Homework 4 - Distributed Systems

Foundations of Blockchain Technology and cryptocurrencies

Fall 2019

Sharif Blockchain Lab
Sharif University of Technology
Department of Electrical Engineering

Deadline: Sunday 1398/9/10 At the beginning of the class

(The Homeworks will be collected at the beginning of the class. 3 minutes after the start of the class the homeworks will not be accepted).

Section 1) Leader Election

1. In the class, we discussed the leader election problem in a synchronous ring. Let's consider asynchronous context. One of the algorithms that solve this problem in asynchronous context is Peterson's algorithm. read about it in the section 15.1.3 of the book "Distributed Algorithms" and answer the following question:

   – Consider the *PetersonLeader* algorithm in a ring with $n = 15$ nodes, in which the UIDs for processes P_1 , . . . , P_{16} are, respectively, 25, 3, 6, 15, 19, 8, 7, 14, 4, 22, 21, 18, 24, 1, 10, 23. Which process is elected as leader?

Section 2) Synchronous Consensus with Link Failure

2. Consider the following variant of the (deterministic) coordinated attack problem. Assume that the network is a complete graph of $n > 2$ participants. The termination and (weak) validity requirements are the same as those in slide 33. However, the agreement requirement is weakened to say: "If any process decides 1, then there are at least two that decide 1." (That is, we want to rule out the case where one general attacks alone, but allow two or more generals to attack together.) Is this problem solvable or unsolvable? Prove.

Section 1) Ben-or Algorithm

3. Consider the following simple randomized Byzantine Agreement protocol. The protocol is fully asynchronous, i.e., different processes are executed at different speeds.
   Assume that there are n processes $\{p_1, p_2, . . ., p_n\}$, where at most t can become faulty. Once a process p_i decides on a value, then it cannot revert the decision. Now the protocol at any process p_i having initial value x_i is as follows. (numbers in sent massages show the step, r is for round number, v is the value and D shows deciding)

Step 0 Set round number r := 0.

Step 1 Send (1, r, x_i) to all processes.

Step 2 Wait until messages of type $(1,\ r,\ *)$ are received from $n-t$ distinct processes. If more than $(n+t)/2$ messages have the same value v, then send $(2, r, v, D)$ to all processes, else send $(2, r, \perp)$ to all processes.

Step 3 Wait until messages of type $(2, r, *)$ are received from $n-t$ distinct processes.

(a) If at least $(n+t)/2$ of these contain v, then decide v.

(b) If at least $t+1$ messages contains v, set $x_i := v$.

(c) Else, $x_i \leftarrow \{0, 1\}$, i.e., the binary value for x_i is set to 0 or 1, each with probability 1/2.

Step 4 Increment r and go to step 1.

Prove if $N > 5t$, For any schedule and any initial values of the processes, the above protocol guarantees, with probability 1, that: all the correct processes will eventually decide on the same value v.

4. In the Figure 1 you see the $global-coin$ version of the $Ben-Or$ algorithm. Instead of tossing independent coins, processes toss a global coin with parameter ρ where $0 < \rho \leq 1/2$; this is a coin such that for each possible outcome $v \in \{0, 1\}$, with probability at least ρ, all the processes that toss the coin get the same outcome v. The strongest global coin is the one with parameter $\rho = 1/2$: all the processes that toss this coin are guaranteed to get the same random bit, i.e., they all get 0 or they all get 1, each case with probability 1/2.
Assume for each round $k \geq 1$ and each $u \in \{0, 1\}$, with probability 1/2, $global-coin(k) = u$ at all processes.

Every process p executes the following:

```
0  procedure consensus(v_p)                          {v_p is the initial value of process p}
     x ← v_p                                         {x is p's current estimate of the decision value}
     k ← 0
     while true do
       k ← k + 1                                     {k is the current round number}
       send (R, k, x) to all processes
       wait for messages of the form (R, k, *) from n − f processes   {"*" can be 0 or 1}
       if received more than n/2 (R, k, v) with the same v
       then send (P, k, v) to all processes
       else send (P, k, ?) to all processes
       wait for messages of the form (P, k, *) from n − f processes   {"*" can be 0, 1 or ?}
       if received at least f + 1 (P, k, v) with the same v ≠ ? then decide v
       if received at least one (P, k, v) with v ≠ ? then x ← v       { toss coin }
       else x ← global-coin(k) {get global coin of round k}
```

Figure 1: $Global-coin$ version of the $Ben-Or$ algorithm

a. Assume $n = 3$ and $f = 1$. show that under a strong adversary, the $global-coin$ variant. of $Ben-Or$s algorithm does not satisfy the liveness property of consensus, namely, termination

with probability 1.
A strong adversary is one that can see the process states and message contents, and schedule the process steps and message receipts accordingly. (Hint: You can consider initial values 0, 1, 1 for the 3 processes respectively. Then as a strong adversary schedule the processes steps and message receipts so that the algorithm never terminates.)
b. Assume $2f + 1 \leq n \leq f$. show that under a strong adversary, the $global - coin$ variant of $Ben - Or$s algorithm does not satisfy the liveness property of consensus, namely, termination with probability 1.

Section 2) PBFT

5. **(Optional)** Please calculate the minimum and maximum message count for each phase in case of f faulty nodes.

6. **(Optional)** The algorithm provides safety if all non-faulty replicas agree on the sequence numbers of requests that commit locally.
a. Show that if $prepared(m, v, n, i)$ is true, $prepared(m', v, n, j)$ is false for any non-faulty replica j and any m' such that $D(m') \neq D(m)$. $D(m)$ is the digest of message m (The same as cryptographic hash).

b. Imagine that there is a decentralized fruit shopping system. Each client sends requests like INSERTION to or SELECTION from her shopping cart to the primary and the PBFT algorithm is executed. (Be careful that the order of operations is important in this system.) Assume that a backup replica has accepted the following requests.

Sequence number 1: INSERT (APPLE) INTO FRUIT
Sequence number 4: INSERT (GRAPE) INTO FRUIT
Sequence number 5: SELECT (PEAR) FROM FRUIT

The replica will be stuck waiting for request with sequence number 2, so it will suspect that the primary is faulty (this happens using timeouts, thus this part depends on the synchrony assumption) and decides to change the primary. The replica that wants to begin a view change sends a ⟨VIEW-CHANGE⟩ message to everyone. Explain what information this ⟨VIEW-CHANGE⟩ message must contain and when will the new primary send the ⟨NEW-VIEW⟩ message to all replicas.

Section 3) Cap Theorem

7. Figures 2 to 4 show different assumptions for a model of a database which stores some information. Imagine that the information needs to be updated. We send the update to the system, and our message happens to go to node 1. Node 1 relays the update to all accessible nodes to update their states. Assume that we later query node 2 for the stored information. Explain what the system does in each case.

Section 4) Paxos

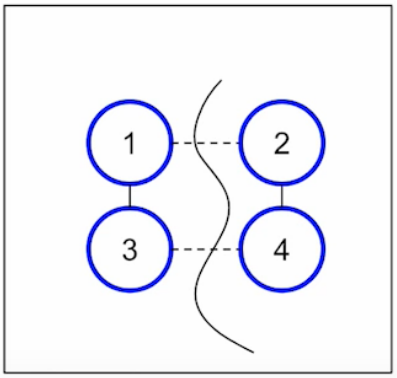

Figure 2: The system is partition tolerant and available

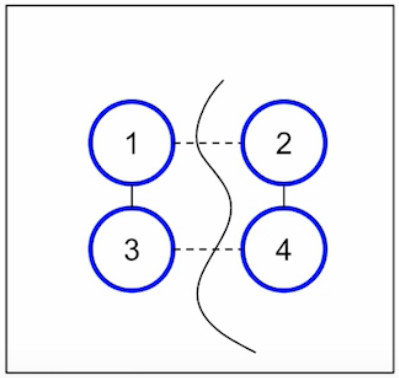

Figure 3: The system is partition tolerant and consistent

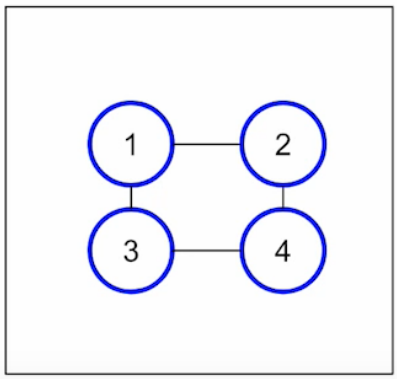

Figure 4: The system is consistent and available

8. Paxos algorithm described in page 143 of lecture slides can solve consensus on a single command from the client. In the figure below you see a modification of paxos which solves consensus on a non-duplicate sequence of commands from the client , "sequence-paxos". C is a command from the client and $\oplus$ is concatenation of non-duplicate commands.

$$\langle C_1, ..., C_m \rangle \oplus C = \begin{cases} \langle C_1, ..., C_m \rangle & \text{if } C \text{ equals some } C_i,\ i \in \{1, ..., m\} \\ \langle C_1, ..., C_m, C \rangle & \text{otherwise} \end{cases} \tag{1}$$

Consider the following properties for simple paxos:

- Validity: only proposed values can be decided.
- Uniform Agreement: no two processes decide different values.
- Integrity: each process can decide at most one value
- Termination: every correct process eventually decides a value.

express the validity, uniform agreement, integrity and temination for the sequence-paxos algorithm.

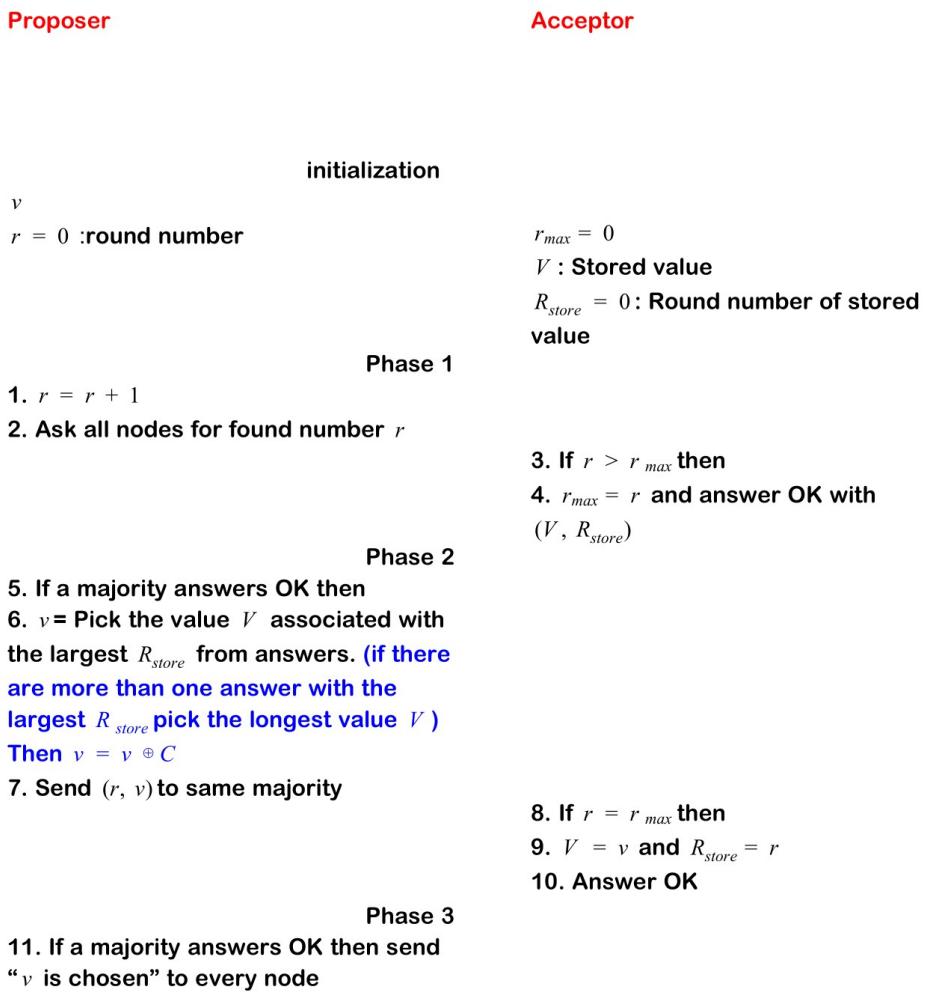

Figure 5: sequence-paxos