# Course: Introduction to Blockchain (CSL7490)
# Question Paper- Major Exam

Nov 20, 2024 06:00 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.** You operate a payment routing node in a payment channel network. There are three channels connected to you, each with varying channel balances, routing fees, and transaction success probabilities.

The details of each channel is as follows:

**Channel-1:**

- Balance: $B_1 = 5\,\text{BTC}$
- Routing fee: $f_1 \sim \text{Uniform}(0.01, 0.03)\,\text{BTC}$
- Success probability: $p_1 = 0.9$

**Channel-2:**

- Balance: $B_2 = 8\,\text{BTC}$
- Routing fee: $f_2 \sim$ normal distribution with mean 0.02 BTC
- Success probability: $p_2 = 0.8$

**Channel-3:**

- Balance: $B_3 = 3\,\text{BTC}$
- Routing fee: $f_3 = 0.025\,\text{BTC}$ (fixed)
- Success probability: $p_3 = 0.95$

Transactions arrive according to a Poisson process with an average arrival rate of $\lambda = 10$ transactions per minute. Note that this is calculated average on all three channels connected to the node. The transaction arrival rate per minute at individual channel would also follows the Poisson distribution with $\lambda = 2$ and $\lambda = 3$ for channel-1 and channel-2, respectively. Given this answer the following.

**a).** Compute the expected earnings for each channel per hour and the total expected earnings for a node per hour. **5 marks**

**Ans:** The expected earning of each channel-i is calculated as $E_{earnings_i} = \lambda_i * p_i * f_i$, where $f_i$ is the routing fee for channel-i.

We know that $\lambda_1 = 2$, and $\lambda_2 = 3$. Because transaction arrival rate per channel also follows a Poisson distribution, $\lambda = \lambda_1 + \lambda_2 + \lambda_3$. So $\lambda_3 = 5$.

Therefore, $E_{earnings_1} = 2 * 0.9 * \frac{0.01+0.03}{2} = 0.036$ BTC per minutes. So $0.036 * 60 = 2.16$ BTC per hour.

Similarly, $E_{earnings_2} = 3 * 0.8 * 0.02 * 60 = 2.88$ BTC per hour. & $E_{earnings_3} = 5 * 0.95 * 0.025 * 60 = 7.125$.

So, $E_{earnings_{overall}} = 2.16 + 2.88 + 7.125 = 12.165$.

**b).** Lets say the channel imbalance factor is calculated as $\frac{|B_{AB} - B_{BA}|}{B_{AB}+B_{BA}}$, where $B_{AB}$ is the balance of the channel from A to B, i.e. $A's$ contribution in the channel. The channel rebalancing is required when the imbalance factor drops below 0.5. The transaction outgoing rate per minute from the channels also follows the Poisson distribution with $\lambda = 4$. Furthermore, each incoming and outgoing transaction to the node transact the value of 1 BTC. Given this, calculate the time required to initiate the rebalancing for channel-1, if the initial imbalance factor of the channel-1 is 0.7. **5 marks**

**Ans:** We are given that imbalance factor is 0.7. Lets say the node you are running is labeled as $A$. So we know $B_{AB} = 5$. Now our task is to calculate $B_{BA}$ using the following equation:

$0.7 = \frac{|5-B_{BA}|}{5+B_{BA}}$

The equation $|5 - B_{BA}|$ has two cases:

**Case 1:** $5 - B_{BA} \geq 0$

If $5 - B_{BA} \geq 0$, then

$$|5 - B_{BA}| = 5 - B_{BA}.$$

Substituting into the equation:

$$3.5 + 0.7B_{BA} = 5 - B_{BA}.$$

Simplify:

$$3.5 + 1.7B_{BA} = 5,$$

$$1.7B_{BA} = 1.5,$$

$$B_{BA} = \frac{1.5}{1.7} \approx 0.882.$$

**Case 2:** $5 - B_{BA} < 0$ If $5 - B_{BA} < 0$, then

$$|5 - B_{BA}| = -(5 - B_{BA}) = B_{BA} - 5.$$

Substituting into the equation:

$$3.5 + 0.7B_{BA} = B_{BA} - 5.$$

Simplify:

$$3.5 + 0.7B_{BA} - B_{BA} = -5,$$

$$3.5 - 0.3B_{BA} = -5,$$

$$-0.3B_{BA} = -8.5,$$

$$B_{BA} = \frac{8.5}{0.3} \approx 28.33.$$

So,

$$B_{BA} \approx 0.882 \quad \text{or} \quad B_{BA} \approx 28.33.$$

For the case where $B_{BA} = 28.33$, the imbalance factor for channel-1 will never drop below 0.7. Instead, it will continue to increase over time, ultimately leading to the channel becoming imbalanced. This is because the outgoing $\lambda = 4$ while the incoming $\lambda = 2$.

On the other hands, in case of $B_{BA} = 0.882$, it will drop below 0.5 even by performing one transaction from A to B, i.e. transferring 1 BTC. which wont take more than 30 sec, as incoming transaction rate per minute is with $\lambda = 2$, while the outgoing transaction rate per minute is with $\lambda = 4$. So the time required to initiate the rebalancing for channel-1 is $\approx 30$ sec.

**c).** Lets say the channel rebalancing incurs an extra cost, for obvious reason. What would be your criteria for routing the transaction on the channel and why? **2 marks**

**Ans:** Prioritize routing through channels with higher expected earnings and lower exhaustion (channel unidirectional) risk.

**Q2.** In a PoS network, validators who double-sign or propose invalid blocks are slashed (penalized) by losing a fraction of their stake. The slashing penalty is $p = 0.2$ for a 20% penalty. Assume that a validator misbehaves with a probability of $q$. Lets say 5 blocks are generated in the system, where each block has reward of 1 coin. The validator's initial stake is 10 coins and the total stake of the system is 100. The slashing penalty $p = 0.2$. The probability of misbehavior $q = 0.05$ (i.e., the validator has a 5% chance of misbehaving). Note that validator can misbehave only when he is selected to propose a block. The validator is banned to take part in the consensus for coming $2^i$ blocks, when he misbehave $i^{th}$ time, where $i$ starts from 1. Given this answer the following:

**a).** What is the expected stake remains with the validator at the end of the experiment **7 marks**?
**Ans:** To determine the expected stake of the validator at the end of the experiment, we will compute two things: 1) Expected Number of Blocks Proposed by the validator, 2) Expected Number of Misbehaviors.
The validator's probability of being selected to propose a block is:
$P_{propose} = \frac{validator'sstake}{totalstake} = \frac{10}{100} = 0.1$
The probability of misbehaving is given as $q = 0.05$, but misbehavior can only occur if the validator is selected to propose. Thus, the joint probability of being selected and misbehaving is:
$P_{misbehave} = P_{propose} * q = 0.1 * 0.05 = 0.005$

Expected Rewards without misbehavior is:
$E_{proposed} = N * P_{propose} = 5 * 0.1 = 0.5$, where N is the number of blocks generated in the systems.
The expected number of misbehaviors by the validator is:
$E_{misbehaviors} = E_{proposed} * q = 0.5 * 0.05 = 0.025$.

The reward for proposing an honest block is 1 coin. If the validator behaves honestly, they earn rewards for their proposals:
$E_{honestrewards} = (E_{proposed} - E_{misbehaviors}) * 1 = (0.5 - 0.025) * 1 = 0.475$ coins
Similarly, $E_{penalty} = E_{misbehaviors} * p = 0.025 * 0.2 = 0.005$.

So, the validator's expected final stake is their initial stake plus rewards from honest blocks minus penalties for misbehavior:
$E_{finalstake} = initialstake + E_{honestrewards} - E_{penalty} = 10 + 0.475 - 0.005 = 10.475 \approx 10.48$

**Please note that, the validator earns rewards only if not banned. A ban occurs after each misbehavior:**

- **First misbehavior: The validator is banned for $2^1 = 2$ blocks.**
- **Second misbehavior: Banned for $2^2 = 4$ blocks.**
- **Additional misbehaviors are unlikely due to the small $q$.**

**The expected number of blocks the validator is not banned:**

$$\mathbb{E}(\textbf{unbanned blocks}) = 5 - \mathbb{E}(\textbf{bans caused by misbehavior}).$$

**The impact of bans depends on the number of misbehaviors, but for small $q$, it's unlikely the validator will be banned for most of the blocks. Thats the reason we are not considering expected number of banned blocks in our answer. However, you should clearly state this in your answer, i.e., the reason for not considering banned blocks.**

**b).** What would be the probability that validator will be selected to propose the next block at the **end** of the experiment **2 marks**.
**Ans:** The validator's probability of being selected to propose a next block at the end experiment is:
$P_{proposeEXEnd} = \frac{validator'sstake}{totalstake} = \frac{10.48}{100} = 0.1048$

**c).** In addition to the above constraints, let $e$ denotes the probabilty that the misbehavior is caught. Given this, calculate the stake remains with the validator after 2 blocks are generated in the system in terms of $e$. **3 marks**
**Ans:** Expected Rewards without misbehavior is:

$E_{proposed} = N * P_{propose} = 2 * 0.1 = 0.2$, where N is the number of blocks generated in the systems.
The expected number of misbehaviors by the validator that are caught is:
$E_{misbehaviors} = E_{proposed} * q * e = 0.2 * 0.05 * e = 0.01 * e$.

The reward for proposing an honest block is 1 coin. If the validator behaves honestly, they earn rewards for their proposals:
$E_{honestrewards} = (E_{proposed} - E_{misbehaviors}) * 1 = (0.2 - 0.01 * e) * 1$ coins
Similarly, $E_{penalty} = E_{misbehaviors} * p = 0.01 * e * 0.2 = 0.002 * e$.

So, the validator's expected final stake is their initial stake plus rewards from honest blocks minus penalties for misbehavior:
$E_{finalstake} = initialstake + E_{honestrewards} - E_{penalty} = 10 + (0.2 - 0.01 * e) - 0.002 * e = 10.2 - 0.012 * e$

**Q3.** Compare and comment on the attack vector of Proof-of-Work (PoW) and Proof-of-Stake (PoS) **2 marks**
**Ans:** PoW relies on computational power, making attacks like a 51% attack or Sybil attack prohibitively expensive due to hardware and energy costs.

PoS relies on token ownership, making attacks like 51% control potentially easier for wealthy entities or through market manipulation.

PoS has fewer attack vectors compared to PoW but requires less physical infrastructure, making some attacks theoretically easier.

**Q4.** Consider a vulnerable smart contract, $C_1$, that allows users to *deposit* and *withdraw* funds. Note that $C_1$ contains variable named *balance* which contains the amount that user added to it. In other words, *balance* represents the consolidated balance of all the users that deposited amount/tokens till particular time instance. Furthermore, it also maintains the balance against particular user, $balance_u$. The *withdraw* function follows these steps:

- Check if the user has sufficient balance, i.e. $balance_u > amount$.
- Transfer the balance to the user's account
- Update the user's balance to zero , i.e. $balance = balance - balance_u$, $balance_u = 0$.

An attacker deploys a malicious contract, $C_2$, to exploit the vulnerability. The malicious contract: 1) Calls the vulnerable contract's ($C_1's$) withdraw() function. and 2) Uses a fallback function to repeatedly call withdraw() before the vulnerable contract updates the attacker's balance. Lets consider the following scenario: 1) The attacker initially deposits 10 ETH into the $C_1$. 2) The contract's total balance is 50 ETH, i.e. $balance = 50$, and 3) The fallback function can execute up to 5 recursive calls before running out of gas of which every 4th fallback function call is failed. Given this, answer the following:

**a).** How much ETH can the attacker drain using this exploit? **2 marks**
**Ans:**

- On the first call, the contract checks the balance (10 ETH) and transfers it to the attacker.
- The fallback function is triggered, initiating another withdrawal call before the balance is updated to zero.
- This repeats for up to 5 recursive calls. However 4th call fails. So the total successful fallback calls are 4.
- The attacker withdraws 10 ETH in each successful recursive call.

So the Total ETH drained: 10 ETH (initial balance) $*4$ (fallback recursive call)= 40 ETH, i.e. the attacker drains the entire contract balance of 40 ETH.

**b).** If a fallback function call requires at least 2300 gas for execution, can the above issue be fixed if contract introduces a gas limit of 2000 units for external calls (call made by user to the smart contract), can the attacker still execute the exploit? Why or why not? **5 marks**
**Ans:** If we set a gas limit of 2000 for external calls, there won't be enough gas remaining to execute a fallback call, which requires at least 2300 gas. Since the attacker cannot trigger fallback calls, they will be unable to carry out the exploit.

Thus, the issue can be mitigated by enforcing a gas limit below 2300 for external calls.

**Q5.** A decentralized exchange (DEX) allows users to swap tokens. The DEX processes transactions in the order they appear in the block. An attacker front-runs a large swap. [Can delay the inclusion of the transaction that performs large swaps and hence delay the execution. At the same time other transactions (pending transactions) are included before this does]. Front-running consists of following steps:

- Observing a pending transaction that will significantly increase the token price.
- Placing a buy order before the observed transaction.
- Selling the token after the price increases.

The attacker starts with 10 ETH. A pending transaction swaps 100 ETH for a token, increasing its price by 50%. The attacker's buy order executes first, followed by the pending transaction. Given this answer the following:

**a).** How much profit can the attacker make if they exchange all their available tokens? How does it impact the execution of Proof-of-Stake consensus. **5 marks**

**Ans:** Let the initial price of the token be $P$. The attacker buys tokens worth 10 ETH:
Tokens Bought$= 10/P$
After the price increases by 50%, the new price is:
$P_{new} = 1.5P$
Selling these tokens at the new price:
Sell Value $=$ Tokens Bought$*P_{new} = 10/P * 1.5P = 15$ ETH
So, the attacker's profit is:
$Profit =$ Sell Value $-$ Initial Investment $= 15 - 10 = 5$ ETH

**b).** What role does gas price bidding play in enabling/disabling front-running attacks? **3 marks**

**Ans:** Gas price bidding is critical to front-running attacks as it determines the order in which transactions are included in the block. Here's how it works:

Higher Gas Price Priority: Miners prioritize transactions offering higher gas fees since they earn more from these transactions.

Front-Running Execution: The attacker observes the pending transaction and submits their buy order with a higher gas fee, ensuring their transaction is included in the block before the victim's transaction.

Economic Incentive: By paying a premium in gas fees, the attacker secures a position to profit from the price change caused by the victim's transaction.

So, gas bidding ensures transaction prioritization, enabling attackers to front-run victims.

**c).** What if the transaction fee for the pending transaction is lower, would it impact front-running. **Explain with the example 5 marks**

**Ans:** If the transaction fee offered by the pending transaction is low, miners are less likely to include it in the block, rendering the front-running attack unsuccessful. For the front-running attack described above to succeed, the attacker must ensure that their transaction offers a higher fee compared to the pending transaction involved in the attack.

To clarify this concept, it is essential to include an example. TAs are instructed to deduct 1 mark if the explanation lacks a supporting example.

**Q6.** A consensus protocol requires electing a leader among $n = 12$ nodes, where: Each node has an equal probability of being elected. Due to partial synchrony, up to $k = 3$ nodes may not receive the leader's proposal within $\Delta$ seconds. Each election round takes $t = 2$ seconds. Given this answer the following:

**a).** What is the probability of an honest leader being elected if $f = 4$ nodes are Byzantine? **3 marks**

**Ans:** We divide this into two cases where $\Delta > 2$ and $\Delta \leq 2$. When $\Delta > 2$, upto 3 honest node may not receive the leader's proposal (as mentioned in the question). So overall malicious/byzantine nodes would be $f_{overall} = 4 + 3 = 7$. This is because nodes not receiving honest leader's proposal may end up supporting byzantine node for the leader election.

Therefore the $P_{honest} = 5/12$, when $\Delta > 2$

On the other hands, $P_{honest} = 8/12 = 2/3$, when $\Delta \leq 2$. This is because although node wont receive leader's proposal for $\Delta$ seconds, they would receive it by the end of round. Note that each round lasts for 2 sec. So ultimately node will contribute in the electing honest leader.

**b).** Calculate the expected time for a successful leader election if the protocol retries in case of failure. **3 marks**

**Ans:** The leader election is said to be successful if the honest leader is selected.
$P_{success} = 2/3$, when $\Delta \leq 2$
Expected number of rounds for success:
$E_{round} = \frac{1}{P_{success}} = 3/2 = 1.5$, can be rounded to 2. both would receive full marks.

On the other hands, in case of $\Delta > 2$, $E_{round} = 12/5 = 2.4$, can be rounded to 2 or 3 depends on the floor or ceil function. However, the answer should be there with proper justification.

**Q7.** A distributed system with $n = 7$ nodes operates under asynchronous communication. The system attempts to reach consensus using a deterministic algorithm, but one node may crash during execution. Given this answer the following:
**a).** Calculate the number of possible configurations of the system's state after one round of communication, assuming each node can independently be either faulty or non-faulty. **2 marks**
**Ans:** Because one node crashes during the execution, it would produce one one output. However, on the other hands, remaining nodes would produce two outputs as they can be either faulty or non-faulty independently. So the total number of configuration $2^{n-1} * 1 = 2^6 * 1 = 64$

Because question mentioned one node MAY crash. So some students might have considered two possible output of that one node, i.e., the one when is crashed and one when the node is up. Note that output when the node is crashed is same as the one when node is faulty. In this case answer would be $2^7 = 128$.

Lastly, some students might have considered faulty node output and crashed node out put is different, then for that node there are 3 different output. First, when the node is honest, second, when the node is faulty, and third, when node is honest. In this case the answer would be $2^{n-1} * 3 = 2^6 * 3 = 192$

@TA please give full marks for all the cases, provided the assumption is clearly stated.

**b).** If a single crash occurs at a random point during the consensus process, what is the probability that the system reaches a consistent state within two rounds? **2 marks**
**Ans:** TBD
**Q8).** Explain there exist an initial ambiguous state configuration in the asynchronous network, while achieving consensus **2 marks**
**Ans:** This is lemma 1 of FLP impossibility proof. Please refer notes for this.

**Q9).** Comment on the Blockchain-as-a-Service based on the three verticals that we have discussed in the class. **2 marks** **Ans:** 1. Service-Based Applications: Blockchain-as-a-Service (BaaS) enables deploying and managing smart contracts to deliver specific application-level services.

2. Node as a Service: BaaS offers full blockchain nodes to users, commonly used for collecting blockchain data for analytics and monitoring.

3. Full Blockchain Network: BaaS provides an entire blockchain network environment, allowing researchers to experiment with and test modifications at the consensus protocol level.

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