Online Written Assignment 22 Nisan 2020

Pledge of Honor

- I pledge on my honor that I will not give, recieve or use any unauthorized assistance on the online assignments.
- I pledge to obey rules for taking the online assignments of COMP417/517

Signature: ______

Student ID: 0070881

Name Surname: Erhan Tezcan

1. a) Node-3 buffer Node-3 V.C (2,2,2)

Node-1 3,3,2	Node-1 4,3,2	....

- When node-3 recieves (2,4,2) (from node-2 presumably) it will not deliver it, but instead put it in buffer because the conditions for delivery is not met.
- Then when (2,3,2) is recieved (again from node-2 presumably) it will deliver because both conditions will be met. Then the messages in the buffer will be attempted to be delivered
- The order of delivery is given below:

1st	2nd	3rd	4th
(2,3,2)	(3,3,2)	(4,3,2)	(2,4,2)

1. b) Here I assume a successor does not necessarily mean "immediate" successor, thus I have this figure I.

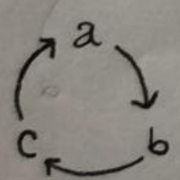

Fig. I.

"a" discovers that its successor is not consistent as "b" changed its predecessor. This would be explained by a new member joining, as seen on figure II.

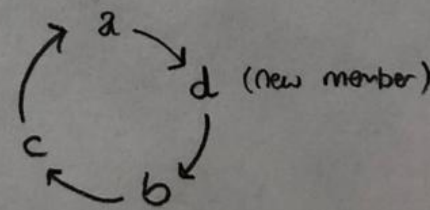

Fig. II

So in short, a new guy "d" must have joined between "a" and "b" where "d" preceeds "b" and succeeds "a". Perhaps "a" was not properly notified about this, therefore it had an inconsistent successor at it's own side.

Erhan Tezcan

2) B: write(i, 55) → B has lock on i.

A: t = read(i) → B released lock i, so it is now in shrink phase

B: write(k, 66) → B seems to acquire lock for k, which would be in growing phase

A: write(k, 44) → A acquires lock on k, so B must've released it.

(where process can only have one lock at a time)

This interleaving should $\boxed{\text{not}}$ occur in two-phase locking. We see that B acquires a lock, releases it and then acquires another lock, then it releases that. That is not two-phase. One possibility of this happening would be when B is allowed to have multiple locks: thus locking both i and k, and then releasing them one by one, so in that case this could occur in a two-phase locking.

3)

P1: W(x)10

P2: R(x)10 W(y)15 W(x)20

P3: R(x)20

P4: R(x)10 R(y)15

P5: R(x)10 R(y)15

This interleaving is causally consistent. However, there is one thing that we should note: if at some point P_3 will do a R(x)10 then that would break causality, but we do not see that here. The figure given as it is above, is causally consistent.

4)

L1: $W_1(D_1)$

L2: $W_1(E_2)$ $W_3(D_3;D_4)$

L3: $W_2(D_1|D_2)$ $W_1(D_1;D_3)$ $W_3(E_2;E_3)$ $W_1(E_3;E_4)$

This data store provides monotonic writes consistency. Other than the obvious ones, the problematic part could be $W_2(D_1|D_2)$, but it is correct since at $W_1(D_1;D_3)$ we see that the previous $W_2(D_1|D_2)$ resulted in D_1 and that was the recent version. It is monotonic write consistent.

Erhan Tezcan

5.a) 20 servers, we would need $N_R + N_W > 20$ and $N_W > 10$.

We are given $N_R = 7$ and $N_W = 12$, which gives:

us $\cancel{19 > 20}$ and $12 > 10$. One of them is wrong

So this is invalid.

5.b) We want 2-fault-tolerant group against Byzantine failures, which would require $2*2+1=5$ members (min.). We have 18 servers, we want minimum write quorum size

A correct quorum configuration would be:

Write quorum: 12, Read Quorum: 9

The reasons being:

- We have at least 5 servers to meet criteria of 2-fault-tolerance.
- If 2 writers fail, we still have $N_W' > 9$ and $N_W' + N_R > 18$ where $N_W' = N_W - 2 = 12 - 2 = 10$, and $N_R = 9$
- If 2 readers fail we still have $N_W > 9$ and $N_W + N_R' > 18$ where $N_R' = N_R - 2 = 9 - 2 = 7$ and $N_W = 12$
- Same conditions also are met if one writer and one reader fails.