Course: Introduction to Blockchain (CSL7490)
Question Paper Minor

Sep 19, 2024 04:30 PM IST

Instructions:

- The exam has a total of **60 marks**.
- Clearly and concretely list all assumptions if you are making any.
- A direct answer (result) without any accompanying explanation will receive 0 marks.
- Please aim for precision in your answer to the question. Any unnecessary elaboration or attempt to deceive will result in a 50% deduction in marks.
- If anyone is found copying before, during, or after the exam, he/she will be removed from the course and given a FAILED grade.
- You are free to any of the theorem covered in the class. You don't need to explain the theorem details.

****

Q1. State the four assumptions typically considered when discussing consensus protocols. **2 marks**
Ans: 1. Permissioned Network 2. Public Key Infrastructure (PKI) 3. Synchronous Network 4. Fraction of honest nodes and/or malicious node.

Q2. In the PoW blockchain, if the network propagation delay is Δ (excluding queuing delay), the queuing delay at each node is ζ, and the Diameter of the blockchain network is D. The fraction of the total (maximum) time required for a block to reach the destination to the block inter-arrival time (T_{itr}) defines the probability of the block being a part of a fork (the situation where two blocks are at the same height). Assume that each block contains $|B|$ number of transactions. **Given this**, answer the following:

a.) Calculate the effective throughput of the blockchain. **4 marks**
Ans: Without considering propagation delay, queuing delay, and forks, the raw transaction throughput is determined by the block size and block interval: $R_t = \frac{|B|}{T_{itr}}$
The maximum time required for a block to reach the destination is calculated as: $T_{max} = \Delta + \zeta * D$.
So the probability of a block being part of the fork is: $F = \frac{T_{max}}{T_{itr}}$.

Hence, the effective throughput is $R_{eff} = R_t * (1 - F) = \frac{|B|}{T_{itr}} * (1 - \frac{\Delta + \zeta * D}{T_{itr}})$

b.) If the transaction arrival follows the Poisson process, and D, ζ, Δ and the T_{itr} is constant, then comment on the variation in throughput. **3 marks**
Ans: If D, ζ, Δ and the T_{itr} is constant, the effective throughput is at the peak for some duration as it is a Poisson process. while rest of the time is degrades (in worst case to zero). Poisson process indicates the large number of transactions arrives at the node in a small duration and for rest of the time the transaction arrival rate is very low. In other words, if we calculate the transaction inter-arrival time, it follows a exponential distribution.
The above statement is true of mean rate of transaction arrival is greater than the R_{eff}. Formally, let the transaction arrival rate is λ. If $\lambda > |B|$. On the other hands, if $\lambda \leq |B|$, then R_{eff} is proportional to λ

c.) Determine how throughput changes when the block validation and creation time increases from a negligible value to a significant value, say θ. Please write an expression. **3 marks**
Ans: In current blockchain systems, block validation, creation, and consensus (mining) follow a sequential process, where the effective block inter-arrival time is given by $T_{itr}(effective) = T_{itr} + 2 * \theta$. Increasing block validation and creation times from negligible to θ directly impacts the inter-arrival time, reducing R_t and,

consequently, decreasing overall throughput.

Q3. The Byzantine agreement (BA) problem is a one-time consensus problem, similar to Byzantine broadcast (BB) but without a distinguished sender—all nodes have the same role. In BA, each node i has a private input v_i from a known set V of possible values. Unlike BB, where only the sender has a private input, in BA all nodes have private inputs (e.g., transaction orderings in a blockchain). The properties of the BA is defined as follows:

- **Termination -** Every honest node i eventually halts with some output $w_i \in V$.
- **Agreement -** All honest nodes halt with the same output
- **Validity -** If all honest nodes have $v_i = v^*$, then their outputs will be $w_i = v^*$.

Prove that a deterministic consensus protocol exists to solve BA in the synchronous network, even if $f < n$, where f is the number of byzantine nodes and n is the total number of nodes in the network. **5 marks**

Ans: We know that a deterministic consensus protocol exists in synchronous settings to solve the Byzantine Broadcast (BB) problem. Now, we will demonstrate that, by the end of the protocol execution, each honest node will have the same set of information if they execute Byzantine Agreement (BA).

I propose two protocols to solve BA:

First Protocol: Each node sends its private input to the leader in round 0. The leader then runs BB with the combined private inputs received from other nodes. The rest of the protocol proceeds as usual. This protocol requires $f + 2$ rounds: one round to send private inputs to the leader and $f + 1$ rounds to run BB. However, this protocol does not satisfy the validity property if the leader is malicious. To ensure validity, the protocol must continue until an honest leader is selected, which, in the worst case, can take $(n - 1) * (f + 1)$ rounds.

Note: Students who did not mention the final statement will receive 70% of the marks for the question.

Second Protocol: Each node runs BB in parallel with its private input. This protocol satisfies all three properties (agreement, validity, and termination). It terminates after $f+1$ rounds because by the end of round $f+1$, each honest node will have received the same set of information, as every message is signed by at least one honest node. Consequently, each honest node produces the same output, achieving agreement. Moreover, because all honest nodes receive the private inputs of the other honest nodes, they can ensure validity by outputting values based on the information received from the honest nodes. One method for deciding the output is that if a node's output is supported by at least one other honest node, it outputs its private input.

Q4. Consider a distributed system with 4 nodes (N_1, N_2, N_3, N_4) running a consensus protocol, where message delivery experiences non-deterministic delays, i.e. the network is asynchronous. According to the FLP impossibility theorem, consensus **may** not always be achievable in an asynchronous network, even with just one faulty node. However, sometime you will be able to achieve consensus with non-zero probability. Given this, answer the following:

a.) Assume that in each communication round, there is a 40% probability that any message between nodes (source and destination) will be indefinitely delayed. Determine the expected number of rounds needed to achieve consensus, assuming all nodes are non-faulty but experience message delays. Assume that consensus requires each node to receive all the messages. **Hint:** The expected value of a Geometric Distribution is given by E[X] = 1 / p. **5 marks**

Ans: Let's assume that each message has a 40% chance of being delayed indefinitely in each round. Since the system is asynchronous and we are not guaranteed delivery of messages within a fixed time, we need to calculate the expected number of rounds for consensus, considering that messages might be delayed but not indefinitely in every round.

In each round, if there is a 40% chance that messages are delayed indefinitely, there is also a 60% chance that all required messages are delivered for consensus to be reached.

In the first round, the probability of consensus being reached is $(0.6)^4$. If consensus isn't reached in the first round, there's again a $(60\%)^4$ chance of success in the second round, and so on. The expected number of rounds $E(R)$ can be modeled as a geometric distribution with probability $p = (0.6)^4$ (the probability of success per round).

$E(R) = 1/p = 1/(0.6)^4 = 7.72$ rounds

Thus, the expected number of rounds to reach consensus under these conditions is 7.72 rounds.

Because 7.72 is not possible you must have rounded it up or apply ceil function. We will consider both as a valid solution, given the correct explanation. In other words, we will consider 7.72 and/or 8 as a correct answer given a correct explanation

b.) Given the constraints from Q4a, consider a scenario where each node has a 10% failure probability per round. With up to 2 faulty node (i.e. consensus can be achieved with at max 2 faulty node) and considering

the FLP constraints, calculate the expected number of rounds required for the system to reach consensus. **5 marks**

Ans: We want to find the probability for at most 2 failures, which is the sum of probabilities for 0, 1, and 2 failures: Let X be the random variable representing the number of node failure.

$P(X \leq 2) = P(X = 0) + P(X = 1) + P(X = 2)$

$P(X = k) = \binom{N}{k} p^k (1-p)^{N-k}$

$P(X = 0) = \binom{4}{0} (0.10)^0 (0.90)^4 = 1 * 1 * 0.6561 = 0.6561$

$P(X = 1) = \binom{4}{1} (0.10)^1 (0.90)^3 = 4 * 0.10 * 0.729 = 0.29161$

$P(X = 2) = \binom{4}{2} (0.10)^2 (0.90)^2 = 6 * 0.01 * 0.81 = 0.0486$

So, $P(X \leq 2) = 0.6561 + 0.2916 + 0.0486 = 0.9963$.

Considering both node failure probability and message delays, the overall probability of successfully reaching consensus in any round is a combination of the following:

- 99.63 % chance that at max 2 nodes fail.
- 60% chance that messages are delivered without indefinite delay. Because system can tolerate up 2 failures, it will achieve consensus with $n - 2 = 2$ messages.

So, the total probability of successful consensus per round is:
$P(\text{successful round}) = 0.9963 * (0.6)^2 = 0.35866$.

Thus, the expected number of rounds to reach consensus is: $E(R) = 1/p = 1/0.35866 = 2.789 = 3$ rounds.

Q5. Explain the implications in an asynchronous network if the Global Stabilization Time (GST) is known. Justify your answer by explaining why this knowledge affects the network's behavior. **3 marks**

Ans: This question is discussed in the class multiple time by different means. If GST is know then the network is synchronous, where instead of Δ the maximum message deliver time or the legnth of a round is $GST + \Delta$

Q6. Analyze how selecting different CAP trade-offs (such as prioritizing consistency, availability, or partition tolerance) affects system design. Provide examples of real-world systems, such as blockchain systems, and explain how their CAP trade-off choices shape their operational behavior, in terms of liveness and consistency. **5 marks**

Ans: The trade-offs between consistency, availability, and partition tolerance are central to their operation:

1. Bitcoin (CP trade-off):

- Prioritizes consistency and partition tolerance, sacrificing availability during network delays.
- Effect on liveness: Delayed transaction confirmations during network partitions.
- Effect on consistency: Strong consistency is eventually achieved when all nodes agree on the same block.

2. Ethereum (CP trade-off):

- Also prioritizes consistency and partition tolerance, with a similar impact on liveness as Bitcoin.
- Effect on liveness: Network congestion or forks can delay transaction finality.
- Effect on consistency: Consensus protocols ensure eventual consistency across all nodes.

3. Permissioned Blockchains (e.g., Hyperledger, CA systems):

- These systems may prioritize availability and consistency in environments with lower chances of partitioning (trusted networks).
- Effect on liveness: They achieve high availability and fast consensus.
- Effect on consistency: Strong consistency is maintained, but partition tolerance is sacrificed, relying on a controlled network.

Q7. Consider a distributed system with $n = 10$ nodes, where up to $f = 5$ nodes may exhibit Byzantine behavior. The Dolev-Strong Byzantine broadcast protocol is employed, where each message is accompanied by a cryptographic signature that must be validated by multiple nodes to ensure agreement despite the presence of faulty participants.

a.) Assuming that in each round a node signs and broadcasts a message to all other nodes, calculate the total number of signature verifications (across all nodes) required for the entire protocol execution, given that nodes do not verify the sender's signature, as it is assumed to be well-known. **5 marks**

Ans: There are $n = 10$ nodes in the system. In each round, each node sends a message to $n - 1 = 9$ other nodes. In other words, each node will receive message from $n - 1$ other nodes. Each node verifies $r - 1$ signatures to convince them self for a received message, if the message is received in the r^{th} round. This is because sender's signature need not be verified, as mentioned in the question. Furthermore, we knows that the Dolev-Strong protocol runs for $f + 1 = 6$ rounds. Given this the total number of signature verification required are:

$\sum_{r=1}^{f+1=6} n * (n-1) * r = \sum_{r=1}^{6} 10 * 9 * r = 1890$

Some students might have mentioned it is > 1890, is also a correct answer. In this case they should clearly mention that the node won't stop the signature verification process even after they find the message is signed by $r - 1$ distinct nodes.

b.) Signature verification takes 10 milliseconds for 50% ($\lceil n/2 \rceil$) of the nodes and 5 milliseconds for rest of the other 50% ($\lfloor n/2 \rfloor$)across any path in the network. Additionally, network latency for message propagation is 50 milliseconds per hop (per link). Estimate the total time required to complete the protocol, assuming no other delays. **5 marks**

Ans: Here, I am considering the worst-case scenario, which, as I've emphasized multiple times in class, is essential when analyzing protocols. In the worst case, the maximum message latency will be $D50$ ms, where D represents the network diameter and 50 ms is the per-link delay. You may have considered a different network topology, and thus your answer might vary. However, the approach remains the same: $D50$, where D adjusts based on your topology. Since messages are broadcast to all nodes in parallel, we only count this delay once per round.

Additionally, we assume that 50% of the nodes take 10 ms to verify a signature, while the remaining 50% take 5 ms. Therefore, the total time per round can be expressed as:

So total time per round $= [5 * (10 - 1) * r * 10 + 5 * (10 - 1) * r * 5 + D * 50]$.

Hence the overall time to run the protocol for f+1 rounds is $\sum_{r=1}^{6}[5*(10-1)*r*10+5*(10-1)*r*5+D*50]$

In this expression, I assume that each signature verification takes either 10 ms or 5 ms depending on the node.

However, if you assume that the entire signature verification process uniformly takes either 10 ms or 5 ms, the equation simplifies to:

overall time to run the protocol for f+1 $= \sum_{r=1}^{6}[5 * (10 - 1) * 10 + 5 * (10 - 1) * 5 + D * 50] = 6 * [5 * (10 - 1) * 10 + 5 * (10 - 1) * 5 + D * 50]$

Q8. In the naive version of the Dolev-Strong protocol, the set of malicious nodes is fixed: if a node is honest at the start of the protocol, it remains honest throughout, and if a node is malicious (Byzantine) at the start, it remains malicious for the entire run. Now, consider a modified setting where in each round, a different set of f nodes (out of n) can act maliciously. In other words, a node that is honest in one round may act maliciously in the next, and vice versa, although the total number of malicious nodes in each round remains constant. This adversary model is known as a dynamic adversary.

Given this, prove or disprove whether the Dolev-Strong protocol's guarantees (of validity and agreement) still hold under a dynamic adversary. If they do, state the number of rounds (in terms of f) required for the protocol to function correctly in the presence of such an adversary. Support your argument with an example of a small network, such as one consisting of 5 nodes. **10 marks**

Ans: You can approach this question in two ways.

First: In this case, a node cannot become an adversary again in subsequent rounds. In other words, if we take the intersection of the set of adversarial nodes in round r with the consolidated adversaries in other rounds, it should be empty. Formally, if the protocol runs for X rounds and A_r represents the set of adversaries in round r, then $A_r \cap \sum_{i=1, i \neq r}^{X} = \phi$.

Second: This is the opposite scenario, where $A_r \cap \sum_{i=1, i \neq r}^{X} \neq \phi$, meaning that nodes can act as adversaries in multiple rounds. This distinction helps clarify which nodes should be considered honest at the end of the protocol. We will now prove the first case and disprove the second.

Case 1: In this scenario, at some point, we will run out of malicious nodes in a certain round, say the j^{th} round, where the number of remaining nodes $n - f * j < f$. From this point onward, we need an additional $n - f * j + 1$ rounds to establish consensus. Therefore, the total number of rounds required to achieve consensus is $j + (n - f * j + 1)$. Some students might have written $j + f + 1$ rounds, which is also acceptable, but they would receive only 95% of the total marks for the question.

Case 2: In this scenario, you never run out of malicious nodes. Therefore, it becomes unclear which nodes

should be considered honest. In any round during protocol execution, you would require $f+1$ rounds to achieve consensus, and then again another $f+1$ rounds afterward, violating the termination property. Thus, consensus cannot be achieved in this case.

Q9. Explain the importance of PKI assumption in achieving consensus in the synchronous network. **Hint:** Justify your arguments by linking it with the hexagon proof. **5 marks**

Ans: To make the hexagon thought experiment, discssued in the class, compatible with the PKI assumption, we need to include cryptographic keys in the nodes' initialization files. The updated file format is:

(I1) Names, IP addresses, and public keys of two other nodes. (I2) Identification of the sender among the node and the two others. (I3) If the node is the sender, its private input. (I4) A distinct public-private key pair for the node. The thought experiment should be modified accordingly as follows and the the same is detailed in Figure 1

- Node A:

  Neighbors: B and C (name, IP, public key). A is a sender; B and C are non-senders. Private input: 0. Key pair: (pk_A, sk_A).

- Node B:

  Neighbors: A and C' (name, IP, public key). A is a sender; B and C' are non-senders. Key pair: (pk_B, sk_B). Note: C' shares the name and key pair with C but has a different IP.

- Node C:

  Neighbors: A and B' (name, IP, public key). A is a sender; B' and C are non-senders. Key pair: (pk_C, sk_C).

- Node C':

  Neighbors: A' and B (name, IP, public key). A' is a sender; B and C' are non-senders. Key pair: ($pk_{C'}$, $sk_{C'}$).

- Node B':

  Neighbors: A' and C (name, IP, public key). A' is a sender; B' and C are non-senders. Key pair: ($pk_{B'}$, $sk_{B'}$).

- Node A':

  Neighbors: B' and C' (name, IP, public key). A' is a sender; B' and C' are non-senders. Private input: 1. Key pair: ($pk_{A'}$, $sk_{A'}$).

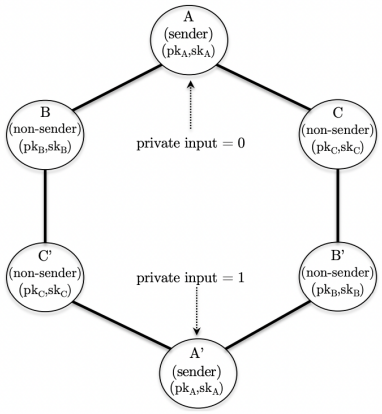

Figure 1: Extension of the hexagon thought experiment to incorporate the PKI assumption.

With the PKI assumption in place, we attempt to replicate the argument from the First world instance of the Hexagon proof discussed above. The proof, which hinges on simulating a hexagon, faces a challenge. Specifically, consider a Byzantine broadcast instance where node X simulates four vertices from the revised hexagon experiment. The goal is to keep honest nodes unaware of their actual environment, applying constraints from the triangle to the hexagon.

However, X faces a limitation: it can only simulate nodes A and A', as it lacks the private keys for nodes B' and C. The Byzantine Broadcast protocol π might require signing every message, but without the private

keys for B' and C, X cannot fully simulate them.

**********All THE BEST**********