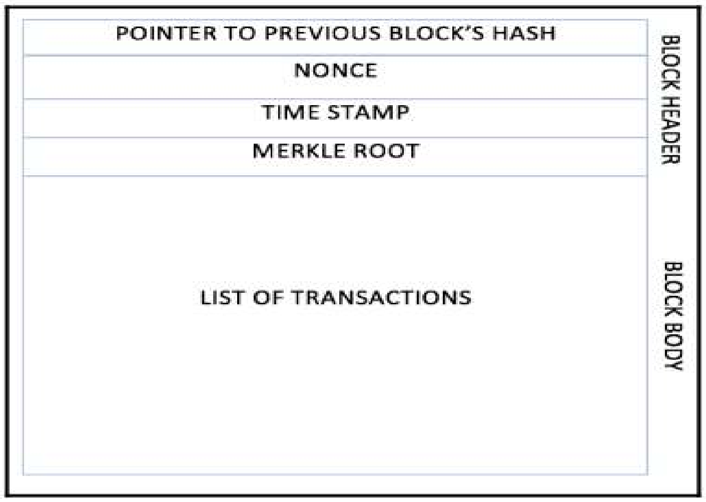
Blockchain Technologies – 22B81A0501

Unit 1

Introduction to Blockchain

Generic Structure of a Block



The generic structure of a block.

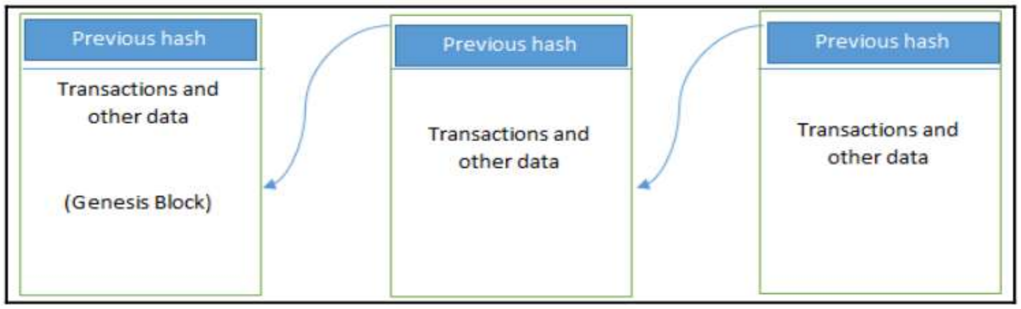
A blockchain block contains several key elements that ensure security, immutability, and traceability:

1. **Nonce**
   - A number used only once.
   - Plays a critical role in cryptographic operations, replay protection, authentication, and mining (proof-of-work).
2. **Merkle Tree & Merkle Root**
   - Transactions in a block are organized into a **Merkle tree**.
   - The **Merkle root**, stored in the block header, is the hash of all transactions.
   - Verifying the Merkle root ensures the integrity of all transactions efficiently, without checking each one individually.
3. **Block Header Components**
   - Previous block hash: links the current block to the previous block.
   - Timestamp: records when the block was created.
   - Difficulty target: used for mining in proof-of-work.

- Merkle root: ensures transaction integrity.
- Nonce: used to solve the proof-of-work puzzle.

Generic Elements of a Blockchain

- Blockchain is a **distributed public ledger** storing **immutable, encrypted data**.
- Ensures **secure transactions** that cannot be altered once confirmed.
- Blocks are linked cryptographically, forming a chain from the **genesis block** (first block) onward.


Previous hash
Transactions and other data
(Genesis Block)
Previous hash
Transactions and other data
Previous hash
Transactions and other data

Generic structure of a blockchain

How Blockchain Works – Step by Step

1. **Transaction Creation**
   - A node creates a transaction and signs it digitally with its private key.
   - Transactions can represent payments, data updates, or smart contract actions.
2. **Transaction Propagation**
   - Transactions are shared with peers using the **Gossip protocol**.
   - Multiple nodes validate the transaction against preset rules.
3. **Block Formation**
   - Validated transactions are grouped into a block.
   - The block is then propagated to the network, confirming the transactions.
4. **Linking Blocks**
   - Each new block contains a hash pointer to the previous block.
   - This cryptographic link ensures immutability.
5. **Transaction Confirmations**

    - Each new block adds another confirmation to previous transactions.
    - Example: Bitcoin typically requires **six confirmations** for a transaction to be considered final.

6. **Mining & Validation**
    - Miners verify transactions and solve computational puzzles (proof-of-work).
    - Successfully mined blocks are added to the blockchain and shared with the network.

Applications of Blockchain

- **Cryptocurrencies:** Bitcoin, Ethereum
- **Decentralized Finance (DeFi):** Lending, borrowing, trading without intermediaries
- **Supply Chain Management:** Track product provenance and reduce fraud
- **Healthcare:** Secure medical records
- **Voting Systems:** Transparent and tamper-proof elections
- **Digital Identity:** Authentication without central authority

Features of Blockchain

- **Decentralization:** No central authority; data managed collectively by nodes
- **Immutability:** Once recorded, data cannot be altered
- **Transparency:** Transactions visible to network participants
- **Security:** Cryptographic techniques protect data integrity
- **Consensus-based:** Network agreement validates transactions

Limitations of Blockchain

- **Scalability:** High computational cost and slower transaction speeds
- **Energy Consumption:** Especially in proof-of-work systems
- **Privacy:** Public ledgers may expose transaction details
- **Regulatory Uncertainty:** Not legally recognized in all jurisdictions
- **Complexity:** Requires technical expertise to implement

Tiers of Blockchain Technology

Blockchain technology has evolved in generations or tiers, each adding new capabilities beyond simple cryptocurrency transactions.

1. Blockchain 1.0 – Cryptocurrencies

- **Definition:** The first implementation of distributed ledger technology focused on digital money.
- **Introduction:** Hal Finney introduced the concept in 2005; Bitcoin was launched in 2009.
- **Purpose:** Solely designed for **cryptocurrencies** like Bitcoin.
- **Features:**
    - Uses cryptography to secure transactions.
    - Peer-to-peer money transfer without intermediaries.
- **Limitation:** Only handles financial transactions; no additional programmable logic.

2. Blockchain 2.0 – Smart Contracts

- **Definition:** A blockchain capable of running **automated programs** called smart contracts.
- **Functionality:**
    - Automatically facilitates, verifies, or enforces agreements.
    - Secure and tamper-resistant execution of programs.
- **Applications:**
    - Enables developers to build **decentralized applications (dApps)** on platforms like **Ethereum**.
- **Advantage:** Goes beyond currency to programmable, trustless contracts.

3. Blockchain 3.0 – Decentralized Applications (DApps)

- **Definition:** Fully decentralized applications built on blockchain infrastructure.
- **Architecture:**
    - Frontend hosted on decentralized storage.
    - Backend interacts with the blockchain via smart contracts.
- **Capabilities:**

    - Leverages decentralized storage and communication.
    - Can provide services similar to traditional apps but without central authority.
- **Examples:** Decentralized marketplaces, peer-to-peer ride-sharing platforms, and secure identity management systems.

Distributed Systems and Blockchain

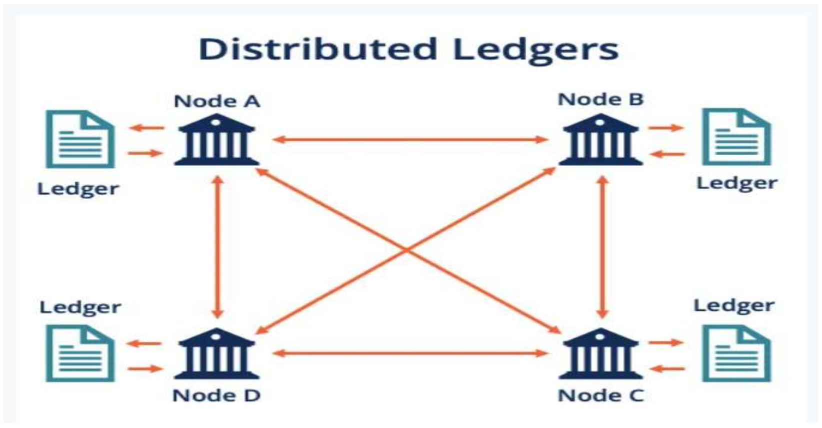
Blockchain is a type of **distributed system**, meaning its components (nodes) are spread across multiple locations and work together to maintain a shared state. In distributed systems, there is no single central authority controlling the network, which ensures redundancy, fault tolerance, and decentralization.

Distributed Ledgers

Definition:
A **ledger** is a record of financial accounts or transactions. When a ledger is **distributed**, it is spread across multiple participants and sites, allowing all nodes to access, update, and verify the data independently.

Key Features:

- **Global control:** Distributed across various geographical locations.
- **Node-based management:** Nodes maintain, validate, and reorganize ledger data.
- **No third party:** Transactions can be verified without intermediaries.
- **Security and transparency:** Decentralization ensures tamper-resistance and visibility.
- **Types:** Can be **public** (open to anyone) or **private** (restricted access).



Example:

- **R3's Corda:** A distributed ledger that does **not use blockchain-style blocks**. It focuses on recording and managing agreements, especially for the financial services sector.

Distributed Ledger Technology (DLT)

Definition:
DLT refers to technologies that enable shared ledgers across a network, often used in finance. Blockchain is a type of DLT, though sometimes the terms are used interchangeably.

Characteristics of DLT:

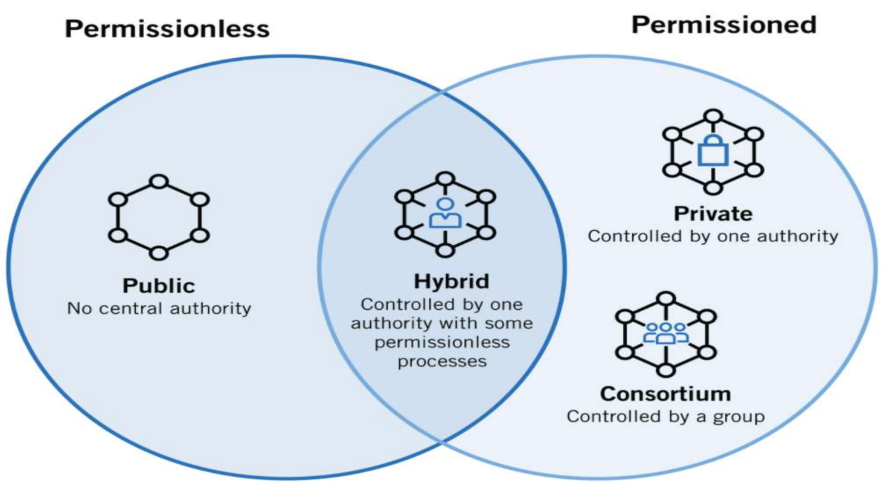
- **Permissioned networks:** Only known and verified participants can access and update the ledger.
- **Shared database:** All participants maintain a copy of the ledger.
- **No native cryptocurrency needed:** Some DLTs, unlike blockchain, do not require mining or tokens to secure the ledger.
- **Applications:** Widely used in banking, insurance, and enterprise ecosystems to track transactions efficiently and securely.

Key Differences Between Blockchain and DLT

Feature	**Blockchain**	**DLT**
Structure	Uses blocks of transactions	May not use blocks
Cryptocurrency	Usually has tokens	May not require tokens
Permission	Public or private	Usually permissioned
Use Case	Broad (finance, supply chain)	Mainly enterprise & finance

Distributed systems, ledgers, and DLT form the foundation of blockchain technology.

Types of Blockchain



1. Public Blockchain

- Also called **permissionless blockchain**.
- Completely open for anyone to join, participate in the consensus process, and maintain the shared ledger.
- **Examples:** Bitcoin, Ethereum.

Advantages:

- Highly secure due to decentralization and distributed consensus.
- Fully transparent, with all transactions publicly verifiable.

Disadvantages:

- Low privacy, as all transactions are visible.
- High computational power and energy consumption, making it less eco-friendly.

2. Private Blockchain

- Also called **permissioned blockchain**.
- Access is **restricted**; participants need an invitation or authorization from the network initiator.
- Used mainly for internal enterprise or organizational purposes.

Advantages:

- Increased privacy as only authorized participants can access the network.
- More eco-friendly; less computational power is required to achieve consensus.

Disadvantages:

- Less secure than public blockchains due to centralized control.
- Reduced transparency compared to public networks.

3. Hybrid Blockchain

- Combines features of both **public and private blockchains**.
- Certain data or functionalities are made public, while sensitive data remains private.
- Allows controlled participation and selective transparency.

Advantages:

- Flexibility to control who can access data.
- Balances privacy, security, and decentralization.

Disadvantages:

- Complexity in governance and consensus management.

4. Consortium Blockchain

- Permissioned blockchain where **multiple organizations share control** over the network.
- Consensus is managed by a **pre-selected group of nodes**, not a single organization.
- Commonly used in banking, finance, and supply chain applications.

Advantages:

- More decentralized than private blockchain but still controlled.
- Faster and more efficient consensus compared to public blockchains.

Disadvantages:

- Less transparent than public blockchains.
- Requires trust among participating organizations.

Consensus in Blockchain

Definition:
Consensus is the backbone of blockchain. It is a process by which distributed and potentially distrustful nodes in a network agree on the final state of data. Simply put, consensus ensures that all nodes share a common agreement on the value or state of the system.

Importance of Consensus

- Provides **decentralization** and eliminates reliance on a single authority.
- Ensures all nodes maintain the **same copy of the ledger**.
- Enables blockchain networks to function securely even in the presence of faulty or malicious nodes (Byzantine nodes).

Conditions for Consensus

A consensus algorithm must satisfy the following:

1. **Agreement:** All honest nodes decide on the same value.
2. **Validity:** The agreed value must match the initial value proposed by at least one honest node.
3. **Termination:** All honest nodes eventually reach a decision and stop the process.
4. **Fault Tolerance:** Should operate correctly even if some nodes are faulty or malicious.

5. **Integrity:** No node can make a decision more than once per consensus cycle.

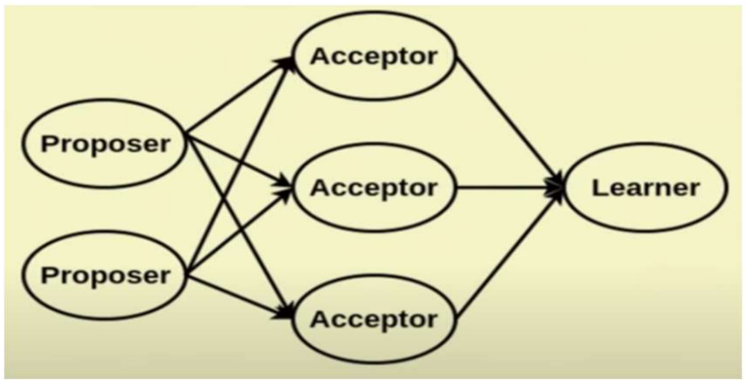
Analogy: Like generals deciding whether to attack or retreat: a coordinated decision avoids disaster, while uncoordinated actions lead to defeat.

Categories of Consensus Mechanisms

1. **Traditional Byzantine Fault Tolerance (BFT)**
   - Handles arbitrary faults in distributed systems.
   - Ensures agreement despite the presence of malicious nodes.
2. **Leader Election-Based Consensus**
   - Nodes elect a leader to propose and commit changes.
   - Common in practical systems like Raft.

Examples of Consensus Protocols

1. **Paxos**
   - Introduced by Leslie Lamport (1989).
   - Nodes have roles: **Proposer, Acceptor, Learner**.
   - Achieves agreement among a majority of nodes, tolerating faulty nodes.



PAXOS – Type of nodes

2. **Raft**
   - Developed by Diego Ongaro and John Ousterhout.
   - Nodes can be **Leader, Candidate, or Follower**.
   - A leader is elected by majority vote; all changes go through the leader and are replicated to followers before committing.

Types of Consensus Algorithms (Refer Assignment)

CAP Theorem (Brewer's Theorem)

- **Introduced:** Eric Brewer, 1998 (as a conjecture)
- **Proven:** Seth Gilbert and Nancy Lynch, 2002

The **CAP theorem** states that a **distributed system** can achieve **at most two** of the following three properties simultaneously:

1. **Consistency (C)**
    - Ensures that **all nodes** in the system have the **same, up-to-date copy of the data**.
    - Any read request returns the latest write.
2. **Availability (A)**
    - Every node is **operational and responsive**, providing data even if some nodes fail.
    - Requests to any node always receive a response (may not be the latest data if consistency is sacrificed).
3. **Partition Tolerance (P)**
    - The system continues to **operate correctly even if some nodes cannot communicate** due to network failures.
    - Critical for distributed networks where node or network failures are inevitable.

CAP Theorem in Blockchain

- Blockchain systems **prioritize Availability (A) and Partition Tolerance (P)** over immediate Consistency (C).
- **Consistency is not instantaneous**: due to network delays, nodes may have slightly different versions of the ledger temporarily.
- Over time, the system reaches **eventual consistency**, meaning all nodes converge to the same ledger state.

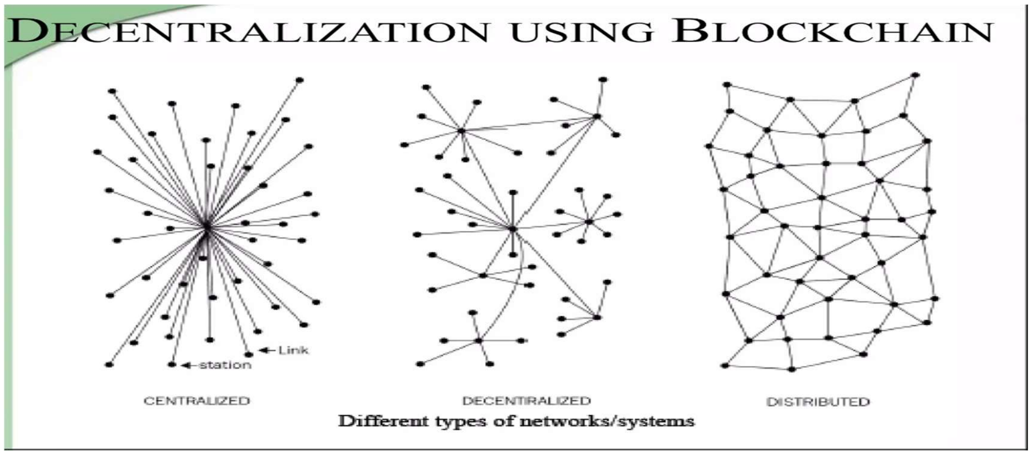
Key Point:

- Unlike traditional databases, blockchain allows temporary inconsistencies but guarantees that all nodes will eventually agree on the same data.

Decentralization Using Blockchain

Definition

Decentralization in blockchain refers to the transfer of control and decision-making from a **central authority** to a **distributed network** of participants. Instead of relying on a single trusted entity, blockchain relies on **consensus mechanisms** and **cryptographic security** to ensure that transactions are valid, transparent, and tamper-resistant. This makes the system more resilient, transparent, and less prone to corruption or single points of failure.


DECENTRALIZATION USING BLOCKCHAIN
Link
station
CENTRALIZED
DECENTRALIZED
DISTRIBUTED
Different types of networks/systems

Centralized Systems

- In a **centralized system**, all data, decision-making, and control are handled by a **single central authority** (e.g., a bank, government agency, or traditional server).
- All users must trust this authority to manage records, process transactions, and maintain security.
- If the central authority fails, gets hacked, or acts maliciously, the entire system is compromised.
- Example: Google, Amazon, eBay and others use this conventional model for delivering services.

Decentralized Systems

- A **decentralized system** distributes control among multiple nodes or participants.
- No single entity has full authority; instead, decisions are made through **consensus protocols** or majority agreement among participants.
- Even if some nodes fail or act dishonestly, the system continues to function securely.
- Blockchain is a prime example: miners/validators collectively maintain and verify transactions.
- Example: Bitcoin network, where multiple nodes verify and record transactions.

Distributed Systems

- A **distributed system** refers to a network where computation and data are spread across multiple nodes, but these nodes may still be controlled by a central entity or coordinated system.
- Distribution improves **speed, scalability, and reliability**, since tasks are shared across different machines.
- However, being distributed does not always mean being decentralized—control could still lie with one organization.
- Example: Google's data centers distributed worldwide but still centrally controlled by Google.

Methods of Decentralization (Refer Assignment)

Routes to Decentralization

Even before blockchain, decentralized systems existed, such as **BitTorrent** and **Gnutella**, which allowed peer-to-peer sharing without a central authority. Blockchain technology, however, has greatly expanded the potential for decentralization, enabling applications like **Bitcoin** and **Ethereum**.

- **Bitcoin**: Primarily a decentralized digital currency.
- **Ethereum**: Provides a flexible platform for building **decentralized applications (dApps)** using **smart contracts**, making it a popular choice for developers.

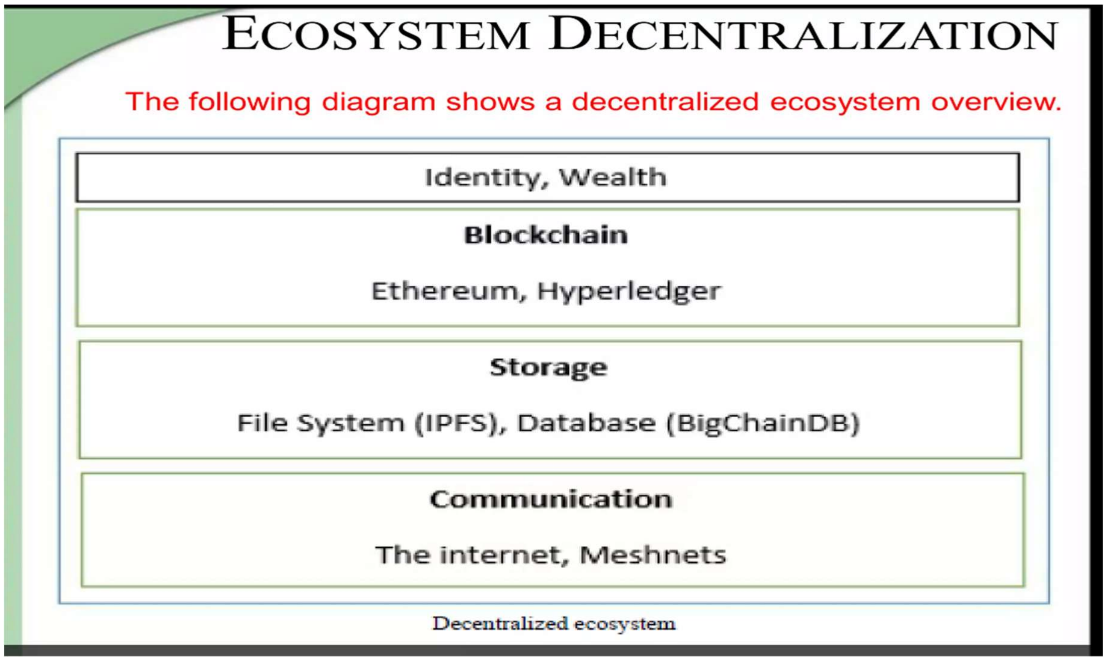
How to Decentralize a System

To design a decentralized system, four critical questions must be addressed:

1. **What is being decentralized?** – Data, computation, identity, assets, or governance.
2. **What level of decentralization is required?** – Full, partial, or layered decentralization.
3. **Which blockchain is used?** – Different blockchains provide varying capabilities and consensus mechanisms.
4. **What security mechanism is implemented?** – Cryptography, consensus protocols, and access controls.

Ecosystem Decentralization

Decentralization is not just about the blockchain itself—it requires that the **entire ecosystem**, including storage, communication, and computation, is decentralized.


ECOSYSTEM DECENTRALIZATION
The following diagram shows a decentralized ecosystem overview.
Identity, Wealth
Blockchain
Ethereum, Hyperledger
Storage
File System (IPFS), Database (BigChainDB)
Communication
The internet, Meshnets
Decentralized ecosystem

1. Storage

- Direct storage on a blockchain is possible but **not suitable for large datasets**.
- Alternatives include **Distributed Hash Tables (DHTs)** and platforms like:
  - **IPFS** (InterPlanetary File System)
  - **Ethereum Swarm**
  - **Storj** and **MaidSafe**
  - **BigchainDB**: Provides a fast, scalable decentralized database that complements decentralized computing platforms.

2. Communication

- The internet is partially decentralized, but **ISPs act as central hubs**, making users dependent on them.
- True decentralization requires giving **control to individual users**, ensuring communication is not dependent on a single authority.
- Alternatives include **mesh networks**, where nodes communicate directly, e.g., **FireChat** on iPhone.
- Future networks could allow fully decentralized, resilient communication without central points of failure.

3. Computing Power

- Blockchain platforms like **Ethereum** enable decentralized computation by executing **smart contracts** across the network.
- Other blockchains also provide decentralized processing layers, reducing reliance on a single server or authority.

Layered Approach to Ecosystem Decentralization

A decentralized blockchain ecosystem can be visualized in layers:

1. **Communication layer**: Internet or mesh networks providing peer-to-peer connectivity.
2. **Storage layer**: Decentralized systems like IPFS, BigchainDB, or Swarm for scalable, distributed storage.
3. **Computation layer**: Blockchain platforms like Ethereum for executing business logic in a decentralized manner.
4. **Top layers**: Identity and wealth management, e.g., authentication systems like BitAuth or OpenID, ensuring secure and decentralized identification.

- **Zooko's Triangle**: A principle stating that naming systems in decentralized networks should be **secure, decentralized, and human-readable**, balancing usability with decentralization.

Decentralization-Smart Contracts

Decentralization enables systems, organizations, and societies to operate without reliance on a single central authority, often leveraging **blockchain technology** for security, transparency, and autonomy.

Smart Contracts

- A **smart contract** is a self-executing, decentralized program containing business logic and a small amount of data.
- It runs on blockchain networks (though not strictly required) to leverage security and immutability.
- Smart contracts can operate autonomously or be triggered by participants when certain conditions are met.

Decentralized Organizations and Autonomous Systems

1. **Decentralized Organizations (DOs)**

    - Software-based organizations that operate on a blockchain using smart contracts.
    - Interactions are governed by pre-defined rules encoded in the software.
    - Human input may still be required for execution.
2. **Decentralized Autonomous Organizations (DAOs)**
    - Fully automated organizations running on blockchain.
    - Contain governance and business logic rules with minimal or no human intervention.
    - DAOs are essentially autonomous versions of DOs.
3. **Decentralized Autonomous Corporations (DACs)**
    - Similar to DAOs but may operate as profit-generating entities.
    - Can issue shares, earn profits, and distribute dividends automatically.
4. **Decentralized Autonomous Societies (DASs)**
    - Entire societal functions can operate via blockchain using multiple smart contracts, DAOs, and DApps.
    - Examples include digital identity systems, government services like passports, birth/death records, and other public services.

Decentralized Applications (DApps)

DApps are applications running on decentralized networks. Examples include:

1. **KYC-Chain** – Secure management of Know Your Customer (KYC) data using smart contracts.
2. **OpenBazaar** – Peer-to-peer marketplace enabling commerce without centralized intermediaries.
3. **Lazooz** – Decentralized ride-sharing platform, rewarding users with Zooz tokens for participation.

- DApps often leverage **Distributed Hash Tables (DHTs)** for peer-to-peer communication and direct data sharing.

Platforms for Decentralization

Blockchain networks provide the infrastructure for decentralization. Some key platforms include:

1. **Ethereum**
    - First blockchain to offer a **Turing-complete language (Solidity)** and virtual machine.
    - Enables development of smart contracts and decentralized applications.
    - Public blockchain; native currency is **Ether (ETH)**.
2. **MaidSafe**
    - SAFE network utilizes unused storage, processing power, and data connections.
    - Data is encrypted, split into chunks, and distributed across the network.
    - Incentivizes contributors via **Safecoin**.
3. **Lisk**
    - Allows DApp development in **JavaScript** using sidechains.
    - Uses **Delegated Proof of Stake (DPoS)** for consensus with 101 elected nodes securing the network.