**Q1. Short Note on Ethereum Virtual Machine (EVM)**

The Ethereum Virtual Machine (EVM) is the core runtime environment where all Ethereum smart contracts are executed. It acts as a virtual computer operating on every node of the Ethereum network, ensuring consistent results across the blockchain and maintaining distributed consensus. Ethereum uses its programming language called Solidity, and the EVM executes the compiled bytecode of these contracts. Unlike Bitcoin, which only stores data, Ethereum can also execute contract logic — allowing automation and decentralized applications.

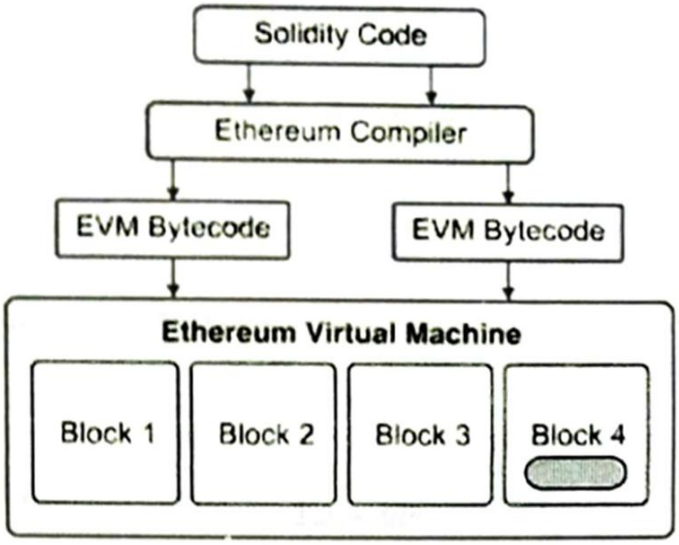



**Fig. 4.5.1**

**Example:** If a contract states that funds should be released only after a product is delivered, the EVM automatically executes the action once the condition is met.

**Benefits:**

- Executes untrusted code safely without risking system data.
- Supports complex and automated smart contracts.
- Ensures distributed consensus and high fault tolerance.

---

**Q2. Explain Types of Test Networks**

Test networks (testnets) are separate blockchains used for testing smart contracts and decentralized applications (dApps) without risking real Ether (ETH). They replicate the Ethereum mainnet environment but use tokens with no real monetary value.

**Types of Test Networks:**

**R**opsten Testnet:

- Based on Proof of Work (PoW) consensus, making it most similar to the Ethereum mainnet.
- Developers use test Ether to simulate real transactions before actual deployment.

Rinkeby Testnet:

- Works on the Proof of Authority (PoA) model, where trusted validators (like the Ethereum Foundation) operate the network.
- More stable than PoW networks, and test Ether is obtained from faucets that often require social media verification.

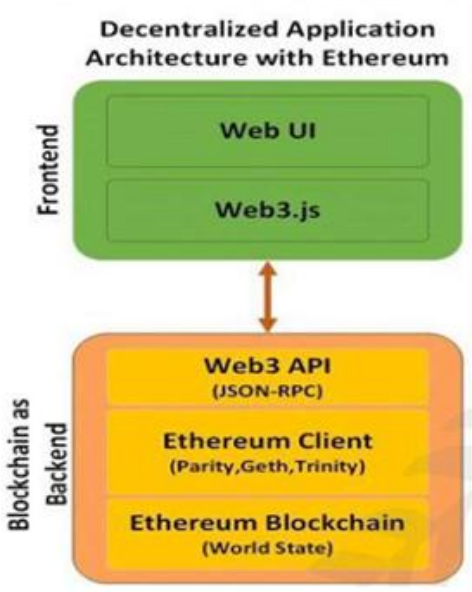
Kovan Testnet:

- Uses the Clique PoA consensus algorithm.
- Offers fast block times and supports Parity's Aura engine for smooth testing..

---

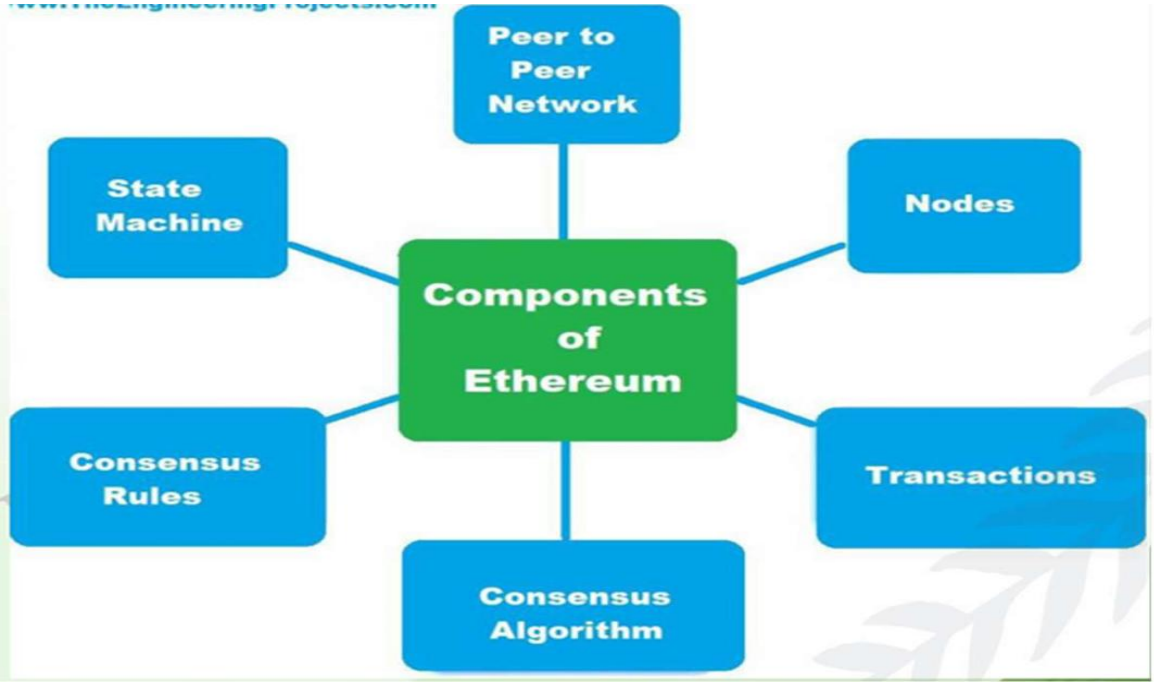
**Q3. Describe the Architecture of Ethereum**

The architecture of **Ethereum** is designed as a layered structure, where each layer performs a specific function to ensure the smooth operation of the blockchain network.
The architecture of Ethereum is layered:



1. **Ethereum Virtual Machine (EVM):**
    1. Acts as a **runtime environment** for executing smart contracts.
    2. Provides a **secure, sandboxed, and deterministic** environment for contract execution.
2. **Consensus Layer:**
    1. Ensures that all **network nodes agree** on the current state of the blockchain.
    2. After *The Merge*, Ethereum uses the **Proof of Stake (PoS)** consensus mechanism instead of Proof of Work.
3. **Data Layer (Blockchain):**
    1. Maintains the **ledger** containing all transactions and smart contract states.
    2. Each block stores **transactions, timestamps, block hashes, and state roots**.
4. **Networking Layer (P2P Network):**
    1. Connects all nodes through a **peer-to-peer network**.
    2. Enables the **broadcasting and synchronization** of transactions and blocks among participants.
5. **Application Layer:**
    1. Provides the **interface** for users to interact with the Ethereum network through **DApps** and smart contracts.
    2. Accessed via **wallets, web interfaces, or decentralized applications**.
6. **Wallets & Accounts:**
    1. **Externally Owned Accounts (EOA):** Controlled by users through private keys.
    2. **Contract Accounts:** Controlled by smart contract code and automatically execute when specific conditions are triggered.

**Q4. Explain Components of Ethereum Blockchain**



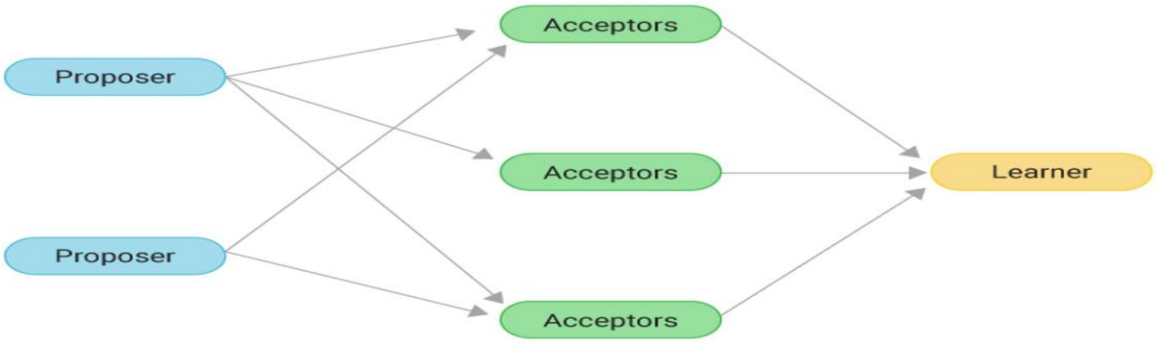
## Components of Ethereum Blockchain

The Ethereum blockchain is made up of several key components that work together to ensure secure, decentralized, and efficient operation.

- **Peer-to-Peer (P2P) Network:**
  - Ethereum operates as a **peer-to-peer network**, where all computers (nodes) are directly connected.
  - This eliminates the need for any **central authority**, making the system fully decentralized.
- **Nodes:**
  - Any device such as a **computer, mobile, or server** connected to the Ethereum network is called a **node**.
  - Each node **stores blockchain data** and communicates with other nodes to share updates and transactions.
- **Transactions:**
  - Transactions are **messages exchanged** between nodes.
  - Each transaction includes a **sender, recipient, amount of Ether**, and **data** (such as smart contract code or function calls).
- **Consensus Rules:**
  - These are the **set of rules** followed by all nodes to verify whether a transaction or block is valid.
  - Consensus rules ensure that all participants **agree on the same blockchain state**.
- **Consensus Algorithm:**
  - It is the **mechanism** that helps all nodes agree on a single valid version of the blockchain.
  - Ethereum initially used **Proof of Work (PoW)** but has now transitioned to **Proof of Stake (PoS)** after *The Merge*.
- **State Machine (Ethereum Virtual Machine - EVM):**
  - Ethereum uses the **EVM** to process transactions and update the blockchain's state.
  - Smart contracts written in **Solidity** are compiled into **bytecode** and executed by the EVM in a secure and deterministic environment.

---

### Q5. Short Note on Paxos Consensus Algorithm



The Paxos consensus algorithm ensures that a group of distributed nodes agree on a single value, even if some nodes fail or the network is unreliable. It is widely used in distributed systems and databases where consistency and reliability are important.

**Roles in Paxos:**

- **Proposers: Propose values to be agreed upon by the network.**
- **Acceptors: Vote on proposals; when a majority of acceptors agree, the value is chosen.**
- **Learners: Learn and announce the final agreed-upon value.**

**Working Principle:**

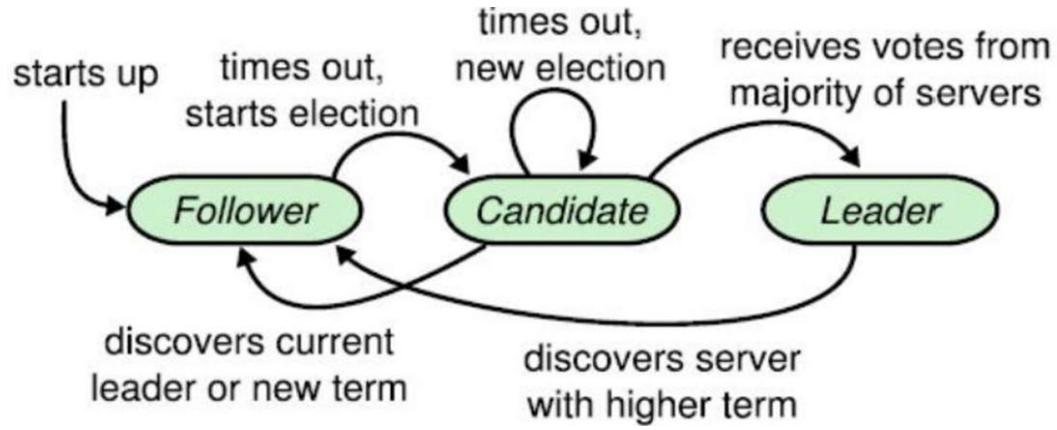
- **Paxos uses ballots (unique identifiers) to order proposals.**
- **A coordinator generates a ballot and requests promises from acceptors.**
- **If a majority of acceptors respond positively, the proposed value is accepted.**
- **This mechanism ensures safety (only one value is ever chosen) and progress**

**Applications:** Distributed databases, file systems, NoSQL stores.

---

**Q6. Short Note on RAFT Consensus Algorithm**

RAFT is a distributed consensus algorithm designed to be simpler and easier to understand than Paxos, while maintaining consistency across nodes.




**Roles:**

**Roles in RAFT:**

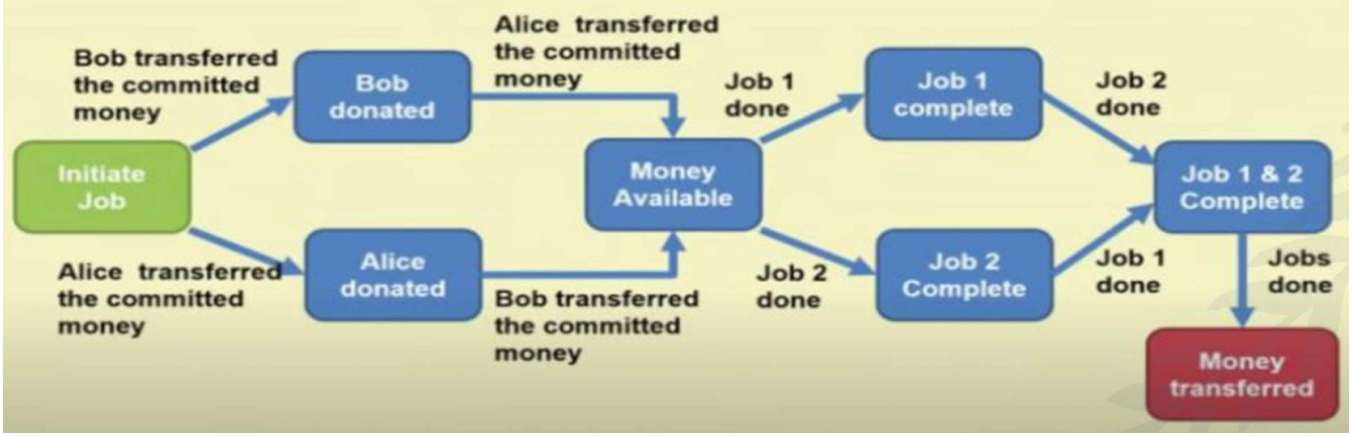
- **Follower:**
  - Every server starts as a follower and waits for messages from the leader.
  - If it does not receive any message for a certain time (timeout), it becomes a **candidate**.
- **Candidate:**
  - Initiates an **election** and requests votes from other servers.
  - If it receives votes from the **majority**, it becomes the **leader**.
  - If it loses the election or detects a higher term, it returns to the follower state.
- **Leader:**
  - The leader **manages and coordinates** the system by handling all client requests.
  - If it discovers a node with a higher term, it steps down and becomes a follower again.

**Example (Supply Chain Blockchain):**

- 5 nodes: Manufacturer (Leader), Supplier, Distributor, Retailer, Auditor.
- Supplier initiates a transaction → Leader validates → Replicates entries → Majority acknowledges → Transaction committed.
- Consensus is maintained even if one node fails.

---

**Q7. Explain State Machine Replication (SMR)**




State Machine Replication ensures consistency in distributed systems by having multiple replicas execute the same operations in the same order. Operations are deterministic, producing the same output and next state across replicas. Fault tolerance allows the system to continue functioning even if some nodes fail. Consensus protocols ensure agreement on operation order.

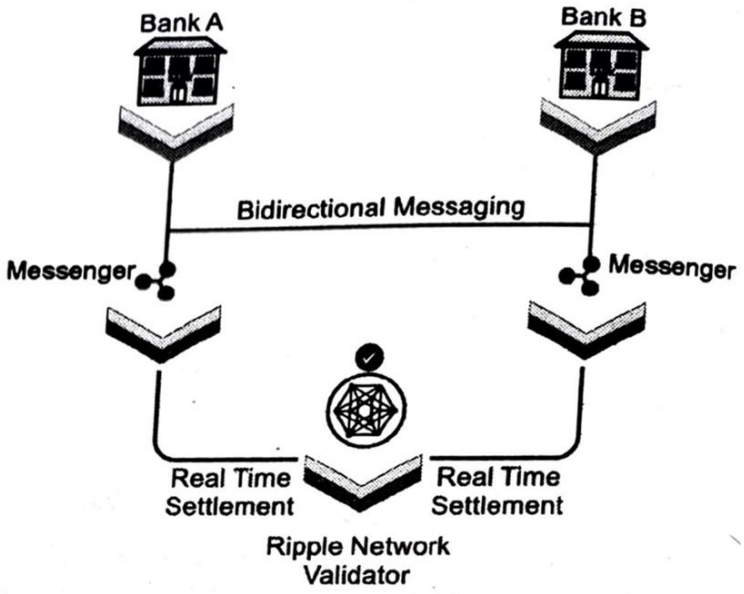
**Example (Crowdfunding):**

- Multiple servers track donations and job progress.
- Initial state: Initiate Job.

- Transactions: Donations processed in same order across all servers.
- State transitions: Money Available → Job 1 Complete → Job 1 & 2 Complete → Money Transferred.
- All servers remain synchronized even if some fail.

---

**Q8. Short Note on Ripple**




(1D6)**Fig. 6.1.1 : Ripple payment system**

**Ripple:**

Ripple is a real-time gross settlement (RTGS) system, currency exchange, and remittance network built on a decentralized, blockchain-like ledger. It primarily focuses on enabling fast, secure, and low-cost international payments between banks and financial institutions.

**Key Features:**

- Ripple's native cryptocurrency, **XRP**, acts as a *bridge currency* to facilitate cross-border transfers between different fiat currencies.
- It uses a **consensus protocol** instead of Proof of Work, making it faster and more energy-efficient.
- Ripple provides a **secure and flexible payment infrastructure**, supporting features like multi-currency transfers and trust-based accounts.
- The network ensures instant settlement, high scalability, and minimal transaction cost.

**Example:**

- Banks exchange payment details via secure messaging.
- Validator nodes verify authenticity and prevent double spending.
- Settlement occurs instantly: funds debited from Bank A and credited to Bank B.

---

**Q9. Short Note on Corda**

**Corda:**

Corda is an **open-source, permissioned blockchain platform** developed by R3 for business applications. It focuses on **secure and private transactions**, particularly for industries like finance, supply chain, and healthcare.

**Architecture & Components:**

- **Nodes & Vaults:** Each participant runs a node with a vault that stores ledger states.
- **States:** Immutable facts on the ledger that evolve by consuming old states and creating new ones.
- **Contracts:** Define rules for updating states, written in JVM languages.
- **Flows:** Govern communication and transaction coordination between nodes.
- **Consensus:** Ensures **validity** and **uniqueness**, preventing double-spending.
- **Notary Service:** Trusted nodes verify uniqueness while maintaining privacy.

**Working of Corda (Transaction Lifecycle):**

1. **Transaction Proposal:** Party A wants to transfer an asset (e.g., bond) to Party B; a new transaction object is created with input (old ownership) and output (new ownership) states.
2. **Verification:** Contract code verifies rules, e.g., “only current owner can transfer the asset.”
3. **Signing:** All required parties digitally sign the transaction.
4. **Notarization:** Transaction is sent to the Notary node to check uniqueness and prevent double-spending.
5. **Recording:** Once notarized, both Party A and Party B record the transaction in their Vaults. Only involved parties see it, unlike public blockchains.

| Basis | Bitcoin | Ethereum |
|---|---|---|
| **Definition** | A decentralized digital currency that can be transferred on the peer-to-peer Bitcoin network. | A decentralized global software platform powered by blockchain technology. Known for its native cryptocurrency, Ether (ETH). |
| **Purpose** | To replace national currencies during the financial crisis of 2008. | To utilize blockchain technology for a decentralized payment network and to store computer code. |
| **Smart Contracts** | Has smart contracts, but they are not as flexible or complete as Ethereum's. | Allows for the creation of smart contracts that are computer codes stored on a blockchain and executed when predetermined conditions are met. |
| **Smart Contract Language** | Written in programming languages like Script and Clarity. | Written in programming languages like Solidity, Vyper, etc. |
| **Transactions** | Generally, Bitcoin transactions are only for keeping notes. | Ethereum transactions may contain executable code. |
| **Consensus Mechanism** | Uses Proof-of-Work (PoW). | Uses Proof-of-Stake (PoS). |
| **Block Time** | Approximately 10 minutes. | Approximately 14 to 15 seconds. |
| **Block Limit** | Has a block limit of 1 MB. | Does not have a block limit. |
| **Structure** | The structure is simple and robust. | The structure is complex and feature-rich. |
| **Rewards** | A miner gets nearly 6.25 BTC along with some additional rewards for successfully adding a new block. | A miner gets nearly 5 ETH along with some additional rewards for successfully adding a new block. |
| **Hash Algorithm** | Runs on the SHA-256 algorithm. | Runs on the Keccak-256 algorithm. |
| **Assets** | The asset of Bitcoin is BTC. | The asset of Ethereum is Ether. |

**Mining Difficulty**

- It is defined as the measure of how difficult it is to find a hash value below a specified target.
- Bitcoin has a global difficulty that changes every 2016 blocks (approximately every two weeks). The aim is to generate one block every 10 minutes, which results in 2016 blocks in two weeks.