Problem Set 4, Part I

Problem 1: Timestamp-based concurrency control

1.1

request	response of the system	any changes to state for A *and/or* explanation of why action wasn't accepted
r2(A)	allowed	RTS = 20
w4(A)	allowed	WTS = 40
r3(A)	denied; rollback	**TS(T3) < WTS(A); 30 < 40**
w1(A)	Denied; rollback	**TS(T1) < RTS(A); 10 < 20**
w2(A)	ignored	TS(T2) >=RTS(A) but TS(T2) < WTS (A)
r5(A)	allowed	RTS = 50

1.2

request	response of the system	any changes to state for A *and/or* explanation of why action wasn't accepted
r2(A)	allowed	RTS = 20
w4(A)	allowed	WTS = 40; c = false
r3(A)	denied; rollback	TS(T3) < WTS(A); 30 < 40; c = false
w1(A)	Denied; rollback	**TS(T1) < RTS(A); 10 < 20; c = false**
w2(A)	Denied; make wait	TS(T2) >=RTS(A) but **TS(T2) < WTS (A) but c = false**
r5(A)	allowed	RTS = 50
c2	allowed	
c4	allowed	c = true, WTS = 40
c5	allowed	
w2(A)	ignored	Already committed

1.3

request	response of the system	any changes to state for A *and/or* explanation of why action wasn't accepted
r2(A)	allowed to read A(0)	RTS(A(0)) = 20, WTS(A(0)) = 0
w4(A)	Allowed to write A(40)	WTS(A(40) = 40, **RTS(A(40))=0**
r3(A)	Allowed to read A(0)	Can read from A(0), A(40), A(0) is valid; **RTS(A(0)) = 30**
w1(A)	denied	No such value of A where **TS > WTS(A)**
w2(A)	Allowed to write A(0))	WTS(A(0)) = 20; [largest **WTS(A) <= TS is 20**, allowed because read time stamp is equal]
r5(A)	Allowed to read A(40)	**RTS(A(40)) = 50**

<u>Problem 2: Replication and distributed concurrency control</u>

2.1) 11 copies of each item, fully distributed locking

voting scheme	***would it work? (yes/no)***	***explanation***
1a) update 6, read 5	no	Although we update a majority of copies, we might need 5 stale ones. r is not > n - w
1b) update 5, read 8	no	**Not updating a majority of the copies**, although the read condition is satisfied
1c) update 8, read 4	yes	**Updating majority** (w > n/2) and also reading > n-w **because 4 > 11-8**
1d) update 3, read 10	no	**Not updating majority of copies**

2.2) 11 copies of each item, primary-copy locking

voting scheme	***would it work? (yes/no)***	***explanation***
1a) update 6, read 5	no	r is not > n-w since 5 == 11 - 6
1b) update 5, read 8	yes	**r > n - w** => 8 > 11-5=6, read majority of copies
1c) update 8, read 4	yes	**r > n - w** => 4 > 11-8 = 3, read majority
1d) update 3, read 10	yes	**r > n - w** => 10 > 11-3 = 8, read majority

2.3)

Using a configuration with more reads can guarantee fault tolerance. **1) (d) would be a good choice** because we have 10 reads out of 11, and 3 updates. In a read-heavy workload, we ensure **not to read any stale data** because **r > n-w** and even if we have a site go down, we have a **very high probability of reading an updated value without incurring latency costs**. Networking would not be too much of a bottleneck either since the data can be read from 10 locations.

2.4)

For a write heavy workload, although it may feel like **1 (c) is a good choice** because we have 8 writes, writes usually incur a higher latency cost. It would seem wise to use 1 (a) because you have a majority of writers which can deal with failed machines and still ensure consistency, and at the same time ensure we **do not incur too high of a writing cost.**