Problem Set 4, Part I

Problem 1: Replication

1-1) 7 copies of each item, fully distributed locking

voting scheme	***would it work? (yes/no)***	***explanation***
1a) update 6 read 3	yes	x = the number of copies that must be locked to acquire a global exclusive lock = 6 > n/2 = 3.5 r = the number of copies that must be locked to acquire a global shared lock = 3 > 7-6 = 1
1b) update 2 read 6	no	x = 2 > 3.5 is an invalid inequality, which may cause the coexistence of more than one global exclusive lock, so it won't work.
1c) update 5 read 2	no	x = 5 > 3.5 r = 2 > 7-5 = 2 is an invalid inequality, which may cause the problem that if one transaction acquires one global exclusive lock, the other transaction can get a global shared lock at the same time, and both of the transactions may work with the wrong versions of the copies.
1d) update 4 read 5	yes	x = 4 > 3.5 r = 5 > 7-4 = 3
1e) update 3 read 4	no	x = 3 > 3.5 is an invalid inequality, which may cause the coexistence of more than one global exclusive lock, so it won't work.

1-2) 7 copies of each item, primary-copy locking

voting scheme	***would it work? (yes/no)***	***explanation***
2a) update 6 read 3	yes	w = 6 r = 3 > 7-6 = 1
2b) update 2 read 6	yes	w = 2 r = 6 > 7-2 = 5
2c) update 5 read 2	no	w = 5 r = 2 > 7-5 = 2 is an invalid inequality. If coincidentally one transaction reads the two copies that have the older version of the data item, given that the other five copies have the latest update, the reader won't be able to know the updated data.
2d) update 4	yes	w = 4

read 5		r = 5 > 7-4 = 3
2e) update 3 read 4	no	w = 3 r = 4 > 7-3 = 4 is an invalid inequality. The reason is the same as 2c: if coincidentally one transaction reads the three copies that have the older version of the data item, given that the other four copies have the latest update, the reader won't be able to know the updated data.

Problem 2: Workload-based configurations

2-1) Since our workload primarily involves reads, the majority of the operations consist of reading data, i.e. more optimization for reads, less for writes; writes are more expensive than reads. Therefore, I believe implementing a system with primary-copy locking, updating 2 and reading 6 copies, is by far the optimal choice. The number of copies for reads is much greater than that for writes. We only need to take care of the potential problem that if the site that stores the primary copy of a data item crashes, operations may be blocked on all items for which it holds the primary copy.

2-2) Since our workload primarily involves writes, the majority of the operations consist of writing data, i.e. more optimization for writes, less for reads; reads are more expensive than writes. Therefore, I believe implementing a system with a fully distributed locking system, updating 6 and reading 3 copies, is by far the optimal choice. The number of copies for writes is much greater than that for reads. Here we don't need to worry about one site as one point of bottleneck but bear in mind the potential risk of hitting deadlocks and overhead of communication over the distributed network of sites.