SESSION 10

Consensus Algorithms – 1

Learning Objectives

- Discuss consensus mechanism in detail
- Describe the Byzantine Generals' Problem

SESSION 10

Introduction

The blockchain lacks a centralized authority, hence it requires a consensus mechanism. The consensus mechanism is a process of reaching an agreement among the nodes on a final state of data. It is quite easy to reach an agreement between two nodes such as client-server architecture, but in a decentralized network where multiple nodes participate, it is quite difficult to reach an unanimous agreement. Achieving consensus among multiple nodes in a decentralized network is referred to as distributed consensus.

In a distributed network, all the nodes possess the same authority. The presence of faulty/malicious nodes in the network prevents it from reaching a consensus. This situation is referred to as the Byzantine Generals' Problem. This problem is resolved by the adoption of Byzantine Fault Tolerant mechanisms which are responsible for sustaining the decentralized architecture of a blockchain network.

The Practical Byzantine Fault Tolerance consensus algorithm also offers a solution to Byzantine Generals' Problem in a distributed network for reaching a common decision.

In this session, you will learn about the consensus mechanism along with its key properties. In addition, you will also be made conversant with the Byzantine Generals' Problem. Towards the end of the session, you will be acquainted with Byzantine Fault Tolerance and Practical Byzantine Fault Tolerance.

Introduction to Consensus Mechanism

[LO - Discuss consensus mechanism in detail]

The consensus mechanism is implemented in distributed networks that helps in agreeing on some specific value of data. The consensus mechanism in a blockchain maintains a transaction-ledger that is updated only when all the participating nodes accept the transaction as valid. The ledger is updated with the agreed transaction and the updated ledger is broadcast to all the nodes in a network. This process is known as a consensus mechanism. The protocol program that enables the operation of a consensus mechanism is known as the consensus algorithm.

The blockchain is a decentralized technology, that is driven by a consensus mechanism; hence consensus mechanism is considered as the heart of blockchain.

Essentially, the role of consensus mechanism is to provide a robust infrastructure layer for the blockchain, thereby making it the most important part. The consensus ensures

the integrity of the blocks that have to be added in the blockchain. The consensus mechanism in a blockchain network makes sure that the multiple nodes in a distributed network agree on the global state of a blockchain. The blockchain network must fulfill the following conditions prior to reaching a consensus:

- **Agreement**: All honest nodes in a network agree on the same state of data.
- **Fault tolerant**: The consensus mechanism should run smoothly irrespective of the presence of faulty or malicious nodes in a blockchain network.
- **Termination**: All honest nodes exit the consensus process for reaching a common decision.
- **Integrity:** All the honest nodes should make the decision just once in a single consensus cycle.
- **Validity**: All the honest nodes agree to the same initial value proposed by at least one honest node.

Properties of Consensus Mechanism

The applicability and effectiveness of the consensus mechanism is determined based on the following core properties:

- **Safety:** The consensus mechanism is considered safe, if the same output, which is deemed valid, is produced by all nodes. This safety feature is also known as the consistency of the shared legislature.
- **Liveness:** It ensures that all the faulty or malicious nodes participate in a consensus mechanism, and eventually produce a result.
- **Tolerance**: This property ensures fault tolerance if the participant nodes recover from failure and participate in a consensus mechanism.

The additional desirable properties to determine the applicability and effectiveness of the consensus mechanism are as follows:

- **Authentication:** This property provides a mechanism to verify the identity of the participant nodes.
- **Quorum structure**: It ensures that for reaching a consensus in a network, the minimum number of votes must be obtained so that an operation can be performed in a distributed system.
- **Non-repudiation:** Property of non-repudiation provides the mechanism to verify the identity of the actual sender so that the sender cannot repudiate the fact that the message was not sent from its network.
- **Decentralized consensus:** A centralized system cannot claim to provide transaction finality, i.e. it can reverse any transaction for the vested interests of the concerned central authority. Therefore, the consensus mechanism must be decentralized in nature wherein every node must have equal authority. It makes sure that the multiple nodes in a distributed network agree on the common state of a network through a voting mechanism.

Byzantine Generals' Problem

[LO - Describe the Byzantine Generals' Problem]

The "Byzantine Generals' Problem" (BGP) states that in a decentralized network, no two nodes can know for sure and guarantee that they both possess the same state of data. It illustrates that a specific number of signed messages for validating the transactions are received from all the honest nodes, even if the faulty or malicious nodes are present in a network. This criterion is followed to reach a common agreement. Some example protocols derived from the solution to the Byzantine Generals' Problem are Paxos, Viewstamped replication and Raft.

The BGP can be broken into the following two aspects:

- When the communication medium is unreliable or potentially hostile
- When one of the nodes is hostile or malicious

Let's now explain the two aspects in detail.

When the communication medium is unreliable or potentially hostile

In computing domain, the Two Generals' Problem is meant to demonstrate the pitfalls and design challenges that arise while coordinating the action during the communication over an unreliable link. There can be a scenario that two Generals can only communicate via sending messenger through enemy territory. In this case, the main issue arises that how they agree on the time to launch an attack as the messenger passing through the enemy territory can be captured. Two Generals' Problem is very much similar to the BGP.

This problem is also known as Two Generals' Paradox, Two Armies Problem or Coordinated Attack Problem. The Two Generals' Problem was the first computer communication issue that was unsolvable, which provided a base of realistic expectations for any distributed consistency protocol.

Let's now understand the Two Generals' Problem.

The two armies led by two different Generals are preparing to attack a fortified city. Both the armies have encamped near the city in its own valley. The two hills separate the third valley and the only possible way for the Generals to communicate is to send a messenger through the valley. But the third valley is occupied by the city defenders and there is a high probability that the messenger can be captured by them. The problem can arise by the capture of any of the messengers which would lead to distorted communication, thereby making the attack ineffective. Though the two Generals agreed on the fact that they would attack the city but have not agreed upon the attack timings. To succeed, the two Generals must attack at the same time; hence they need to communicate for making a proper attack plan.

A practical approach to deal with the Two Generals' Problem is to use a scheme that can accept the uncertainties of the communication channel by not eliminating it rather than mitigating it to an acceptable degree. For instance, the first General can send more than 100 messengers, thus reducing the probability of capturing of all the messengers. In this approach, the first General will attack irrespective of any situation and the second

General will attack, if he receives a message. In another approach, the first General sends numerous messages and the second General would send an acknowledgement for each message received.

When one of the nodes is hostile or malicious

The main aim of the BGP is to understand whether the consensus can be reached in the event of malicious nodes in a network. The secondary aim is to find the ratio of malicious nodes versus honest nodes up to which the consensus can be reached safely and in a guaranteed manner irrespective of the presence of malicious nodes.

This can be proved with the help of the general case of 'n' nodes starting from 1,2,3,4 and 'n' nodes by induction.

- When only one node is present

  There is no issue as one node will always be in consensus.

- When two nodes are present

  No issue will arise as there is no possibility of miscommunication between two nodes.

- When three nodes are present

  In this case, any of the generals or lieutenants could be traitors that would also lead to confusion while transmitting messages, which in turn would make the attack unsuccessful.

Figure 1 displays BGP when one of the two lieutenants is a traitor:

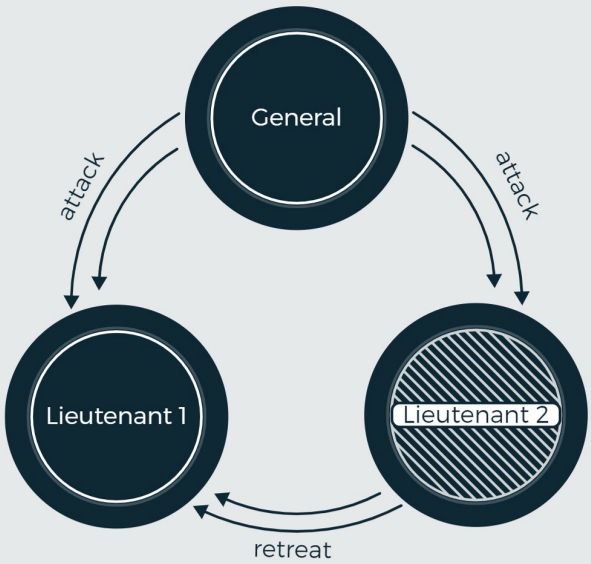

Figure 1: **Byzantine Generals' Problem with Lieutenant 2 as Traitor**

In Figure 1, the striped lieutenant (Lieutenant 2) is a traitor who sends an incorrect message to Lieutenant 1. This makes the task of reaching a consensus impossible. Let's assume that Lieutenant 1 is confused about which order to follow and he follows the General because of strict hierarchy in the army. But this can also create a lot of confusion, if more than one lieutenant is a traitor.

Figure 2 shows BGP when the General is a traitor:

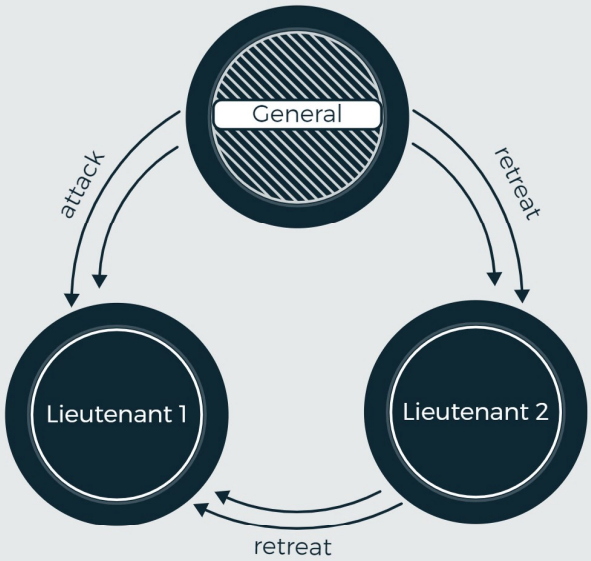

Figure 2: **Byzantine Generals' Problem with General as Traitor**

In Figure 2, the General sends different orders to the lieutenants. Therefore, Lieutenant 1 will receive conflicting messages. The main issue in the given case (1 General, 2 Lieutenants) is that the honest lieutenant cannot figure out who is a traitor, whether it is the General or another lieutenant.

- When four nodes are present

Figure 3 shows BGP when Lieutenant 3 is a traitor:

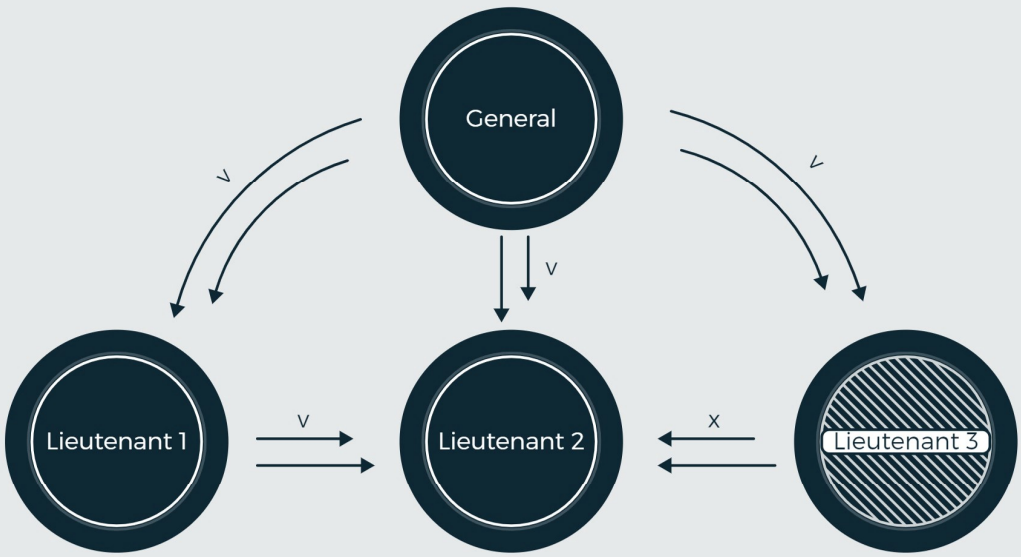

Figure 3: **Byzantine Generals' Problem with Lieutenant 3 as Traitor**

In Figure 3, an honest General sends value v to the lieutenants. All the lieutenants exchange the received values. In case, Lieutenant 3 is a traitor, who sends the false value

x to Lieutenant 2. Hence; the Lieutenant 2 receives values (v, v, x). With the help of the majority function, Lieutenant 2 selects value v.

It is always possible for Lieutenant to reach a common consensus irrespective of the fact that the traitor is General or lieutenant. In case of 4 nodes, it is, therefore, proved that consensus can be reached even though one of the nodes is malicious.

The general inductive proof of this states that if 'F' is the number of malicious nodes, the network will always reach consensus if there are AT LEAST 3F + 1 nodes.

This problem would become extremely complex in a scenario involving hundreds of lieutenants.

Order of Complexity:

Table 1 displays the number of messages required to reach consensus in the case of 'n' nodes with 'm' malicious nodes:

Table 1: Complexity of Nodes

Malicious nodes	Complexity
0	$O(n)$
1	$O(n)^2$
2	$O(n)^3$
3	$O(n)^4$
M	$O(n)^{m+1}$

The examples explained to illustrate BGP encapsulate the greatest difficulty faced in the creation of a trustworthy digital currency before the arrival of bitcoin. Therefore, the solution provided for BGP is considered the biggest achievement of the pseudonymous creator of blockchain technology, Satoshi Nakamoto.

In a distributed network, the BGP arises when thousands of nodes have to verify a transaction and some of the nodes are malicious. In the network, all the nodes are of the same hierarchy, so it is important that all the honest participant nodes need to approve a transaction, thereby nullifying the effect of faulty nodes before reaching an agreement.

In a distributed network, the faults can be categorized into two ways as follows:

- **Fail-stop faults:** These faults are also known as benign faults that prevent the nodes to participate in the consensus mechanism. These faults mostly occur due to hardware or software crashes. When the fail-stop fault occurs, nodes stop responding.
- **Byzantine faults:** These faults can cause the nodes to respond unpredictably.

The solution to a BGP in a distributed network involves hashing, heavy computing work, and communication between all the nodes for verifying the message; however, all this requires a large overhead, making it practically impossible.

Describing Byzantine Fault Tolerance (BFT)

In Byzantine fault, a node can pretend to be an honest node while behaving inappropriately by generating any arbitrary data. BFT is applicable to any system involving a large number of sensors such as airplane engine systems and nuclear power plants. The system can be made fault tolerant by using digital signatures or imposing restrictions on communication among peers in a network.

Byzantine Fault Tolerance is the characteristic that defines a system for tolerating the failures resulted from Byzantine Generals' Problem.

How is Byzantine Fault Tolerance related to Blockchain?

Byzantine Fault Tolerance offers a great solution to a blockchain network from Byzantine Generals' Problem. If the blockchain lacks BFT, then the peer gets the authority to authenticate an invalid transaction; hence challenging the reliability of a blockchain. The inherent design of the blockchain does not have the provision of a central authority to repair the damage done by malicious/faulty nodes. Therefore, the BFT is essential to sustain the decentralized architecture of the blockchain.

Practical Byzantine Fault Tolerance (PBFT) creates BFT networks with lower overheads.

Practical Byzantine Fault Tolerance (PBFT)

PBFT was proposed by Miguel Castro and Barbara Liskov in 1999 that allows BFT applications with low overheads. Hyperledger is one of the primary blockchain solutions that states a roadmap which will make use of PBFT.

PBFT is a consensus algorithm where nodes can be partially trusted. The PBFT uses the concept of primary and secondary nodes. The secondary node authenticates the decisions taken by primary nodes. In case, the primary node is found to be compromised, then the secondary node can switch to a new primary node. A PBFT can easily handle an inconsistent input and provide a correct result at the end. The network requires ***3 F +1*** nodes to tolerate f Byzantine faults.

In PBFT, the node preserves the internal state's copy of the blockchain. As soon as the node receives information, it integrates received information with the internal state to decide on the final state of data. Once the node reaches a conclusion for the newly submitted information, it is broadcast to all nodes in a network. The consensus is reached based on the total number of decisions submitted by all nodes in a network.

The steps performed while reaching a consensus in a network listed are as follows:

1. The nodes in a network submit the valid transactions that are broadcast to all nodes. The transactions are distributed to all the nodes; thus, every node has a full copy of all executed transactions.
2. One of the validating nodes is elected as a leader through a voting mechanism by other nodes (secondary nodes). The order of the transaction is decided by this leader node (primary node) and the transaction order is sent to all the other nodes in a network. The other nodes rearrange the order of transactions according to the order of the leader node.

3. The newly ordered transactions are executed by each validating node and the changes are added to the global state of the blockchain. In case, if an agreement cannot be reached, the transactions are rolled back and rejected.

The above process is repeated for each block; thus, making it fast, reliable and efficient.

Figure 4 displays the PBFT process:

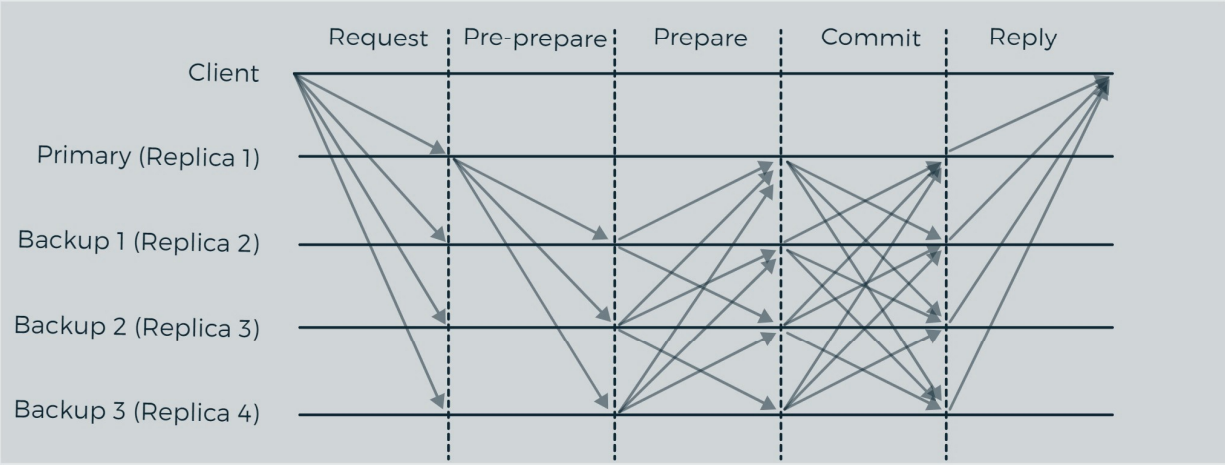

Figure 4: **PBFT Process**

In PBFT process, the leader node is changed during every PBFT consensus cycle and can be substituted by a view change protocol. If leader node fails, then the request is broadcasted to the secondary nodes.

The main disadvantage of PBFT is that all the participating nodes must be authenticated and known to the network. The random node cannot join a network and validate a transaction. Another major disadvantage to PBFT is the exponentially increasing message count as nodes are replicated and added to the set. It is due to the reason that the number of messages required in each step are multiplied by each replica in the set. For instance, for reaching consensus in a network, where 4 nodes are faulty, we need 13 set of replicas and minimum of 237 messages for processing a single request.

SUMMARY

- The consensus mechanism is implemented in distributed networks that helps in agreeing on some specific value of data.
- The blockchain is a decentralized technology, that is driven by a consensus mechanism; hence consensus mechanism is considered as the heart of blockchain.
- Some of the properties of blockchain are liveness, non-repudiation, authentication and decentralized consensus.
- Byzantine Generals' Problem illustrates that a specific number of signed messages for validating the transactions are received from all the honest nodes, even if the faulty or malicious nodes are present in a network, so that a common agreement can be reached. This criterion is followed to reach a common agreement.
- Byzantine Fault Tolerance is the characteristic that defines a system for tolerating the failures resulted from Byzantine Generals' Problem.
- PBFT was proposed by Miguel Castro and Barbara Liskov in 1999 that allows BFT applications with low overheads.