BLOCK CHAIN TECHNOLOGY

DIPLOMA WALLAH

OPEN ELECTIVE

Jharkhand University Of Technology (JUT)

UNIT - II: CONSENSUS PROTOCOLS AND BLOCKCHAIN SCALABILITY

Requirements for Consensus Protocols

Consensus protocols are fundamental mechanisms that enable distributed blockchain networks to agree on a single version of truth without requiring a central authority. Understanding the requirements and design objectives is essential for selecting appropriate consensus mechanisms.komodoplatform+1

Fundamental Requirements

1. Agreement (Safety)

All honest nodes must agree on the same value or state of the ledger. Even if some nodes fail or act maliciously, the remaining honest nodes should reach consensus on identical data.ceur-ws+1

Example: If Node A believes Transaction X is valid and included in Block 100, then all other honest nodes must also agree that Transaction X is valid and in Block 100.

2. Validity

The agreed-upon value must be valid according to the system's rules. Consensus should only be reached on legitimate transactions that follow the protocol specifications.cxotechmagazine+1

Example: Nodes should reject transactions that attempt to double-spend coins or violate cryptographic signature requirements.

3. Termination (Liveness)

The consensus process must eventually complete. All non-faulty nodes must eventually reach a decision within finite time, ensuring the system doesn't halt indefinitely.persistent+1

4. Fault Tolerance

The protocol must continue operating correctly even when some nodes fail or behave maliciously. The system should tolerate:persistent+1

- **Crash faults:** Nodes that stop functioning without warning.geeksforgeeks+1

- **Byzantine faults:** Nodes that behave arbitrarily or maliciously, sending conflicting information to different nodes.omgwiki+1

Fault tolerance threshold: Most Byzantine Fault Tolerant systems require that fewer than one-third (< 33.3%) of nodes are malicious for consensus to be achieved.pdfs.semanticscholar+2

5. Integrity

No honest node should make the same decision twice for different values. Once consensus is reached, it should be final and immutable.geeksforgeeks

Performance Criteria for Consensus Protocols

When selecting a consensus mechanism, several performance metrics must be evaluated:sciencedirect+1

1. Transaction Speed (Throughput)

Definition: The number of transactions the network can process per unit time, measured in Transactions Per Second (TPS).101blockchains+1

Formula:

$$v_T = \frac{n}{t}$$

Where:

- v_T= transaction speed
- n= number of transactions processed
- t= time taken

Requirements: Applications like payment systems and micropayments require high TPS. Traditional payment processors (Visa) handle thousands of TPS, while Bitcoin achieves only 7-10 TPS and Ethereum 12-15 TPS.hedera+2

2. Latency

Definition: The delay between submitting a transaction request and receiving confirmation that it has been added to the blockchain.thesai+1

Impact factors:

- Consensus mechanism complexity101blockchains
- Network bandwidth101blockchains
- Number of validation rounds required101blockchains

Requirements: Time-sensitive applications (trading platforms, real-time payments) require low latency.debutinfotech+1

3. Throughput

Definition: The total volume of data the blockchain can process in a given timeframe.thesai+1

Relationship to latency: These two metrics are inversely related—improving one often degrades the other.101blockchains

4. Scalability

Definition: The ability of the consensus protocol to handle increasing numbers of nodes and transactions without performance degradation.ceur-ws+2

Challenge: As the network grows, maintaining consensus becomes more complex and resource-intensive.treasuryxl+1

5. Energy Consumption

Definition: The computational resources and electrical power required to maintain consensus.sciencedirect+1

Variation: Proof of Work consumes enormous energy, while Proof of Stake and permissioned consensus mechanisms are far more energy-efficient.komodoplatform+2

6. Security

Definition: The protocol's resistance to attacks, including 51% attacks, Sybil attacks, and Byzantine attacks.ceur-ws+1

Trade-off: Higher security often requires more computational resources or slower transaction processing.komodoplatform+1

7. Decentralization

Definition: The degree to which control is distributed among network participants rather than concentrated in a few entities.treasuryxl+1

Importance: Greater decentralization enhances censorship resistance and reduces single points of failure.komodoplatform

8. Finality

Definition: The point at which a transaction is considered irreversible and permanently recorded.persistent+1

Types:

- **Probabilistic finality:** Confidence increases with each additional block (used in PoW).persistent
- **Deterministic finality:** Immediate irreversibility once consensus is reached (used in BFT protocols).geeksforgeeks+1

9. Communication Complexity

Definition: The number of messages exchanged between nodes to reach consensus.halborn+1

Impact: Higher communication complexity increases network overhead and can limit scalability.halborn

Network-Specific Requirements

1. Network Type Consideration

- **Permissionless networks:** Require mechanisms to handle unknown, potentially malicious participants (typically use PoW or PoS).komodoplatform+1
- **Permissioned networks:** Can use more efficient consensus mechanisms since participants are known and vetted (typically use PBFT, Raft, or Paxos).academia+2

2. Trust Assumptions

- **No trust (permissionless):** Assume adversarial environment where participants cannot be trusted.komodoplatform
- **Partial trust (permissioned):** Assume most participants are honest, though some may fail.academia+1

3. Protocol Flexibility

The ability to adjust algorithm parameters to achieve optimal performance under specific conditions.ceur-ws

---

Proof of Work (PoW) - Detailed Explanation

Proof of Work is the original consensus mechanism introduced by Satoshi Nakamoto in the Bitcoin whitepaper (2008). It remains one of the most battle-tested and secure consensus protocols.geeksforgeeks+2

Core Concept

Definition: PoW is a consensus mechanism that requires network participants (miners) to expend significant computational effort to solve complex cryptographic puzzles in order to validate transactions and create new blocks.businessinsider+3

Fundamental principle: The solution should be difficult to find but easy to verify.trezor+1

How Proof of Work Functions

Step 1: Transaction Broadcasting

Users initiate transactions by broadcasting them to the network. Each transaction includes:21lectures+1

- Sender's address
- Recipient's address
- Amount to transfer
- Digital signature (proving sender authorization)
- Transaction fee

Step 2: Transaction Pool (Mempool)

Unconfirmed transactions are collected in a memory pool (mempool) where miners select transactions to include in the next block.21lectures

Step 3: Block Creation

Miners gather valid transactions from the mempool and create a candidate block containing:trezor+1

- **Block header:**
  - Version number
  - Previous block hash (linking to the blockchain)
  - Merkle root (summary of all transactions)
  - Timestamp
  - Difficulty target
  - Nonce (variable number that miners change)
- **Block body:** List of transactions

Step 4: Cryptographic Puzzle Solving (Mining)

The mining challenge: Find a nonce value such that when the block header is hashed using SHA-256, the resulting hash is less than the target difficulty value.geeksforgeeks+2

Mathematical representation:

$$\text{Hash(Block Header + Nonce)} < \text{Target}$$

Process:

1. Miner sets nonce = 0
2. Hash the block header
3. Check if hash < target

4. If no, increment nonce and repeat
5. If yes, broadcast the solution

Characteristics of the puzzle:

- **Computationally expensive:** Requires billions of hash calculations.geeksforgeeks+1
- **Random guessing:** No mathematical shortcut exists; miners must try different nonce values through brute force.trezor
- **Probabilistic:** More computational power = higher probability of finding the solution first.21lectures+1

Example (simplified):

Target:
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Block header data: Previous Hash + Merkle Root + Timestamp + Nonce

Miners try:

- Nonce = 1 → Hash = A3F2... (doesn't meet target)
- Nonce = 2 → Hash = 8B4D... (doesn't meet target)
- ...
- Nonce = 2,847,193 → Hash = 0000AB2F... (meets target! Solution found)

Step 5: Block Verification

Once a miner finds the valid nonce:

1. The miner broadcasts the block to the networkbusinessinsider+1
2. Other nodes verify the solution instantly by computing one hashbusinessinsider+1
3. If valid, nodes add the block to their copy of the blockchainbusinessinsider

Step 6: Reward Distribution

The successful miner receives:debutinfotech+1

- **Block reward:** Newly minted cryptocurrency (Bitcoin: currently 3.125 BTC per block as of 2024, halving every 4 years)businessinsider
- **Transaction fees:** Sum of all fees from transactions included in the blockdebutinfotech+1

Step 7: Difficulty Adjustment

Purpose: Maintain consistent block creation time despite fluctuating total network hash power.rapidinnovation+2

Bitcoin example:

- **Target block time:** 10 minuteswikipedia+1
- **Adjustment period:** Every 2,016 blocks (approximately 2 weeks)rapidinnovation
- **Adjustment mechanism:**
  - If blocks are mined faster than 10 minutes on average → increase difficulty (lower target value)
  - If blocks are mined slower than 10 minutes on average → decrease difficulty (higher target value)rapidinnovation+1

Formula:

$$\text{New Difficulty} = \text{Old Difficulty} \times \frac{\text{Expected Time}}{\text{Actual Time}}$$

Mining Hardware Evolution

1. CPU Mining (2009-2010)

Early Bitcoin mining used standard computer processors. Relatively low hash rates and energy consumption.komodoplatform

2. GPU Mining (2010-2013)

Graphics cards provided parallel processing capabilities, offering 50-100x improvement over CPUs.komodoplatform

3. FPGA Mining (2011-2013)

Field-Programmable Gate Arrays offered more efficiency than GPUs but required technical expertise.

4. ASIC Mining (2013-present)

Application-Specific Integrated Circuits are hardware designed exclusively for mining specific cryptocurrencies.21lectures+1

Characteristics:

- Extremely high hash rates (terahashes per second)
- Most energy-efficient per hash
- Expensive initial investment
- Cannot be repurposed for other tasks

Security Properties of PoW

1. Protection Against Double-Spending

Double-spending attack: Attempting to spend the same cryptocurrency twice.trezor+1

PoW prevention: Once a transaction is included in a block and subsequent blocks are added on top, reversing that transaction requires redoing all the computational work for that block and all following blocks. This becomes exponentially more difficult as more blocks are added.debutinfotech+1

6-block confirmation rule: Bitcoin typically considers transactions with 6 confirmations (6 blocks deep) as irreversible.trezor

2. 51% Attack Resistance

51% attack: If an attacker controls more than 50% of the network's total hash power, they could potentially:geeksforgeeks+1

- Prevent new transactions from being confirmed
- Reverse recent transactions they sent (double-spend their own coins)
- Prevent other miners from mining valid blocks

Why PoW resists 51% attacks:

- **Economic cost:** Acquiring 51% of Bitcoin's hash power would cost billions of dollars in hardware and electricity.debutinfotech+1
- **Diminishing returns:** Successfully attacking would likely crash the cryptocurrency's value, making the attacker's investment worthless.debutinfotech
- **Detectability:** The attack would be obvious to the network, prompting community response.debutinfotech

3. Sybil Attack Resistance

Sybil attack: Creating multiple fake identities to gain disproportionate influence.geeksforgeeks

PoW prevention: Influence is determined by computational power, not number of identities. Creating multiple mining nodes without increasing total hash power provides no advantage.geeksforgeeks+1

Advantages of Proof of Work

1. Proven Security

- Bitcoin has operated securely for over 15 years without a successful 51% attack.businessinsider+1
- Battle-tested against numerous attack attempts.21lectures+1

2. True Decentralization

- Anyone with computational resources can participate as a miner.trezor+1
- No permission required to join the network.komodoplatform

3. No Initial Wealth Requirement

- Unlike Proof of Stake, miners don't need to own cryptocurrency to begin mining.komodoplatform
- Only requires hardware and electricity.komodoplatform

4. Objective Consensus

- Consensus is mathematically verifiable.geeksforgeeks
- No subjective interpretation or voting required.geeksforgeeks

5. Immutability

- Historical transactions become increasingly difficult to alter.debutinfotech+1
- Provides strong guarantees of data permanence.debutinfotech

Disadvantages of Proof of Work

1. Massive Energy Consumption

- **Scale:** Bitcoin network consumes electricity comparable to entire countries (e.g., comparable to Ireland's total consumption).persistent
- **Environmental impact:** Large carbon footprint, especially when powered by fossil fuels.businessinsider+2
- **Sustainability concerns:** Ongoing debate about long-term viability.debutinfotech+1

2. Poor Scalability

- **Limited throughput:** Bitcoin processes only 7-10 transactions per second.hedera+2
- **Comparison:** Visa processes thousands of TPS.hedera+1
- **Bottleneck:** Block size and block time constraints limit transaction capacity.wikipedia+1

3. High Latency

- **Confirmation time:** Bitcoin transactions typically require 10-60 minutes for confirmation.treasuryxl+1
- **Not suitable for:** Real-time payments or instant transactions.geeksforgeeks

4. Resource Wasteful

- **Computational waste:** Millions of unsuccessful hash calculations are discarded.persistent+1
- **Hardware obsolescence:** Mining equipment becomes outdated quickly.komodoplatform

5. Centralization Risks

- **Mining pools:** Individual miners join pools to share rewards, leading to concentration of hash power.geeksforgeeks+1
- **Geographic centralization:** Mining tends to concentrate in regions with cheap electricity.komodoplatform
- **ASIC dominance:** Expensive specialized hardware creates barriers to entry.komodoplatform

6. Economic Barriers

- **High initial investment:** ASIC miners cost thousands of dollars.komodoplatform
- **Operating costs:** Electricity bills can be prohibitive.persistent+1
- **Economies of scale:** Large mining operations have competitive advantages over individuals.komodoplatform

Notable PoW Cryptocurrencies

- **Bitcoin (BTC):** SHA-256 hashing algorithmtrezor+2
- **Litecoin (LTC):** Scrypt hashing algorithmkomodoplatform
- **Monero (XMR):** RandomX algorithm (ASIC-resistant)komodoplatform
- **Ethereum Classic (ETC):** Ethash algorithm (Ethereum switched to PoS)komodoplatform

---

Scalability Aspects of Blockchain Consensus Protocols

Scalability remains one of the most significant challenges facing blockchain technology today. The issue fundamentally stems from the inherent design of distributed consensus systems.debutinfotech+2

The Blockchain Trilemma

Definition: The blockchain trilemma, coined by Ethereum co-founder Vitalik Buterin, describes the challenge of simultaneously achieving three fundamental properties: **decentralization, security, and scalability**.trezor+2

The core challenge: Blockchain networks can typically achieve only two of these three properties effectively, requiring trade-offs.ledger+2

The Three Pillars Explained

1. Decentralization

Definition: Distribution of power and control among multiple nodes or participants in the network.trustmachines+2

Characteristics:

- No single entity controls the systemtrustwallet+1
- Nodes distributed geographically and organizationallyledger
- Decisions made collectively, not by central authoritytrustmachines
- Resilience against single points of failureledger

Benefits:

- Censorship resistancetrustwallet+1
- Increased trust through transparencytrustwallet
- Reduced risk of corruption or manipulationtrustmachines

Impact on other factors:

- **Security:** More nodes = harder to attack, but coordination becomes challengingtrezor
- **Scalability:** Every node processing every transaction limits speedtrezor+1

2. Security

Definition: Protection against attacks, tampering, and unauthorized access to blockchain data.trezor+2

Key security concerns:

- **51% attacks:** Gaining majority control to manipulate the blockchaingeeksforgeeks
- **Double-spending:** Spending the same digital asset twicetrezor
- **Sybil attacks:** Creating multiple fake identitiesgeeksforgeeks
- **Byzantine attacks:** Malicious nodes sending conflicting informationgeeksforgeeks

Security mechanisms:

- Cryptographic hashinggeeksforgeeks+1

- Digital signaturesijrpr+1
- Consensus protocols requiring supermajority agreementomgwiki+1
- Economic disincentives (cost of attack > potential gain)debutinfotech

Impact on other factors:

- **Decentralization:** Strong security attracts participants, but high security costs may reduce accessibilitytrezor
- **Scalability:** Thorough verification processes slow transaction speedsledger+1

3. Scalability

Definition: The network's ability to handle growing transaction volumes and user numbers without performance degradation.debutinfotech+2

Key metrics:

- **Throughput (TPS):** Transactions processed per secondthesai+2
- **Latency:** Time from transaction submission to confirmationthesai+1
- **Cost:** Transaction fees under various load conditionsdebutinfotech

Impact on other factors:

- **Decentralization:** Scaling often requires more powerful nodes, reducing accessibilityledger+1
- **Security:** Faster processing may introduce vulnerabilitiestrezor+1

Why the Trilemma Exists

Fundamental constraint: In a decentralized system, every node must:treasuryxl+1

1. Receive every transaction
2. Validate every transaction
3. Store the entire blockchain history
4. Participate in consensus

As the network grows:

- More transactions to process → requires more computational power per node
- Larger blockchain → requires more storage
- More nodes → requires more communication and coordination

Result: Either nodes must be more powerful (reducing decentralization) or the network must sacrifice throughput (limiting scalability).trustwallet+2

Scalability Challenges in Detail

1. Limited Throughput

Bitcoin constraints:

- **Block size:** Originally 1 MB, effectively limiting transactions per blockgeeksforgeeks+2
- **Block time:** Average 10 minutes between blocksgeeksforgeeks+2
- **Result:** Maximum ~7-10 transactions per secondwikipedia+3

Ethereum constraints:

- **Block time:** ~12-15 secondshedera
- **Gas limit:** Caps computational work per block
- **Result:** ~12-15 transactions per second (before upgrades)hedera

Comparison to centralized systems:

- **Visa:** ~24,000 TPS capacitytreasuryxl+1
- **PayPal:** Hundreds to thousands of TPShedera
- **Modern databases:** Millions of TPShedera

2. Network Congestion

Cause: When transaction demand exceeds network capacity, transactions queue in the mempool.debutinfotech+1

Consequences:

- **Increased fees:** Users compete by offering higher transaction fees to prioritize their transactions101blockchains+1
- **Delayed confirmations:** Low-fee transactions may wait hours or daysdebutinfotech
- **Poor user experience:** Unpredictable costs and confirmation timesdebutinfotech

Real-world examples:

- **Bitcoin (2017):** Transaction fees exceeded $50 during peak congestionwikipedia
- **Ethereum (2020-2021):** DeFi boom caused gas fees to skyrocket, sometimes exceeding $100 per transactiondebutinfotech

3. Bandwidth Limitations

Challenge: Blockchains require high bandwidth for:101blockchains

- Broadcasting transactions to all nodes
- Distributing blocks throughout the network
- Synchronizing new nodes joining the network101blockchains

Bottleneck effect: The slowest node limits overall network performance.101blockchains

Geographic disparities: Nodes in regions with poor internet infrastructure struggle to keep up.101blockchains

4. Storage Requirements

Growing blockchain size:

- **Bitcoin:** Over 500 GB (as of 2024)bitcoinmagazine
- **Ethereum:** Over 1 TB (full archive node)bitcoinmagazine

Implications:

- **Hardware costs:** Requires increasingly expensive storage devicescoindesk+1
- **Node operation barriers:** Fewer users can afford to run full nodescoindesk+1
- **Centralization risk:** Only well-funded entities can maintain full nodesbitcoinmagazine

5. Synchronization Time

Problem: New nodes must download and validate the entire blockchain history.bitcoinmagazine

Bitcoin example:

- Downloading 500+ GB over standard internet connection: hours to daysbitcoinmagazine
- Validating all historical transactions: additional hours to daysbitcoinmagazine

Impact: High barriers to entry for new participants.bitcoinmagazine

6. Consensus Overhead

PoW challenges:

- Computationally expensive puzzle-solvingpersistent+1
- Only one miner's work per block is useful; rest is wastedpersistent+1
- Difficulty adjustment prevents simple speed increasesrapidinnovation

Multi-round consensus challenges (PBFT, etc.):

- Multiple communication rounds requiredomgwiki+1
- Message complexity grows with number of nodeshalborn
- Practical limit of ~20-100 nodes for acceptable performanceacademia+1

Impact of Scalability Issues on Adoption

1. High Transaction Costs

- Congestion drives fees to economically unviable levelsdebutinfotech
- Microtransactions become impossibledebutinfotech
- Developing nations excluded from participationdebutinfotech

2. Poor User Experience

- Unpredictable confirmation timesdebutinfotech
- Complex fee estimationdebutinfotech
- Lost transactions due to low feesdebutinfotech

3. Limited Use Cases

- Cannot support high-frequency applications (gaming, micropayments)debutinfotech
- DeFi applications become too expensive for average usersdebutinfotech
- Enterprise adoption hindered by performance requirementsdebutinfotech

4. Competitive Disadvantage

- Legacy payment systems offer superior speed and costtreasuryxl+1
- Centralized alternatives more practical for most use casestreasuryxl
- Inhibits mainstream adoptionhedera+1

Relationship Between Consensus Mechanism and Scalability

PoW (Proof of Work):

- **Security:** Very highbusinessinsider+1
- **Decentralization:** High (permissionless)komodoplatform
- **Scalability:** Very poor (7-10 TPS)treasuryxl+1

PoS (Proof of Stake):

- **Security:** Highledger
- **Decentralization:** Moderate to highcyfrin

- **Scalability:** Improved but still limited (15-30 TPS for Ethereum post-merge)hedera

DPoS (Delegated Proof of Stake):

- **Security:** Moderatedevelopcoins
- **Decentralization:** Lower (21-101 validators)developcoins
- **Scalability:** Better (hundreds to thousands of TPS)developcoins

PBFT (Practical Byzantine Fault Tolerance):

- **Security:** High (if <33% nodes malicious)geeksforgeeks+1
- **Decentralization:** Low (permissioned, limited nodes)academia+1
- **Scalability:** Much better (thousands of TPS)pdfs.semanticscholar+1

Scaling Solutions Overview

To address scalability challenges, the blockchain community has developed two main categories of solutions:rapidinnovation+2

Layer 1 (On-Chain) Solutions

Definition: Modifications to the base blockchain protocol itself to improve scalability.debutinfotech+2

Approaches:

1. Increasing Block Size

- **Method:** Raise the block size limit to include more transactions per blocktastycrypto+2
- **Example:** Bitcoin Cash increased block size from 1 MB to 8 MB (later to 32 MB)cointelegraph
- **Advantages:** Immediate throughput increasecointelegraph
- **Disadvantages:** Larger blocks increase storage, bandwidth, and validation requirements, potentially reducing decentralizationcointelegraph+2

2. Consensus Mechanism Upgrades

- **Method:** Transition to more efficient consensus protocolsgemini+2
- **Example:** Ethereum's transition from PoW to PoS ("The Merge")tastycrypto+1
- **Advantages:** Dramatic reduction in energy consumption, faster block timestastycrypto

- **Result:** Improved scalability while maintaining securitygemini+1

3. Sharding

- **Method:** Split the blockchain network into smaller partitions (shards) that process transactions in parallelgemini+2
- **How it works:**
  - Each shard handles a subset of transactions independentlytastycrypto+1
  - Shards communicate via cross-shard protocolsgemini
  - Combined throughput = sum of all shard throughputscointelegraph
- **Example:** Ethereum 2.0 planned sharding implementationgemini+1
- **Advantages:** Theoretically unlimited scalabilitytastycrypto
- **Challenges:** Complex implementation, security concerns at shard boundariesgemini+1

Layer 2 (Off-Chain) Solutions

Definition: Protocols built on top of the base blockchain that handle transactions off the main chain, only settling final results on-chain.rapidinnovation+2

Examples:

- **Lightning Network (Bitcoin):** Payment channels for instant, low-fee transactionsrapidinnovation+1
- **Rollups (Ethereum):** Bundle hundreds of transactions into a single on-chain transactionrapidinnovation+1
- **State Channels:** Participants conduct unlimited off-chain transactions, settling final state on-chainrapidinnovation+1
- **Sidechains:** Independent blockchains linked to the main chaindebutinfotech+1

Advantages:

- Dramatically increased throughputrapidinnovation+1
- Significantly reduced feesrapidinnovation
- Faster finalityrapidinnovation
- Base layer remains unchanged (no hard forks required)debutinfotech+1

Permissioned Blockchains

Permissioned blockchains represent a different approach to distributed ledger technology, prioritizing performance, privacy, and governance over radical decentralization.cfte+3

Definition and Core Characteristics

Definition: A permissioned blockchain is a closed network that restricts access and participation to a select group of authorized, verified users.geeksforgeeks+3

Key Characteristics

1. Access Control

- **Restricted entry:** Only authorized participants can join the networkoracle+2
- **Vetting process:** Users must meet specific criteria or undergo authenticationleewayhertz+1
- **Identity verification:** All participants have known, verified identitiesappinventiv+2
- **Role-based permissions:** Different users may have different levels of accessleewayhertz+1

Example: A supply chain blockchain where only registered suppliers, manufacturers, distributors, and retailers can participate.

2. Controlled Governance

- **Centralized or consortium-based:** Governed by a single organization or group of organizationsmoonpay+2
- **Defined decision-making:** Clear rules for network changes and upgradestransfi+1
- **Faster decisions:** No need for network-wide consensus on governancefreemanlaw+1
- **Regulatory compliance:** Easier to implement and enforce compliance requirementsmoonpay+1

3. No Anonymity

- **Known participants:** Every user's identity is recorded and verifiableosl+2
- **Accountability:** Actions can be traced to specific entitiesoracle+1
- **Chain of custody:** Verifiable audit trails for all transactionsappinventiv+1

4. Controlled Transparency

- **Selective visibility:** Organizations can control who sees what dataosl+2

- **Privacy by design:** Sensitive information can be kept confidentialdebutinfotech+2
- **Customizable disclosure:** Different transparency levels for different participantsleewayhertz+1

5. Performance Optimization

- **Higher throughput:** Limited validators enable faster transaction processing debutinfotech+2
- **Lower latency:** Fewer communication rounds requiredtransfi+1
- **Reduced costs:** Less computational power needed compared to PoWdebutinfotech+1
- **Scalability:** Can handle higher transaction volumesmoonpay+2

6. Energy Efficiency

- **No mining:** Most permissioned blockchains don't use Proof of Workleewayhertz
- **Efficient consensus:** Use voting-based or rotation-based mechanismsprestmit+2
- **Lower environmental impact:** Dramatically reduced energy consumption compared to Bitcoinleewayhertz

Design Goals of Permissioned Blockchains

1. Enterprise-Grade Performance

Objective: Meet the demanding performance requirements of business applications.cfte+2

Requirements:

- **High throughput:** Thousands of transactions per secondpdfs.semanticscholar+2
- **Low latency:** Transaction confirmation in seconds, not minutestransfi+1
- **Predictable performance:** Consistent response times under varying loadsoracle+1
- **Deterministic finality:** Immediate irreversibility of confirmed transactionsgeeksforgeeks+1

Why important: Businesses cannot tolerate the unpredictable confirmation times and high fees of public blockchains.transfi+1

2. Privacy and Confidentiality

Objective: Protect sensitive business information while maintaining blockchain benefits.cfte+3

Approaches:

- **Selective disclosure:** Transactions visible only to relevant partiesappinventiv+1
- **Private data collections:** Off-chain storage of sensitive data with on-chain hashesleewayhertz
- **Confidential transactions:** Encrypted transaction amounts and participantsdebutinfotech+1
- **Zero-knowledge proofs:** Prove validity without revealing detailsrapidinnovation+1

Use cases:

- Healthcare: Patient data privacyleewayhertz
- Finance: Proprietary trading informationcfte+1
- Supply chain: Competitive pricing informationleewayhertz

3. Regulatory Compliance

Objective: Meet legal and regulatory requirements for identity, auditing, and data handling.techtarget+2

Features:

- **Know Your Customer (KYC):** Identity verification for all participantstechtarget+1
- **Anti-Money Laundering (AML):** Transaction monitoring and reportingtechtarget+1
- **Data protection:** GDPR and other privacy regulation compliancemoonpay+1
- **Audit trails:** Complete, verifiable history of all actionsoracle+1
- **Right to be forgotten:** Ability to remove personal data (where legally required)moonpay

Contrast with public blockchains: Permissionless networks' anonymity makes regulatory compliance extremely challenging.techtarget

4. Governance and Control

Objective: Enable efficient decision-making and network management.freemanlaw+3

Mechanisms:

- **Defined governance structure:** Clear roles and responsibilitiesfreemanlaw+1
- **Upgrade procedures:** Streamlined process for protocol updatesmoonpay
- **Access management:** Authority to grant/revoke permissionsgeeksforgeeks+1
- **Dispute resolution:** Established procedures for handling conflictscfte

Benefits:

- Quick adaptation to changing business needstransfi+1
- Coordinated responses to security issuesmoonpay
- Aligned incentives among participantscfte

5. Interoperability and Integration

Objective: Seamlessly integrate with existing enterprise systems and other blockchains.debutinfotech+1

Requirements:

- **API compatibility:** Standard interfaces for system integrationleewayhertz
- **Legacy system connection:** Bridges to existing databases and applicationsdebutinfotech
- **Cross-chain communication:** Ability to interact with other blockchainsleewayhertz
- **Flexible data models:** Support for diverse business logicappinventiv+1

6. Customization and Flexibility

Objective: Tailor the blockchain to specific organizational needs.appinventiv+2

Customizable elements:

- **Consensus mechanism:** Choose based on trust model and performance needspdfs.semanticscholar+1
- **Smart contract languages:** Support for preferred development environmentsleewayhertz
- **Permission models:** Define granular access controlsappinventiv
- **Data structures:** Optimize for specific use casesdebutinfotech+1

7. Cost Efficiency

Objective: Minimize operational costs while delivering blockchain benefits.debutinfotech+1

Cost factors:

- **Infrastructure:** Fewer, more powerful nodes vs. many lightweight nodestransfi
- **Energy:** Efficient consensus vs. energy-intensive miningleewayhertz
- **Transaction fees:** Minimal or zero fees vs. market-driven feestransfi+1
- **Maintenance:** Centralized management vs. distributed coordinationtransfi

8. Security with Efficiency

Objective: Maintain strong security without sacrificing performance.pdfs.semanticscholar+1

Approach:

- **Byzantine Fault Tolerance:** Tolerate malicious nodes without expensive miningacademia+1
- **Cryptographic protection:** Strong encryption and signaturesleewayhertz
- **Access controls:** Prevent unauthorized participationappinventiv+1
- **Trust assumptions:** Leverage known participant identitiesacademia+1

Permissioned vs. Permissionless: Comparative Analysis

Aspect	Permissionless	Permissioned
Access	Open to anyonemoonpay+1	Restricted to authorized participantsmoonpay+1
Identity	Anonymous/pseudonymousgeeksforgeeks+1	Known, verified identitiesgeeksforgeeks+1
Governance	Decentralized, community-drivenmoonpay+1	Centralized or consortium-basedmoonpay+1
Transparency	Fully transparent public ledgermoonpay+1	Controlled, selective transparencymoonpay+1
Transaction Speed	Slow (seconds to minutes)hedera+1	Fast (milliseconds to seconds)academia+1
Throughput	Low (7-30 TPS)hedera+1	High (hundreds to thousands TPS)academia+1

Energy Consumption	High (PoW) to moderate (PoS)komodoplatform+1	Low (efficient consensus)leewayhertz
Consensus	PoW, PoS, etc.komodoplatform	PBFT, Raft, Paxos, etc.academia+1
Scalability	Limitedhedera+1	Betterleewayhertz+1
Finality	Probabilistic (PoW) or fast (PoS)persistent	Immediate (deterministic)persistent+1
Security Model	Cryptoeconomic incentiveskomodoplatform+1	Access controls + BFTleewayhertz+1
Use Cases	Cryptocurrencies, DeFi, public dAppsmoonpay	Enterprise, supply chain, consortiumscfte+1
Regulatory Compliance	Challengingtechtarget	Easiermoonpay+1
Cost	Transaction fees can be highdebutinfotech	Lower operational costsdebutinfotech+1

Advantages of Permissioned Blockchains

1. Superior Performance

- **High throughput:** Can process thousands of transactions per secondpdfs.semanticscholar+2
- **Low latency:** Near-instant transaction confirmationacademia+1
- **Predictability:** Consistent performance characteristicsleewayhertz

2. Enhanced Privacy

- **Data confidentiality:** Sensitive information protectedmoonpay+2
- **Selective sharing:** Information visible only to authorized partiesappinventiv+1
- **Regulatory alignment:** Meets data protection requirementsmoonpay+1

3. Energy Efficiency

- **No mining:** Eliminates wasteful computational workleewayhertz
- **Sustainable:** Dramatically lower environmental impactleewayhertz

4. Governance Efficiency

- **Rapid decision-making:** No need for network-wide consensusfreemanlaw+1
- **Clear accountability:** Known participants and defined rolesappinventiv
- **Coordinated upgrades:** Streamlined protocol evolutionmoonpay

5. Regulatory Compliance

- **Identity verification:** Built-in KYC/AMLoracle+1
- **Audit support:** Complete traceabilityoracle+1
- **Legal clarity:** Easier to navigate regulatory landscapemoonpay+1

6. Cost Effectiveness

- **Lower infrastructure costs:** Fewer nodes requiredtransfi+1
- **Minimal transaction fees:** Often negligible or zerodebutinfotech+1
- **Operational efficiency:** Reduced overheaddebutinfotech+1

Disadvantages of Permissioned Blockchains

1. Centralization Concerns

- **Single point of failure:** Governing authority could be compromisedfreemanlaw
- **Censorship risk:** Controlling entities can block transactionstechtarget
- **Trust requirement:** Must trust the permissioning authorityfreemanlaw+1

2. Limited Decentralization

- **Fewer nodes:** Concentrated among consortium membersfreemanlaw+1
- **Geographic concentration:** Often limited to specific regions or organizationscfte
- **Collusion risk:** Small group of validators could cooperate maliciouslydevelopcoins+1

3. Reduced Transparency

- **Selective visibility:** Not all participants can see all dataosl+1
- **Accountability challenges:** Limited public oversighttechtarget
- **Trust assumptions:** Users must trust operators' honestyfreemanlaw

4. Vendor Lock-In

- **Proprietary solutions:** May be tied to specific platformscfte
- **Migration challenges:** Difficult to switch providerscfte

- **Limited interoperability:** May not work with other systemsleewayhertz

5. Barriers to Entry

- **Permission requirements:** Cannot freely joingeeksforgeeks+1
- **Exclusivity:** May lock out potential participantscfte
- **Power concentration:** Early members have advantagescfte

Use Cases for Permissioned Blockchains

1. Supply Chain Management

- Track products from origin to consumerleewayhertz
- Verify authenticity and provenanceappinventiv+1
- Share information with authorized partners onlyleewayhertz
- **Example:** IBM Food Trust, Walmart's supply chain tracking

2. Financial Services

- Interbank settlements and clearingcfte+1
- Trade finance and letters of creditcfte
- Securities trading and settlementcfte
- **Example:** R3 Corda, JPM Coin

3. Healthcare

- Secure patient records sharingleewayhertz
- Drug traceability and anti-counterfeitingleewayhertz
- Insurance claims processingleewayhertz
- **Example:** MedRec, Guardtime

4. Government and Public Sector

- Land registry and property recordscfte
- Identity management systemscfte
- Voting systemscfte

5. Enterprise Consortiums

- Industry-specific collaborationscfte+1
- Shared infrastructure among competitorscfte
- B2B transaction networksleewayhertz

Consensus Protocols for Permissioned Blockchains

Permissioned blockchains typically employ consensus mechanisms optimized for known participants and higher performance requirements. These protocols assume partial trust among participants and focus on efficiency rather than incentivizing unknown miners.drops.dagstuhl+2

1. Practical Byzantine Fault Tolerance (PBFT)

Origin: Introduced by Barbara Liskov and Miguel Castro in 1999.geeksforgeeks+2

Design Purpose: Achieve consensus efficiently in asynchronous systems while tolerating Byzantine faults (malicious or arbitrarily faulty nodes).omgwiki+2

Core Principles

Fault tolerance threshold:

$$n > 3f + 1$$

Where:

- n= total number of nodes
- f= maximum number of faulty nodes

Interpretation: PBFT can tolerate up to f malicious nodes if there are at least $3f + 1$total nodes. This means fewer than 33.3% of nodes can be Byzantine (malicious) for the system to function correctly.jatit+3

Example: With 10 total nodes, PBFT can tolerate up to 3 malicious nodes.

PBFT Process: Four Phases

Phase 0: Request

1. A client sends a transaction request to the primary node (leader).jatit+1
2. The request includes operation details, timestamp, and client signature.omgwiki

Phase 1: Pre-Prepare

1. The primary node receives the request and validates it.jatit+1
2. The primary assigns a sequence number to the request.omgwiki
3. The primary broadcasts a PRE-PREPARE message to all backup (replica) nodes.jatit+1
4. The message contains: view number, sequence number, request digest, and primary's signature.omgwiki

Phase 2: Prepare

1. Each backup node validates the PRE-PREPARE message:halborn+1
   - Verifies signatures
   - Checks sequence number validity
   - Ensures it hasn't accepted conflicting messages
2. If valid, the node broadcasts a PREPARE message to all other nodes.jatit+1
3. Each node collects PREPARE messages from other nodes.omgwiki
4. A node enters the "prepared" state when it has received:halborn+1
   - The original PRE-PREPARE message
   - PREPARE messages from at least $2f$ different backup nodes (including itself)

Prepare certificate: The collection of messages proving the prepared state.omgwiki

Phase 3: Commit

1. Once in the prepared state, each node broadcasts a COMMIT message to all nodes.jatit+1
2. Nodes collect COMMIT messages.omgwiki
3. A node enters the "committed" state when it has received:halborn+1
   - Its prepare certificate
   - COMMIT messages from at least $2f + 1$different nodes (including itself)

Commit certificate: The collection of messages proving the committed state.omgwiki

Phase 4: Reply

1. Once committed, each node executes the requested operation.jatit+1
2. Nodes send reply messages back to the client.omgwiki
3. The client waits for $f + 1$matching replies from different nodes.omgwiki
4. If matching replies received, the result is considered valid and final.omgwiki

View Changes (Leader Replacement)

When needed: If the primary node fails, is too slow, or acts maliciously.halborn+1

How it works:

1. Backup nodes detect primary failure (via timeout).halborn+1
2. A backup initiates a view change by broadcasting a VIEW-CHANGE message.omgwiki
3. When $2f + 1$nodes agree, they move to a new view with a new primary.halborn+1
4. The new primary is selected in round-robin fashion.omgwiki
5. The new primary ensures all nodes are synchronized before accepting new requests.omgwiki

Advantages of PBFT

1. **High throughput:** No computationally expensive miningsciencedirect+2
2. **Low latency:** Typically milliseconds to seconds for finalitysciencedirect+1
3. **Energy efficient:** Minimal computational requirementssciencedirect+1
4. **Deterministic finality:** Once committed, transactions are immediately finalgeeksforgeeks+1
5. **Proven security:** Mathematically proven to tolerate up to fByzantine faultsgeeksforgeeks+1

Disadvantages of PBFT

1. **Scalability limitations:** Communication complexity $O(n^2)$limits practical size to ~20-100 nodespdfs.semanticscholar+2
2. **Known participants required:** Best suited for permissioned networkssciencedirect+1
3. **View change overhead:** Leader changes can be expensivehalborn+1
4. **Sybil attack vulnerability:** If used in permissionless settings (not recommended)sciencedirect

Variants of PBFT

- **dBFT (Delegated Byzantine Fault Tolerance):** Used by NEOpdfs.semanticscholar
- **Federated Byzantine Agreement (FBA):** Used by Stellarpdfs.semanticscholar
- **Istanbul BFT (IBFT):** Used in Ethereum-based permissioned chainskaleido
- **Tendermint:** BFT consensus for blockchain applicationsgeeksforgeeks

Use Cases

- **Hyperledger Fabric:** Enterprise blockchain platformdrops.dagstuhl+1

- **Zilliqa:** High-throughput blockchainsciencedirect
- **NEO:** Smart contract platform (dBFT variant)pdfs.semanticscholar

---

2. Paxos Consensus Algorithm

Origin: Developed by Leslie Lamport in 1989, first published in 1998 ("The Part-Time Parliament" paper).studocu+2

Design Purpose: Choose a single value in a distributed system under crash faults or network failures (not Byzantine faults).geeksforgeeks+2

Core Concept

Objective: Achieve consensus on a single value among distributed nodes, even when some nodes crash or messages are lost.sciencedirect+2

Trust model: Assumes nodes are honest but may fail (crash fault tolerance, not Byzantine fault tolerance).geeksforgeeks+2

Types of Nodes in Paxos

1. Proposers

- Propose values for consensusgeeksforgeeks+2
- Initiate the consensus processstudocu
- Can be multiple proposers competing simultaneouslygeeksforgeeks+1

2. Acceptors

- Vote on proposed valuesgeeksforgeeks+2
- Store accepted proposalsgeeksforgeeks
- Form the quorum for consensussciencedirect+1

3. Learners

- Learn the chosen value once consensus is reachedstudocu+2
- Do not participate in votingsciencedirect
- Typically, all nodes are learnersstudocu

Note: In practice, a single node can play multiple roles.geeksforgeeks+1

Paxos Process: Three Main Phases

Phase 1: Prepare Phase

1. A proposer selects a unique proposal number n(higher than any previously used).geeksforgeeks+2

2. The proposer sends a PREPARE(n) message to a majority of acceptors.sciencedirect+1
3. Each acceptor receives PREPARE(n):geeksforgeeks+2
   - If *n*is greater than any previously seen proposal number, the acceptor:
     - Promises not to accept any proposals with numbers less than *n*
     - Responds with a PROMISE message
     - Includes the highest-numbered proposal it has already accepted (if any)
   - If *n*is not greater, the acceptor ignores or rejects the request

Phase 2: Promise Phase

1. The proposer waits for PROMISE responses from a majority of acceptors.geeksforgeeks+2
2. If a majority responds:sciencedirect+1
   - The proposer checks if any acceptor included a previously accepted proposal in their PROMISE
   - If yes, the proposer must use the value from the highest-numbered accepted proposal
   - If no, the proposer can choose its own value
3. If no majority responds within a timeout, the proposer restarts with a higher proposal number.sciencedirect

Phase 3: Accept Phase

1. The proposer sends an ACCEPT(n, value) message to acceptors.geeksforgeeks+2
2. Each acceptor receives ACCEPT(n, value):geeksforgeeks+2
   - If the acceptor has not promised to reject proposals numbered *n*(i.e., hasn't received a higher proposal number since promising), it:
     - Accepts the proposal
     - Records *n*and *value*
     - Responds with ACCEPTED(n, value)
   - Otherwise, it rejects the proposal
3. Once a majority of acceptors accept a proposal, the value is chosen.geeksforgeeks+1

Phase 4: Learn Phase (Informational)

1. Acceptors inform learners about the accepted value.geeksforgeeks+1
2. Once a learner receives ACCEPTED messages from a majority of acceptors for the same $(n, value)$, it knows consensus has been reached.sciencedirect+1

Example Scenario

Setup: 5 nodes (A, B, C, D, E); need majority = 3 nodes

Scenario:

1. **Proposer 1** wants to propose value V1
   - Sends PREPARE(10) to all acceptors
   - Receives PROMISE from A, B, C (majority achieved)
   - None have accepted any previous proposals
2. **Proposer 1** proceeds to accept phase
   - Sends ACCEPT(10, V1) to all acceptors
   - A, B, C accept and respond ACCEPTED(10, V1)
   - Consensus reached: value V1 is chosen

Concurrent proposal scenario:

1. **Proposer 1** sends PREPARE(10)
2. **Proposer 2** sends PREPARE(15) (higher number)
3. Acceptors receive both:
   - First promise to Proposer 1 (number 10)
   - Then promise to Proposer 2 (number 15), superseding the first promise
4. Proposer 1's ACCEPT(10, V1) is now rejected (acceptors promised not to accept < 15)
5. Proposer 2's ACCEPT(15, V2) succeeds
6. Result: V2 is chosen

Unique Proposal Numbers

Critical requirement: Proposal numbers must be unique across all proposers.martinfowler+1

Common strategy:

- Proposer ID concatenated with a counter

- Example: Proposer 3's proposals: 3.1, 3.2, 3.3, ...
- Global ordering: 1.1 < 2.1 < 1.2 < 3.1 < 2.2 < ...

Advantages of Paxos

1. **Proven correctness:** Mathematically proven to achieve consensus safelymartinfowler+2
2. **Robust:** Handles network delays, message loss, and node crashesgeeksforgeeks+1
3. **No single point of failure:** Works as long as a majority of nodes are operationalmartinfowler+1
4. **Flexible:** Supports dynamic changes in node membershipsciencedirect

Disadvantages of Paxos

1. **Complexity:** Notoriously difficult to understand and implement correctlystudocu+2
2. **Livelock potential:** Competing proposers can prevent consensus indefinitelymartinfowler+1
3. **Performance:** Multiple round trips can increase latencygeeksforgeeks
4. **Not Byzantine fault tolerant:** Cannot handle malicious nodesgeeksforgeeks+2
5. **Limited scalability:** Performance degrades with many nodesacademia+1

Optimizations and Variants

- **Multi-Paxos:** Optimizes for multiple consecutive consensus decisions by electing a stable leadermartinfowler
- **Fast Paxos:** Reduces latency in the common casesciencedirect
- **Egalitarian Paxos (EPaxos):** Improves throughput and latencysciencedirect

Use Cases

- **Google Chubby:** Distributed lock servicestudocu+1
- **Apache ZooKeeper:** Coordination service (uses Zab, inspired by Paxos)sciencedirect
- **Permissioned blockchains:** Where crash fault tolerance is sufficientdrops.dagstuhl+2

---

3. Raft Consensus Algorithm

Origin: Designed by Diego Ongaro and John Ousterhout at Stanford University in 2014.scribd+2

Design Purpose: Provide a more understandable alternative to Paxos while achieving the same fault tolerance guarantees.geeksforgeeks+3

Key principle: "Understandability above all else"—Raft prioritizes being easy to understand and implement.wikipedia+1

Core Concepts

Server states: Every node in a Raft cluster exists in one of three states:arorashu.github+2

1. **Leader:** Handles all client requests, manages log replicationarorashu.github+2
2. **Follower:** Passive; responds to leader and candidate requestsgeeksforgeeks+2
3. **Candidate:** Temporarily exists during leader electionwikipedia+2

Term numbers: Time is divided into arbitrary-length terms, each identified by a monotonically increasing term number.linkedin+2

- Each term begins with an electiongeeksforgeeks+1
- If election succeeds, one leader for that termwikipedia+1
- If election fails (split vote), term ends with no leader, and a new term beginsgeeksforgeeks+1

Purpose of term numbers:

- Detect stale informationlinkedin+1
- Ensure only one leader per termwikipedia+1
- Coordinate electionslinkedin

Raft Process: Three Main Components

Component 1: Leader Election

Triggering an election:

1. Initially, all nodes start as followers.arorashu.github+2
2. Followers expect periodic heartbeats from the leader.arorashu.github+2
3. If a follower receives no heartbeat within the **election timeout** (randomized, typically 150-300ms), it assumes the leader has failed.arorashu.github+2

Election process:

1. **Become candidate:**

- Follower increments its current term numberlinkedin+2
- Transitions to candidate stategeeksforgeeks+2
- Votes for itselfwikipedia+2

2. **Request votes:**
    - Candidate sends RequestVote RPCs to all other nodes in parallelarorashu.github+2
    - Message includes: candidate's term, candidate ID, last log index, last log termlinkedin+1
3. **Voting rules:**
    - Each node votes for at most one candidate per term, first-come-first-servedgeeksforgeeks+2
    - Node grants vote only if:wikipedia+2
        - Candidate's term ≥ voter's term
        - Voter hasn't already voted in this term
        - Candidate's log is at least as up-to-date as voter's log
4. **Election outcomes (three possibilities):**

a) Candidate wins:

- Receives votes from a majority of nodesarorashu.github+2
- Becomes leader for the current termwikipedia+1
- Immediately sends heartbeat messages to all nodes to establish authoritygeeksforgeeks+2

b) Another leader emerges:

- Candidate receives heartbeat from another node claiming to be leaderlinkedin+2
- If the leader's term ≥ candidate's term, candidate accepts the leader and returns to follower statelinkedin+2
- If leader's term < candidate's term, candidate rejects the message and continues as candidatearorashu.github+1

c) Split vote (no winner):

- No candidate receives majority (e.g., multiple candidates split the votes)geeksforgeeks+2

  - Candidates time out and start a new election by incrementing term and restarting the processwikipedia+2
  - **Split vote prevention:** Randomized election timeouts (e.g., 150-300ms) ensure candidates don't time out simultaneouslylinkedin+3

Heartbeat mechanism:

- Leader sends empty AppendEntries RPCs (heartbeats) to all followers at regular intervalsarorashu.github+2
- Prevents followers from starting electionsgeeksforgeeks+1
- Serves as a "keep-alive" signallinkedin+1

Component 2: Log Replication

Purpose: Ensure all nodes maintain identical, ordered logs of commands.stanford+2

Process:

1. **Client request:**
   - Client sends command to the leaderwikipedia+1
   - If request sent to follower, follower redirects to leadergeeksforgeeks+1
2. **Leader appends to log:**
   - Leader appends the command to its own log as a new entrystanford+2
   - Entry includes: command, term number, log indexgeeksforgeeks
3. **Leader replicates:**
   - Leader sends AppendEntries RPC to all followers in parallelstanford+2
   - Message includes: log entries, leader's term, previous log index, previous log term, leader's commit indexgeeksforgeeks
4. **Followers respond:**
   - Each follower receives AppendEntries:stanford+2
     - Checks if its log matches the leader's (via previous log index and term)
     - If match, appends new entries and responds with success
     - If mismatch, responds with failure
   - **Log matching:** Ensures followers' logs are consistent with leader's logstanford+1
5. **Leader commits:**

    - Once a majority of followers acknowledge replication, the leader commits the entrystanford+2
    - Committed entries are applied to the leader's state machinewikipedia+1
    - Leader includes the new commit index in the next heartbeatstanford+1

6. **Followers commit:**
    - Followers learn about committed entries from the leader's commit indexstanford+1
    - Followers apply committed entries to their state machines in orderwikipedia+1

Safety guarantee: If a log entry is committed, it will eventually be executed by all available state machines.stanford+2

Component 3: Safety

Election restriction:

- A candidate cannot win an election unless its log contains all committed entries.linkedin+2
- Voters deny votes to candidates with less complete logs.linkedin+1

Log matching property:

- If two logs contain an entry with the same index and term, then:stanford+2
    - The logs contain identical entries up to that index
    - The entries are identical

Leader completeness:

- Once a log entry is committed, it will be present in the logs of all future leaders.stanford+1

State machine safety:

- If a node has applied a log entry at a given index, no other node will apply a different command at that index.wikipedia+1

Advantages of Raft

1. **Understandability:** Much easier to grasp than Paxosscribd+3
2. **Implementation simplicity:** Clearer specification makes implementation less error-pronegeeksforgeeks+1
3. **Strong leadership:** Single leader simplifies log managementlinkedin+1

4. **Deterministic finality:** Committed entries are immediately finalgeeksforgeeks+1
5. **Good performance:** Suitable for permissioned networks with moderate node countsscribd+1

Disadvantages of Raft

1. **Leader bottleneck:** All client requests go through the leaderlinkedin+1
2. **Not Byzantine fault tolerant:** Assumes honest (non-malicious) nodesacademia+2
3. **Limited scalability:** Performance degrades with many nodesacademia+1
4. **Network partitions:** Can result in temporary unavailability during splitswikipedia

Comparison: Raft vs. Paxos

Aspect	**Paxos**	**Raft**
Understandability	Very complexgeeksforgeeks+1	Much simplergeeksforgeeks+1
Leader	No fixed leader (Multi-Paxos has optional leader)sciencedirect+1	Strong, stable leader requiredgeeksforgeeks+1
Log structure	Not explicitly definedsciencedirect	Clearly defined, ordered loggeeksforgeeks+1
Term/View	View changes complexsciencedirect	Term-based, straightforwardgeeksforgeeks+1
Implementation	Difficult, many edge casesgeeksforgeeks+1	Easier, clearer specificationgeeksforgeeks+1
Performance	Similargeeksforgeeks	Similargeeksforgeeks
Fault tolerance	Crash faultsgeeksforgeeks+1	Crash faultsgeeksforgeeks+1

Use Cases

- **etcd:** Distributed key-value store used by Kubernetesscribd+1
- **Consul:** Service discovery and configurationwikipedia
- **Permissioned blockchains:** Where crash fault tolerance is sufficient and simplicity is valuedkaleido+2
- **CockroachDB:** Distributed SQL databasewikipedia

Summary of Key Concepts

This unit covered essential aspects of consensus protocols and blockchain scalability:

1. Requirements for Consensus Protocols: Explored the fundamental requirements (agreement, validity, termination, fault tolerance, integrity) and performance criteria (throughput, latency, scalability, energy consumption, security) that consensus mechanisms must satisfy.

2. Proof of Work (PoW): Detailed examination of Bitcoin's consensus mechanism, including the mining process, difficulty adjustment, security properties (double-spending prevention, 51% attack resistance), and trade-offs between security and scalability/energy consumption.

3. Scalability Challenges: Analyzed the blockchain trilemma (decentralization, security, scalability), examined specific limitations (limited throughput, network congestion, storage requirements), and introduced Layer 1 and Layer 2 scaling solutions.

4. Permissioned Blockchains: Covered design goals (performance, privacy, regulatory compliance, governance), key characteristics (access control, known identities, controlled transparency), and comparative advantages over permissionless networks for enterprise use cases.

5. Consensus Protocols for Permissioned Blockchains: In-depth study of three major protocols:

- **PBFT:** Byzantine fault tolerance with $3f + 1$nodes, four-phase consensus (request, pre-prepare, prepare, commit), suitable for up to ~100 nodes
- **Paxos:** Crash fault tolerance, three-phase process (prepare, promise, accept), mathematically proven but complex to implement
- **Raft:** Crash fault tolerance with emphasis on understandability, strong leader model, three main components (leader election, log replication, safety)

Practice Questions

Question 1: Explain the blockchain trilemma and discuss why achieving all three properties (decentralization, security, and scalability) simultaneously is challenging. Provide examples of how different consensus mechanisms make trade-offs among these properties.

Question 2: Compare and contrast Practical Byzantine Fault Tolerance (PBFT) and Raft consensus algorithms. Discuss their fault tolerance models, communication patterns, performance characteristics, and suitable use cases. Under what circumstances would you choose one over the other for a permissioned blockchain implementation?

Diploma Wallah

Made with 💗 by Sangam