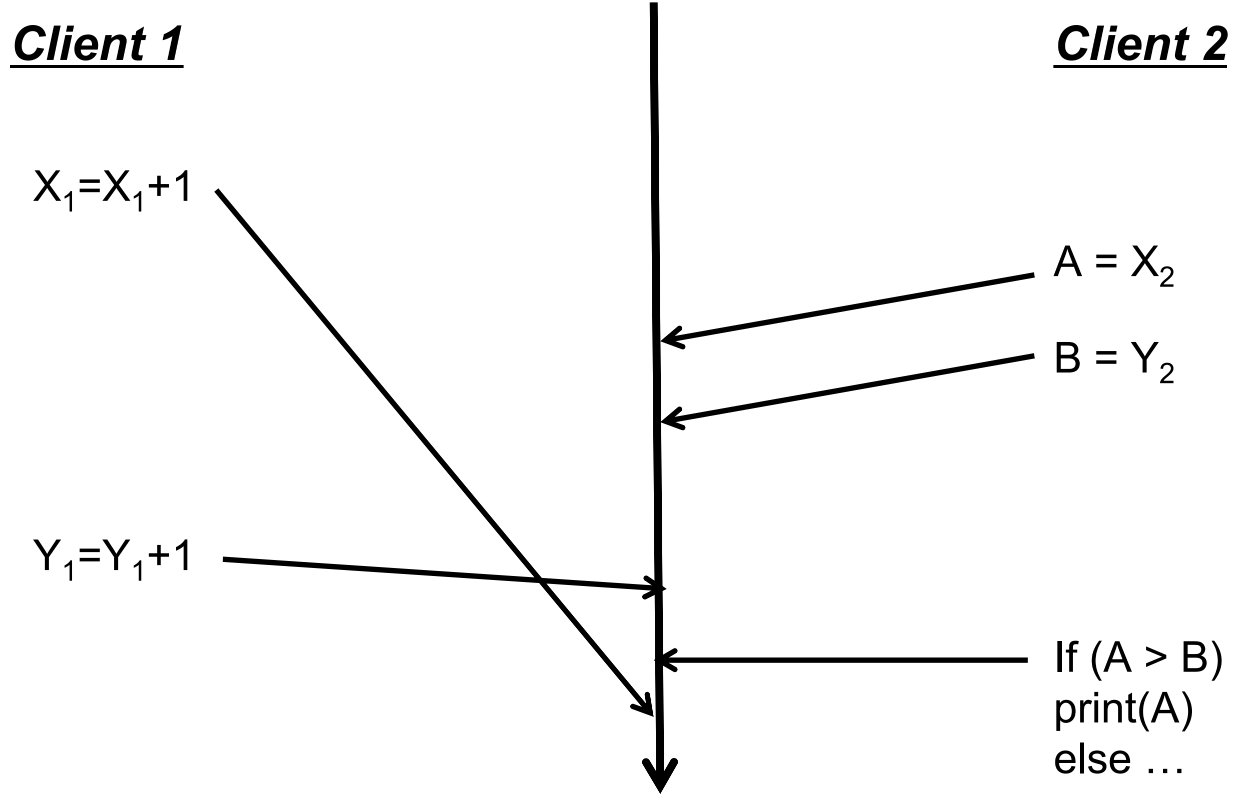
Example: Linearizable



Example: Not Linearizable, sequentially consistent



Example: Not Linearizable nor sequentially consistent



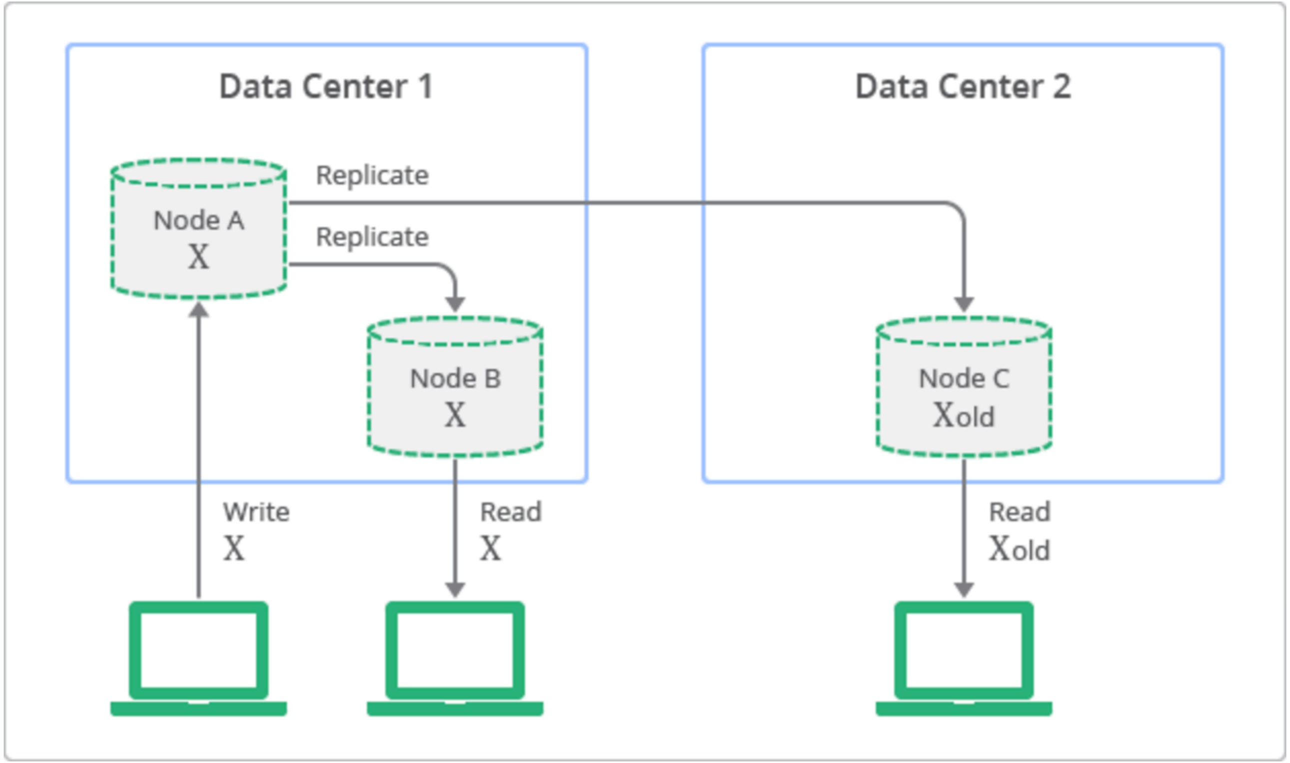
Eventual Consistency

Eventual consistency

- ***Eventual consistency***
  - Writes are *eventually* applied in total order
  - Reads might not see most recent writes in total order

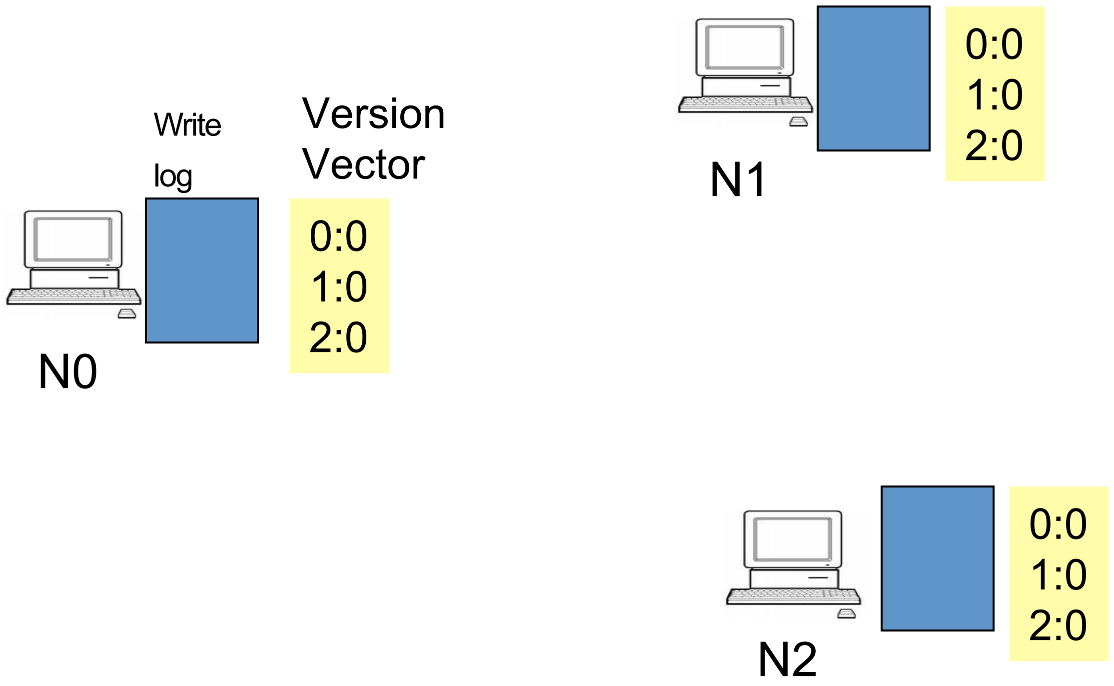
- **Why do people like eventual consistency?**
  - Fast read/write of local copy (no primary, no Paxos)
  - Disconnected operation

Replication with eventual consistency

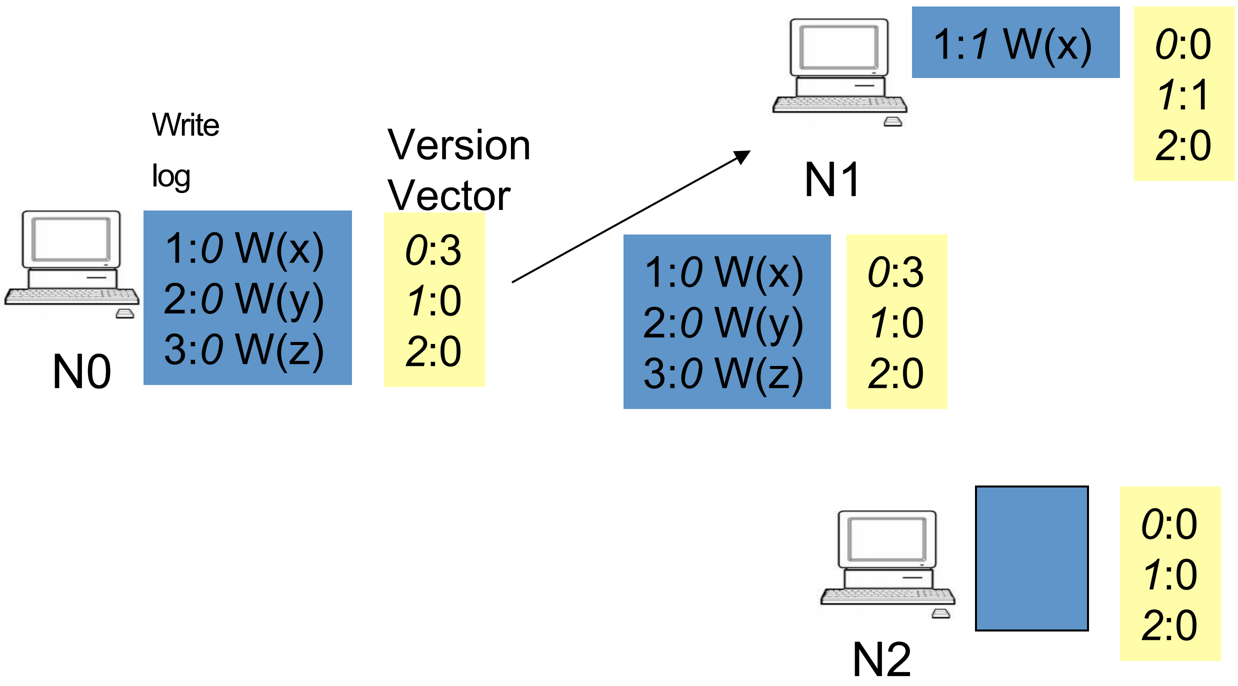


The diagram illustrates that although replicas are always available to read, some replicas may be inconsistent with the latest write on the originating node, at a particular moment in time. In the diagram, Node A is the originating node and nodes B and C are the replicas.

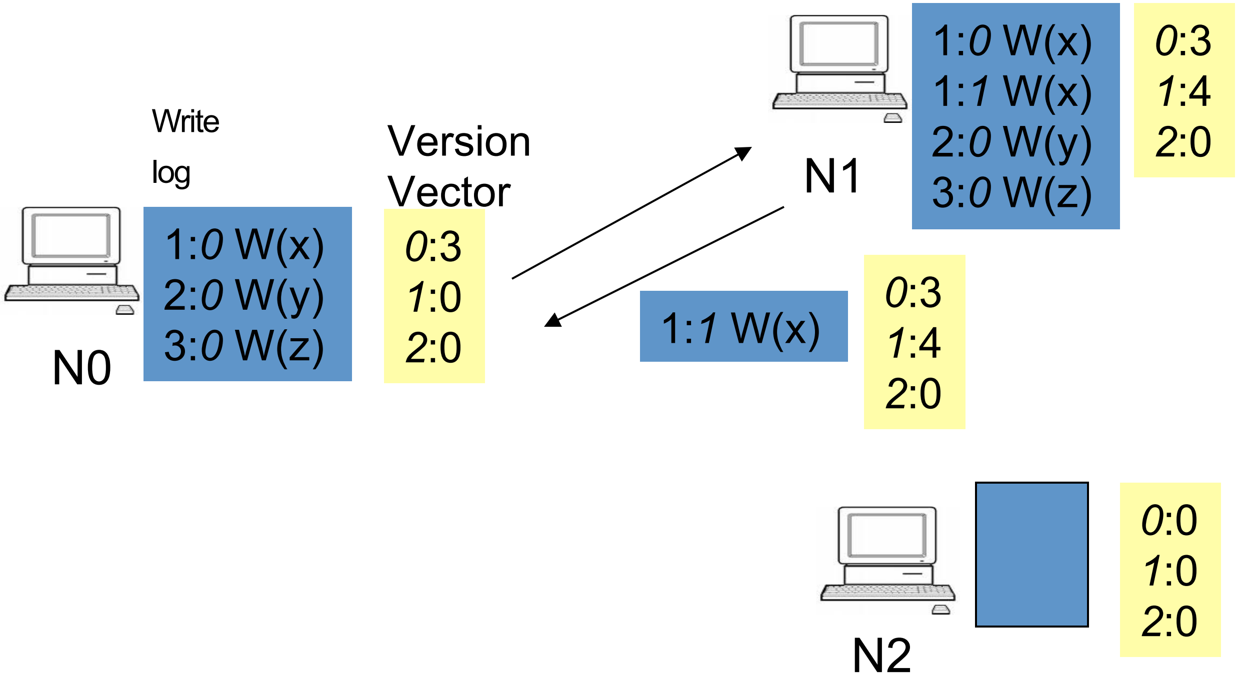
Bayou



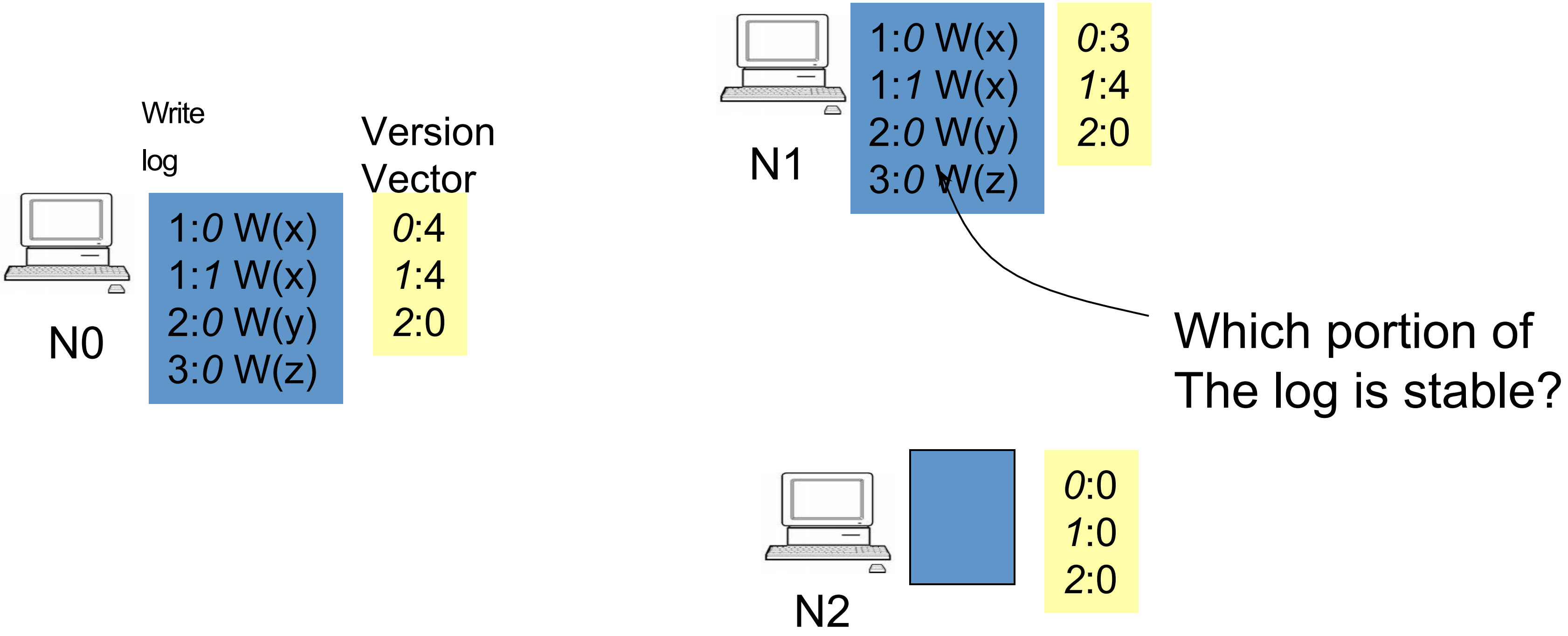
Bayou propagation



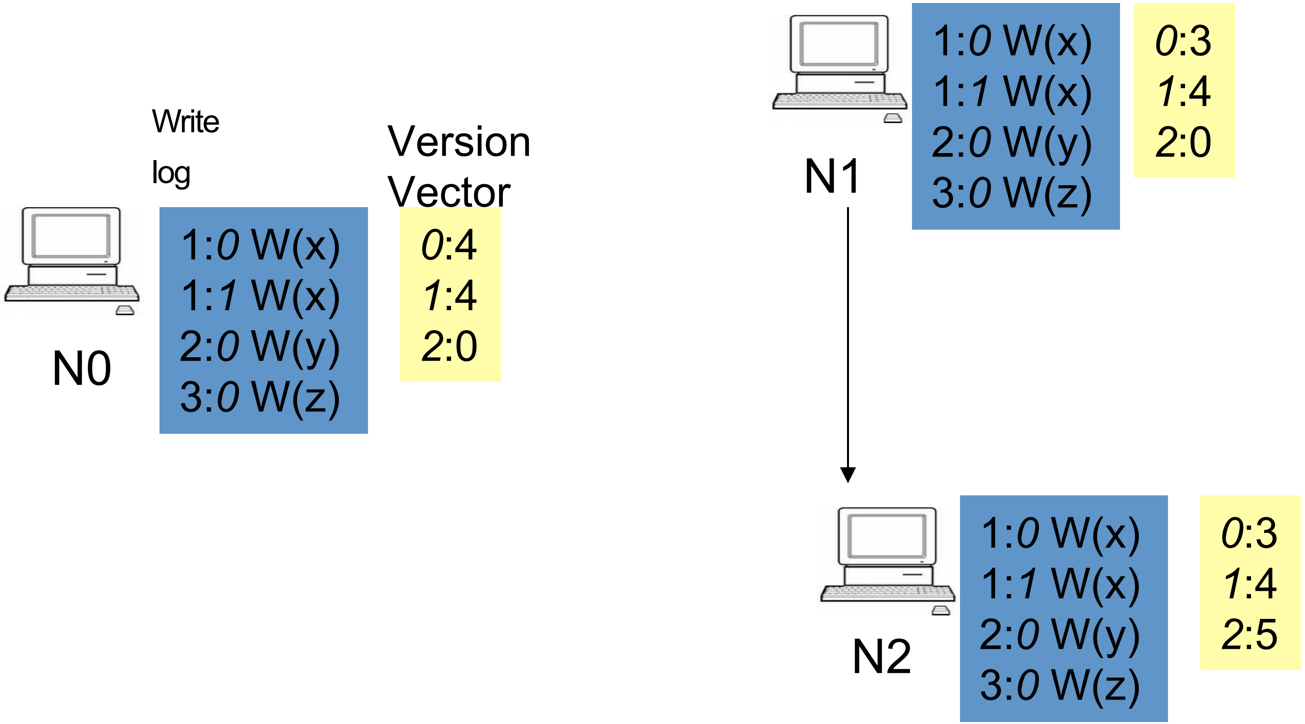
Bayou propagation



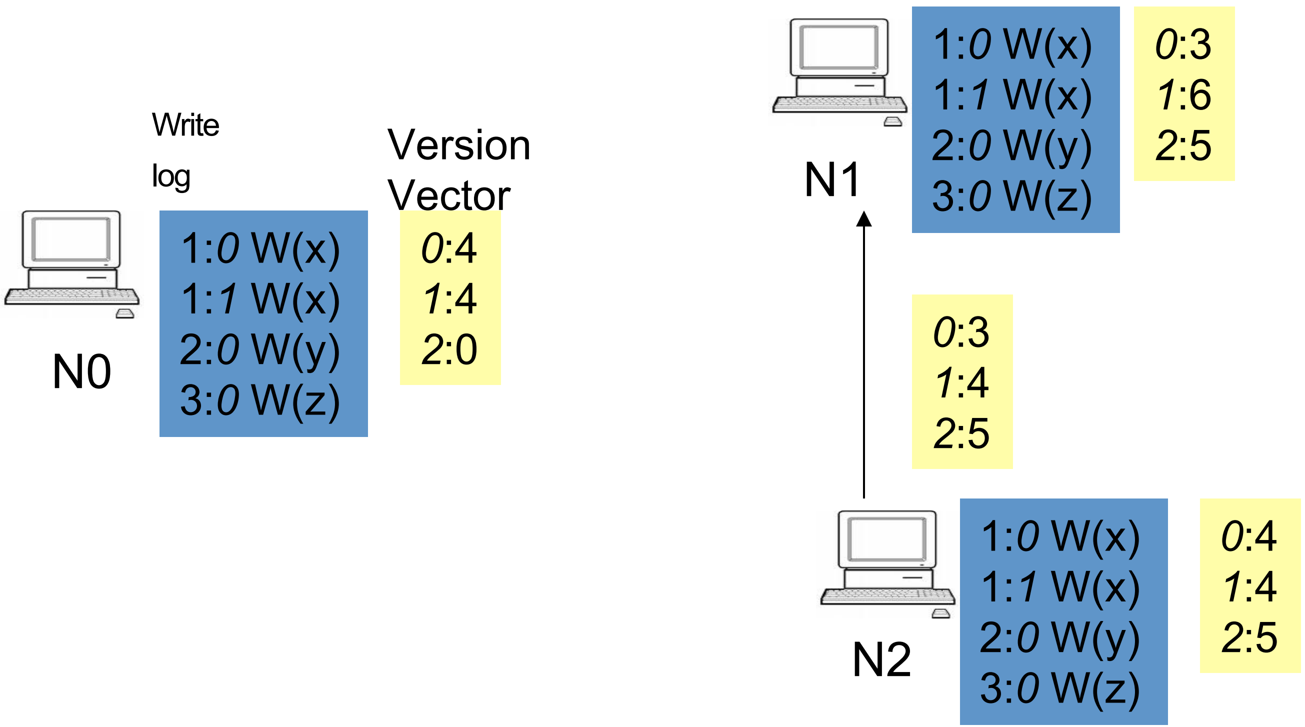
Bayou propagation



Bayou propagation



Bayou propagation



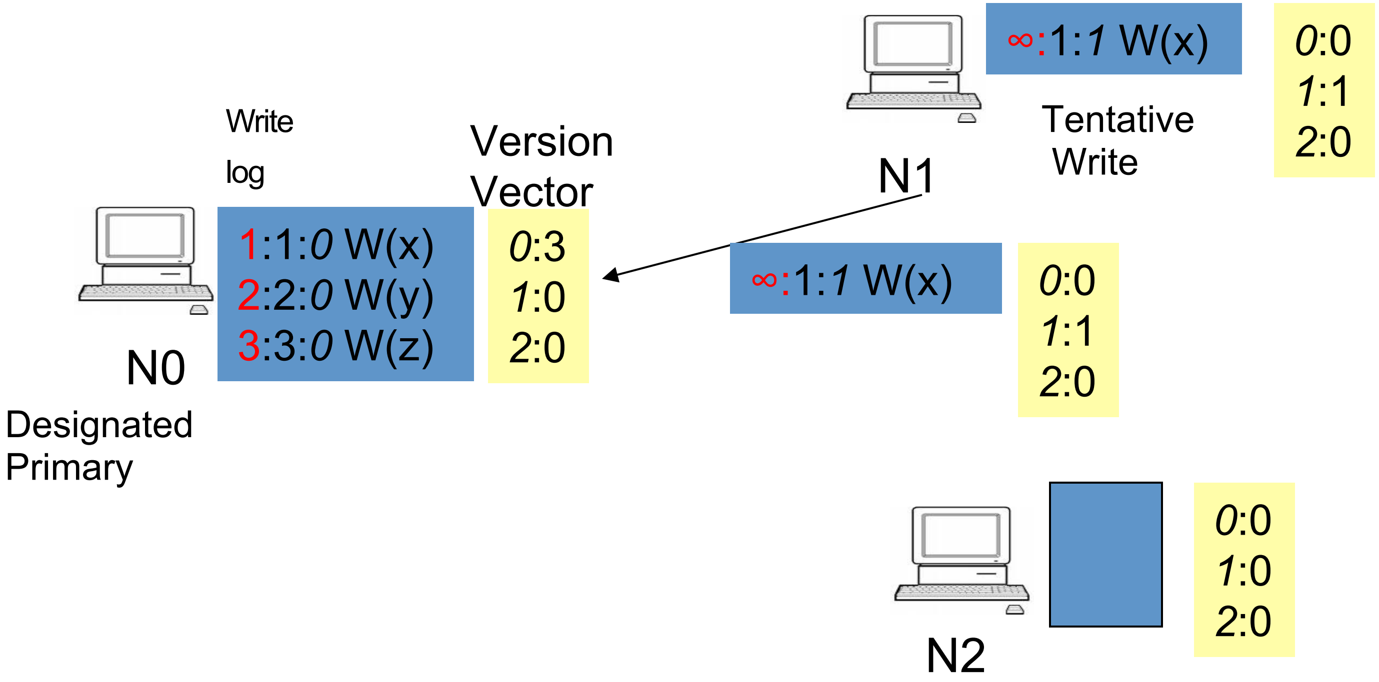
Bayou uses a primary to commit a total order

- Why is it important to make log stable?
  - Stable writes can be committed
  - Stable portion of the log can be truncated

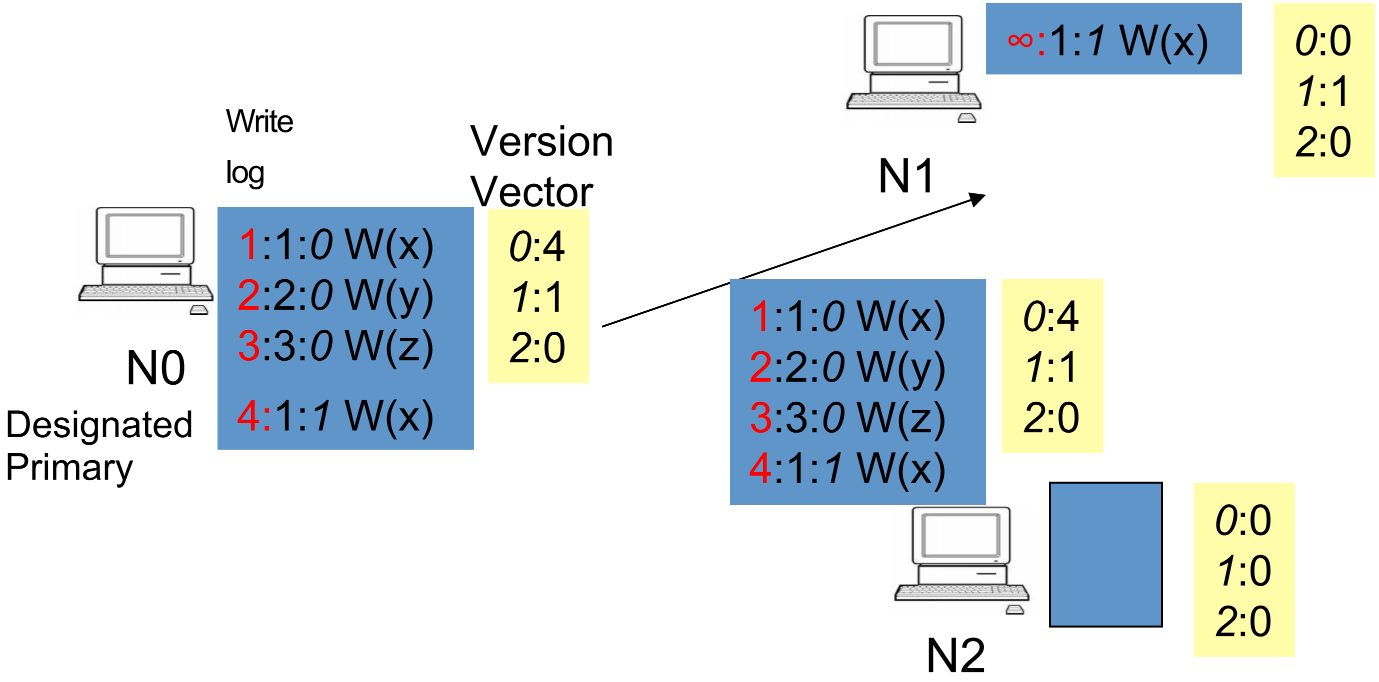
- Problem: If *any* node is offline, the stable portion of all logs stops growing

- Bayou's solution:
  - A designated primary defines a total commit order
  - Primary assigns CSNs (commit-seq-no)
  - Any write with a known CSN is stable
  - All stable writes are ordered before tentative writes

Bayou propagation



Bayou propagation



Assignment 3

Due November 21

Monday’s topic

Concurrency Control, Locking and Recovery