ECE428-HW3

Shengjian Chen (3180111657)

QUESTION 1

(a) receive-message

We just need to send to all its lower pid in its group if the node itself is not failed, which means that it will be the new leader. So, should be like.

self.leader = self.pid

for pid in (self.group):

    if pid < self.pid:

    unicast(self.pid, ”Coordinator”)

(b) total messages

the first will be P4 starting election to P7. Then P7 is the highest, so it should be the leader sending ”Coordinator” to P1-P6. So, in total, its 7 messages.

(c) time take

Starting from P4 sends to P7, it takes T. Then, ignoring the processing time, it all takes T from P7 to P1-P6, which is all T’s time but they are in parallel. Then all know the leader should be P7. So, 2T times in total.

(d) P7 fails

Then P4 still sends to P7 but no respond. Then P4 send to P6 and P6 boardcast to P1-P5. So in total it’s 1 + 1 + 5 = 7 messages.

(e) P7 fails, time

Then now the timeout I think it should be 2T, since we need to confirm. So in general it's 2T + T + T = 4T.

(f) best, worst

The best case should be P7 denotes failure then just T to boardcast to all left.

The worst case should be P1 denotes and all one by one failed. Then it's 2T * 6 = 12T.

QUESTION 2

(a) Safety?

NO! it did not, since it might be possible for two nodes have same value in one cycle and the second time it would be recognized as decided.

Along with the forwarding message(a,b), there should also be a set includes all the confirmed node's id. For example, when node 1 send(PROPOSAL, x), then it should be like (PROPOSAL, x, 1). Then in the condition of a==PROPSAL and send(DECIDED, YK). the node's itself should be in the set otherwise the process is adding the node to the set.

(b) majority

```
PROPOSAL = 0
DECIDED = 1

self_ID = # should be defined uniquely for each process
self_Input = # my input
Y_K = 'X' # its a holder now, should wait till confirmed

def Start_consensus():
    if self_Input == 1:
        send(self_ID, PROPOSAL, 1)
    else:
        send(self_ID, PROPOSAL, -1)
```

```
def receive_message(ID, b, holder):
    if Y_K != 'X':
        continue # ignore messages since decided
    if b == DECIDED:
        if holder>0:
            Y_K = 1
        elif holder<0:
            Y_K = 0
        else:
            Y_K = 2
        send(ID, b, holder)
    else:
        # now b is PROPOSAL and still need further confirm
        if ID == self_ID:
            if holder>0:
                Y_K = 1
            elif holder<0:
                Y_K = 0
            else:
                Y_K = 2
            send(ID, DECIDED, holder)
        else:
            if self_Input == 1:
                send(ID, PROPOSAL, holder+1)
            else:
                send(ID, PROPOSAL, holder-1)
```

QUESTION 3

(a). B or R

B-multicast is enough for this algorithm. Because we do not care about the failure in this situation since we have already take the timeout in consideration. So we do not need that reliable. Pair-to-pair B-multicast is enough for this algorithm.

(b). timeout

Timeout should be 2T. Just the forward and backward connection between P_i and P_j.

(c). Decision multicast

This time, R-multicast should be used since we might encounter the sender's failure. But, we do not care about further input since the decision has been made and we just need to broadcast the decision reliably.

(d). time used

I think it requires at least 3T. first T to broadcast to everyone else to gain infos. second T is getting them back. And the third T for decision.

(e). Safety

QUESTION 4

(a). Prepare #1

A1 and A2.

(b). Prepare #2

A1, A2 and A3.

(c). Accept

For P1, it will send at TIME5(3+2). For P2, it will send at TIME9(7+2)

(d). Accept

Yes, it is possible. Since P1 can directly send Accept to A2 at time 5. Then in A2, there will be a time zone for #1. Also another part of the majority(1 over 2) should be A3. (5+1 = 6 which is less then #2 in A3(7))

(B). Prepare #2

A1, A2 and A3.

(Bii). Prepare #2

A2 and A3.

(Biii). Prepare #2

time 13.

(Biv). Prepare #2

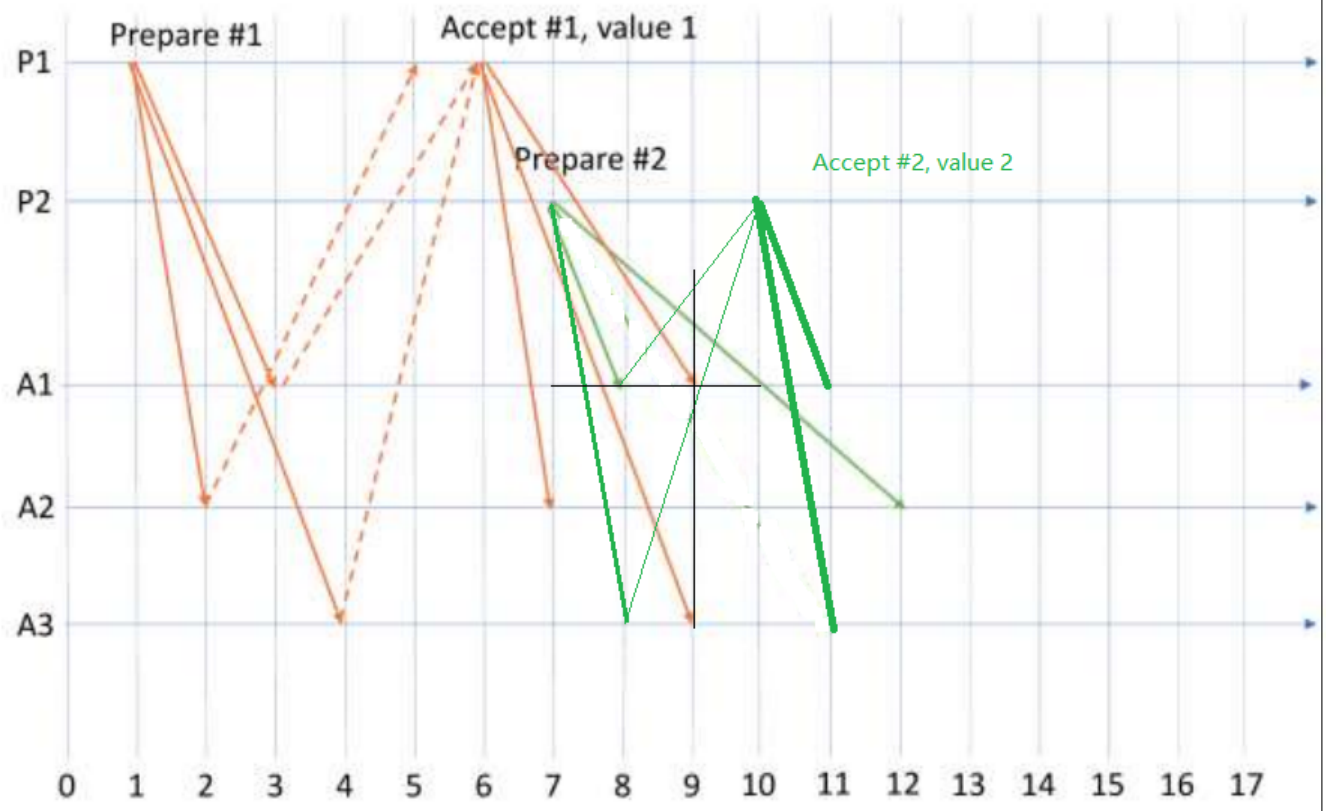
All process would accept. A1, A2 and A3.

(Bv). Prepare #2

Now still value 1 since P2 has not started ACCEPT.

(Bvi). Prepare #2

This time, in A3, Prepare2 arrives before Accept1



QUESTION 5

(a). which one

I think it should be P5.

(b). new leader

P3 received first at 5ms, and vote P3.

P2 and P3 starts election same time, so also vote P2 itself.

for P4, P3's election is faster than P2, so he vote P3.

finally P5, P2 is closer so he vote P2.

(c). Decision multicast

P5 is the first to vote. And P5 is for itself P5.

P4 is 95ms but 65+15 = 80 which is less then 95, also vote P5.

P2 and P3 is same since long time, so both of them vote P5.

(d). time used

I think it should be P2, P3, P5, P1, P4.