Department of Electrical Engineering
Sharif University of Technology

Foundations of Blockchain
by: Dr. Mohammad A. Maddah-Ali
Fall 1399(2020)

Problem Set 2

Issued: Tuesday, Aban 13, 1399 **Due: Wednesday, Aban 21, 1399**

Please email the following address with any questions or concerns:
mahmud.kazemi@gmail.com

Question 1: For the LCR algorithm

- **a)** Give a UID assignment for which $\Omega(n^2)$ messages are sent.
- **b)** Give a UID assignment for which only $O(n)$ messages are sent.
- **c)** Show that the average number of messages sent is $O(nlogn)$, where this average is taken over all possible orderings of the processes on the ring, each assumed to be equally likely.

Question 2: Imagine a synchronous 2D network that consists of $n \times n$ nodes in which any node labelled as $N_{i,j}$ $(0 \leq i, j < n)$ can send messages to nodes labeled as $N_{i,((j+1) \bmod n)}$ and $N_{((i+1) \bmod n),j}$. Find an algorithm to elect a leader in this network. (The number of rounds should be in the order of $\Omega(n)$ or less)

Question 3: 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 validity requirements are the same as those in the Slides of "Consensus with Link Failure" section. 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.

Question 4: Consider the FloodSet algorithm for f failures. Suppose that instead of running for $f + 1$ rounds, the algorithm runs for only f rounds, with the same decision rule. Describe a particular execution in which the correctness requirements are violated.

Question 5: Trace the execution of the EIGStop algorithm for four processes and two failures, where the processes have initial values 1, 0, 0, and 0, respectively. Suppose that processes 1 and 2 are faulty, with process 1 failing in the first round after sending to 2 only, and process 2 failing in the second round after sending to 1 and 3 but not to 4.

Question 6: We proved that it is impossible to reach consensus if $\geq \frac{1}{3}$ of the processors are faulty. Now suppose that each processor has a secret private key and it's signature is verifiable for other processors. With respect to the correctness conditions mentioned for byzantine agreement what can we say about the lower bound on the number of processors required to tolerate failures in this situation? describe an algorithm for solving byzantine agreement in this scenario. Analyze time and message complexity of your algorithm.

Question 7: Consider a Synchronous Consensus with Byzantine Failures problem for n processes with general agreement and termination conditions and a validity condition which states that the decided value must be the input of at least one node. Suppose that each process has an initial value that is an integer in $[0, m-1]$ (m different values). Show that with the presence of f byzantine processes, a consensus is possible if and only if $n \geq f \times max(3, m) + 1$.

Question 8: We have investigated Consensus in a fully connected network with no failure and unlimited bandwidth where each node can send its value to all other nodes. However, in practice, bandwidth limitations are often of great importance as well as failures. Now we consider a network of nodes with unique IDs (e.g. 1 to n) where there is no crash, and every node can only send one message containing one value to one neighbor per round.

- **a)** Develop an algorithm that solves consensus in this scenario. Optimize your algorithm for runtime!
- **b)** What is the runtime of your algorithm?
- **c)** Assume that you not only need to solve consensus, but the more challenging task that every node must learn the input values of all nodes. Show that this problem requires at least $n - 1$ time units!

Question 9: Image that in a distributed system with n processing nodes, at most $\frac{n}{3}$ nodes may fault. Considering this point, the Paxos algorithm is modified such that, in any part of the algorithm $\frac{n}{3} + 1$ valid responses are expected rather than $\frac{n}{2} + 1$ ones. Does this change violates safety and eventual liveness properties? Explain.

Question 10: Answer these questions about the Paxos algorithm. If the expression is true prove it else give a counter example.

- **a)** If we don't have node failure in the Paxos algorithm, we necessarily reach consensus.
- **b)** If in the second phase of the algorithm, the leader does not care about the round number of the received values and accepts the maximum value, algorithm is still safe.

Question 11: 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 $\frac{(n+t)}{2}$ 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 $\frac{(n+t)}{2}$ 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 $\frac{1}{2}$.
- **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.

Question 12: 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 $\rho = \frac{1}{2}$: 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 $\frac{1}{2}$. Assume for each round $k \geq 1$ and each $u \in \{0, 1\}$, with probability $\frac{1}{2}$, $global - coin(k) = u$ at all processes.

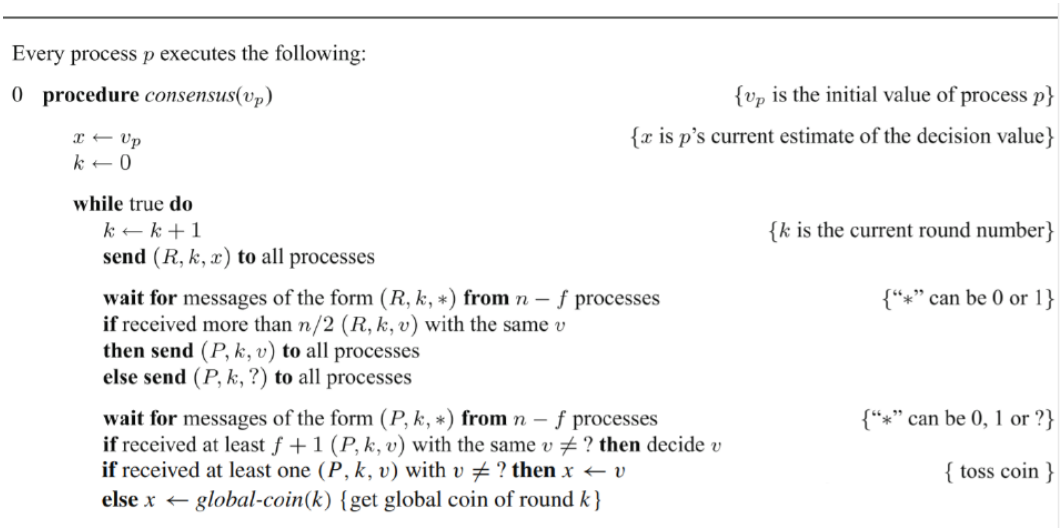

Figure 1: Global-coin version of the Ben-Or algorithm

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

Question 13: **(Cap Theorem)** 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.

Question 14: **(Optional)** For the PBFT algorithm, please calculate the minimum and maximum message count for each phase in case of f faulty nodes.

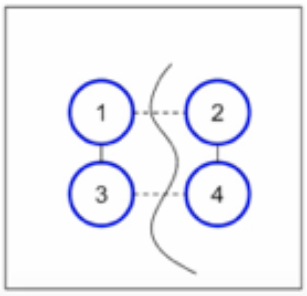

Figure 2: The system is partition tolerant and available

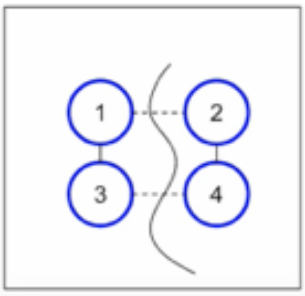

Figure 3: The system is partition tolerant and consistent

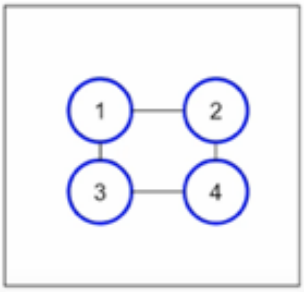

Figure 4: The system is consistent and available