PREM KRISHNA CHETTRI

Distributed System Assignment 3 Submission **Date: 20th Oct '15**

Solution 1>

a> Eventual Consistency provides the fastest model.

b> Sequential and Linearizabe are the most consistent models among the given options.

c> In Linearizabe access are ordered using global timestamp clock where as in sequential consistency, we have to pass through the server in order to determine the sequence of the event. So the later may also cause some server bottleneck.

Solution 2> No, as soon as the updating process receives an acknowledgement that its update is being processed, it may disconnect from the data store and reconnect to another replica. No guarantees are given that the update has already reaches that replica. In contrast with the blocking, the updating process can disconnect only after its update has been fully propagated to other replicas as well. Hence, it does not provide the read your write consistency but improves the performance.

Solution 3 / 4 / 5 from the attachment later

Solution 6>

In order to remove any consistency using quorum, we must ensure that there must be at least one overlapping of replica must exist between read and write. Considering this

a> R + W = N, No we can't. As we can't have two separate replica's quorum and hence, this does not prevent the write-write conflict.

b> W = N, This can prevent the write-write conflict as all the replicas are being considered for updating, so here R =0, if R > 0, we will have an overlapping satisfying the condition enough to prevent write-write conflict.

c> R + W > N/2, No it does not prevent the write-write conflict as there is no way to ensure we have the overlapping replicas, we must also have W > N/2, to ensure only write can achieve write quorum, which is not given to us.

d> R + W > 3 N / 2, No, we cannot say it can prevent the write-write conflict among the replicas, until we are sure that W > N/ 2, even through R+W is greater than 1.5 times the number of replicas.

e> R + W > 3 N / 2, W > 2N/3, yes it can prevent the write-write conflict as there is some overlapping replicas between them.

Solution 7> Only replica (4, 5, 8) will satisfy the query as other replicas hasn't processed atleast one updated seen by front end and resulting timestamp of the front end will become (4, 5, 7).

Solution 8>

a>

T1	T2
W(a, foo)	
R(b)	
	R(b)
	W(b, bar)
R(a)	
	W(a, baz)

They are serially equivalent All conflicting pairs w(a,foo,T1)/w(a,baz,T2), r(b,T1)/w(b,bar,T2) and r(a,T1)/w(a,baz,T2) follows order T1->T2

b>

T1	T2
	R(b)
	W(b, bar)
W(a, foo)	
R(b)	
	W(a, baz)
R(a)	

They are not serially equivalent: Conflicting pairs w(a,foo,T1)/w(a,baz,T2) has order T1->T2, but conflicting pairs r(b,T1)/w(b,bar,T2) and r(a,T1)/w(a,baz,T2) both have order T2->T1

c>

T1	T2
W(a, foo)	
	R(b)
	W(b, bar)
R(b)	
	W(a, baz)
R(a)	

They are not serially equivalent: Conflicting pair w(a,foo,T1)/w(a,baz,T2) has order T1->T2, but conflicting pairs r(b,T1)/w(b,bar,T2) and r(a,T1)/w(a,baz,T2) both have order T2->T1.

d>

T1	T2
	R(b)
W(a, foo)	
R(b)	
R(a)	
	W(b, bar)
	W(a, baz)

They are serially equivalent: All conflicting pairs w(a,foo,T1)/w(a,baz,T2), r(b,T1)/w(b,bar,T2) and r(a,T1)/w(a,baz,T2) follows order T1->T2.

Solution 9>

a> Strict 2P Locking :-

T-1	T-2	T-3
writeLock(x)		
x = 3		
	waitlock(x);	
	Read(x)	
		waitlock(x);
		Read(x)
unlock(x);		
	Read(x) = 3	Read(x) = 3
	writeLock(x);	
	x = 2	
	unlock(x);	
Read(y) = 5		
	writeLock(y);	

	y = 1	
		wait(y);
	unlock(y);	
		Read(y) = 1
	Read(z) = 6	
		Read(z) =6

Final Value would be

W1 (x, 3), R2(x = 3), R3(x = 3), W2(x, 2), R1(y = 5), W2(y, 1), R3(y = 1), R2(z =6), R3(z = 6).

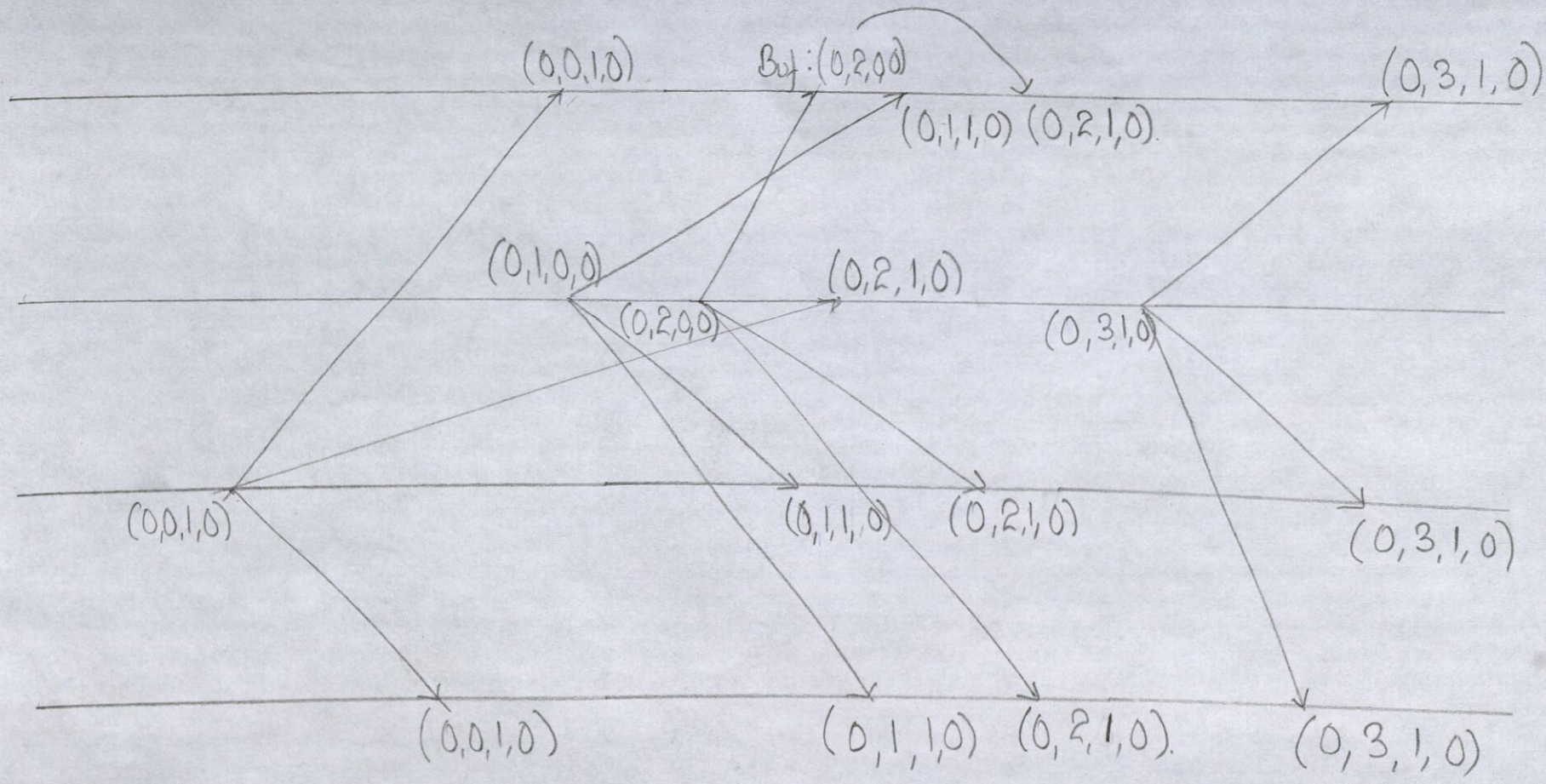
b> TimeStamp Ordering :-

T-1	T-2	T-3
write(X) = 3		
	read(x) = 3	
		read(x) = 3
	write(x) = 2	
read(y) = 5		
	write(y) = 1	
		read(y) =1
	read(z) = 6	
		read(z) = 6

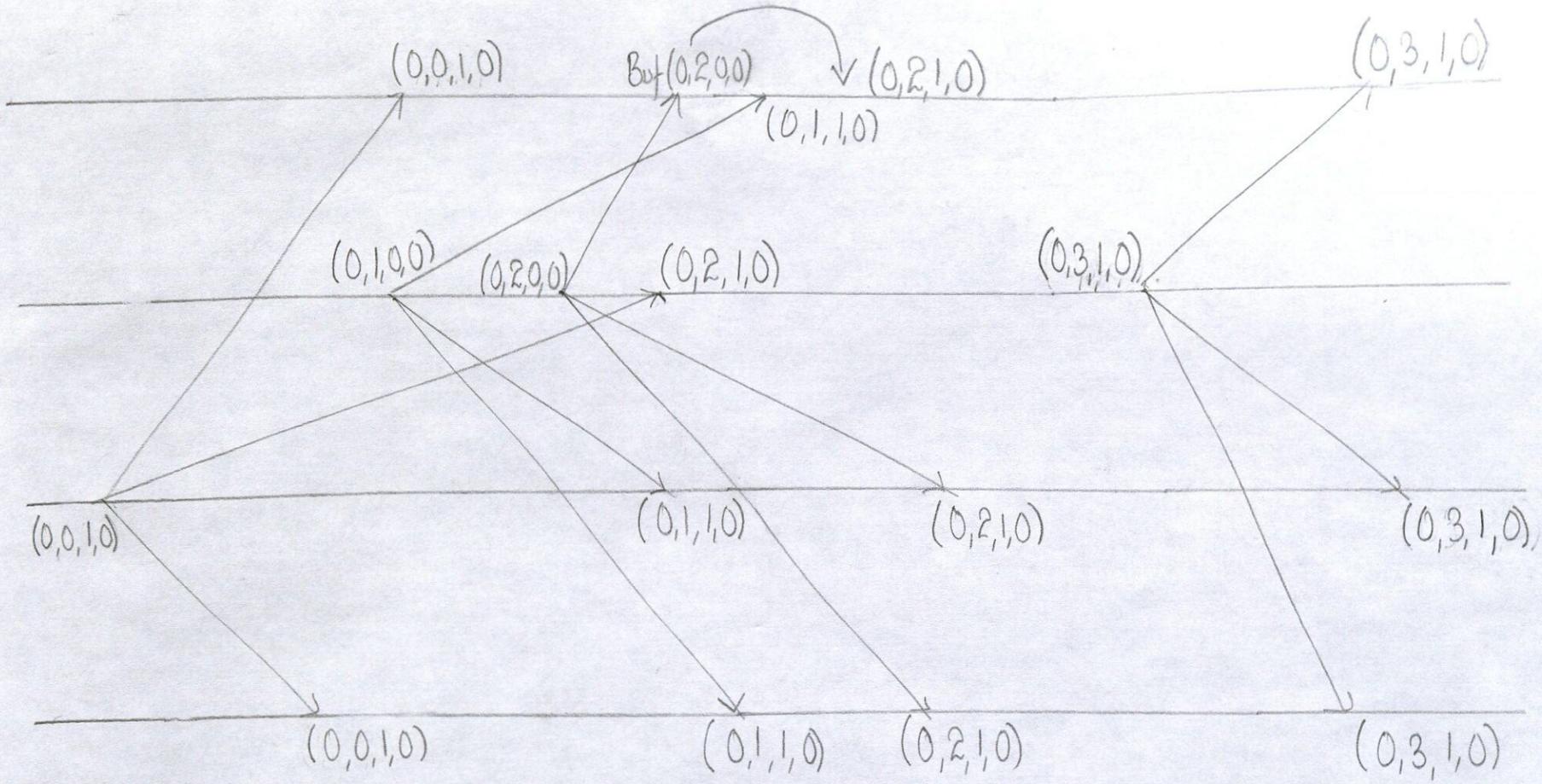
Final Value

W1(x, 3), R2(x), R3(x), W2(x, 2), R1(y), W2(y, 1), R3(y), R2(z), R3(z)

Solution 4.

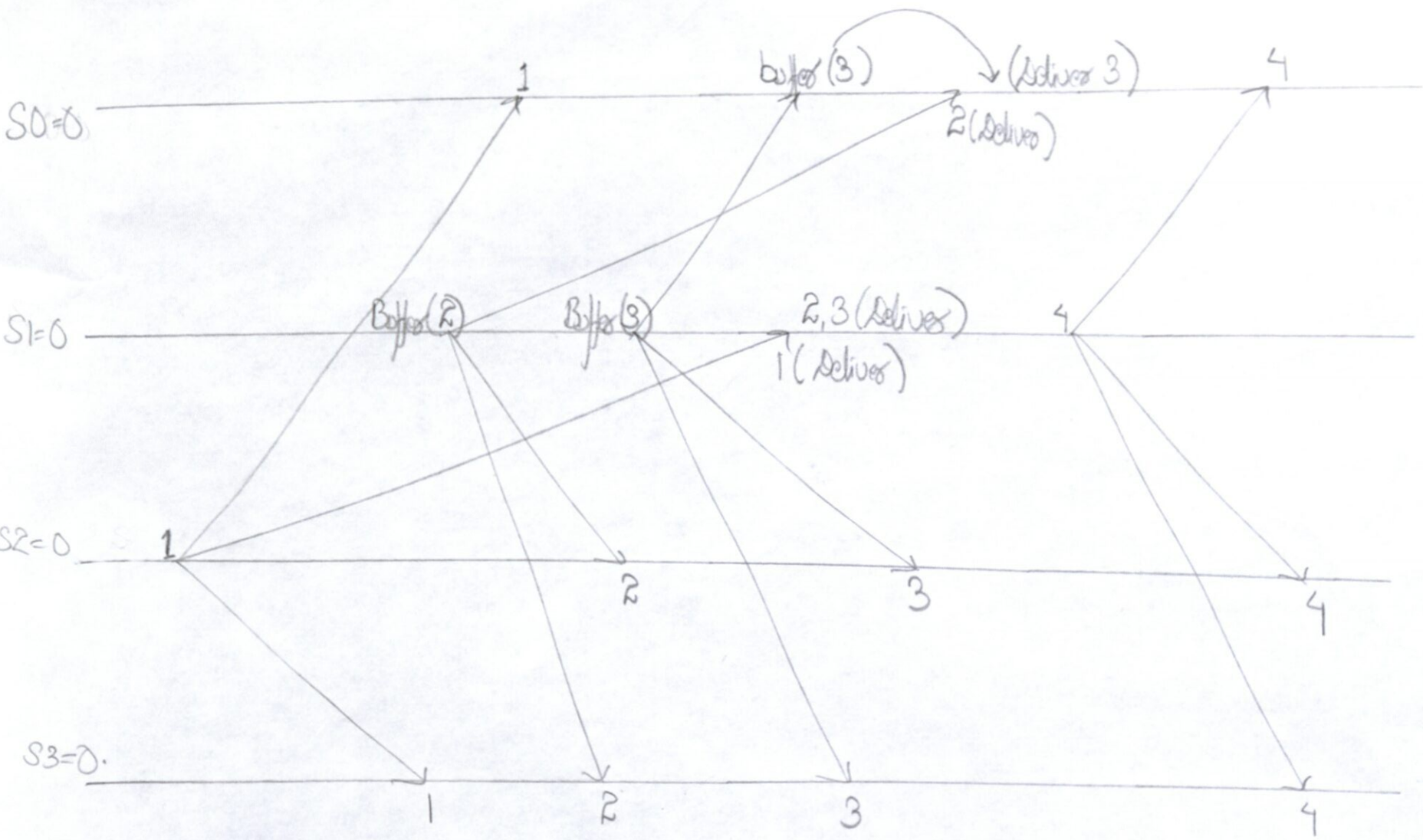

(0,0,1,0)
Buy: (0,2,0,0)
(0,3,1,0)
(0,1,1,0)
(0,2,1,0)
(0,1,0,0)
(0,2,1,0)
(0,2,0,0)
(0,3,1,0)
(0,0,1,0)
(0,1,1,0)
(0,2,1,0)
(0,3,1,0)
(0,0,1,0)
(0,1,1,0)
(0,2,1,0)
(0,3,1,0)

Solution 3.



Solution 4.

Solution 5


S0=0
S1=0
S2=0
S3=0
1
buffer(3)
(deliver 3)
4
2(deliver)
Buffer(2)
Buffer(3)
2,3(deliver)
4
1(deliver)
1
2
3
4
1
2
3
4