Lesson 2a: DLT Fundamentals

Module A: Blockchain Foundations

BSc Blockchain & Cryptocurrency

University Course

2025

The Coordination Problem

[COMIC: Byzantine generals disagreement]

Placeholder for comic illustration:

Four generals around a city. General A sends "ATTACK"

but traitor D tells B "retreat" and C "attack".

Caption: "When you can't trust the messenger..."

The Setup

- Generals must coordinate attack
- Messages may be lies
- Some generals are traitors

The Question

- How do loyal generals agree?
- How many traitors tolerable?
- Sound familiar? (blockchain nodes)

This 1982 thought experiment defines the core problem that DLT solves

Learning Objectives

By the end of this lesson, you will be able to:

1. Define Distributed Ledger Technology (DLT) and distinguish it from blockchain
2. Explain the Byzantine Generals Problem and its relevance to consensus
3. Compare network topologies: centralized, decentralized, and distributed

Prerequisites: L01 - What is Blockchain?

This is Part 1 of 2 on DLT concepts. Part 2 (L02b) covers block structure and node types.

Building on L01: What is Blockchain?

The Problem: How do we reach agreement in unreliable networks?

Part 1/2: Distributed Ledger Technology (Fundamentals)

The Challenge

Financial systems require all nodes to agree on the state of transactions, but some nodes may be offline, compromised, or deliberately malicious. How can we build consensus when we cannot trust every participant?

Why It Matters

- Without Byzantine fault tolerance (the ability to reach agreement despite some malicious participants), a single malicious node can cause payment systems to fail
- Traditional databases rely on trusted administrators; DLT must work in adversarial environments

What We Need

- Trustless verification mechanism
- Consensus protocols that tolerate up to $< 1/3$ malicious actors
- Network architectures that prevent single points of failure

The Cryptoeconomics Question

Coordinating without trusted intermediaries

Today's lesson: How Distributed Ledger Technology addresses this challenge

Continued

What is Distributed Ledger Technology?

DLT Definition

A **Distributed Ledger Technology (DLT)** is a database architecture where data is synchronized, replicated, and shared across multiple nodes in a network, without a central administrator.

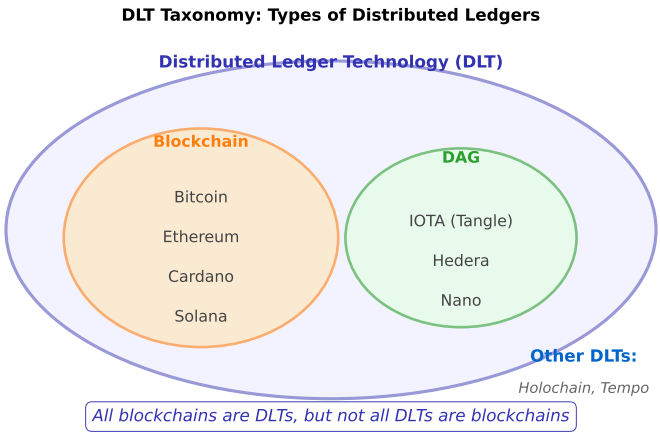
Key Characteristics:

- **Distributed**: Data stored across many nodes (no single source of truth)
- **Synchronized**: All nodes maintain consistent state through consensus
- **Replicated**: Each node has a complete or partial copy of the ledger
- **Shared**: Multiple parties can read/write to the ledger
- **Consensus-driven**: Agreement mechanism validates changes

All blockchains are DLTs, but not all DLTs are blockchains

DLT solves distributed agreement by replicating state across nodes without requiring a trusted coordinator

What Are the Different Types of DLT?



Key Insight: Blockchain is a subset of DLT. Other approaches include DAGs (IOTA), Holochain, and Tempo (Radix).

What Are the Different Types of DLT? – visual summary

How Does Blockchain Differ from Other DLTs?

Blockchain

- Data organized in sequential blocks
- Blocks linked via cryptographic hashes
- Linear chain structure
- Append-only (immutable history)
- Examples: Bitcoin, Ethereum

Directed Acyclic Graph (DAG)

- Transactions reference previous transactions
- No blocks, no miners
- Parallel processing
- Examples: IOTA (Tangle), Hedera Hashgraph

Different DLT architectures optimize for different trade-offs

Holochain

- Agent-centric (not data-centric)
- Each agent maintains own chain
- Distributed Hash Table (DHT)
- No global consensus required

Tempo (Radix)

- Sharded state machine
- Logical clocks for ordering
- Scalable to millions of TPS
- Hybrid consensus

Compare the approaches shown above

What's the Difference Between Shared and Distributed Ledgers?

Shared Ledger (Traditional)

- Multiple parties access same database
- Central authority controls access
- Single version of truth
- Fast, but trust required

Example: Banking consortium database

Challenges:

- Single point of failure
- Administrator can alter records
- Reconciliation between systems

Distributed Ledger (DLT)

- Each party has own copy
- No central authority
- Consensus creates truth
- Slower, but trustless

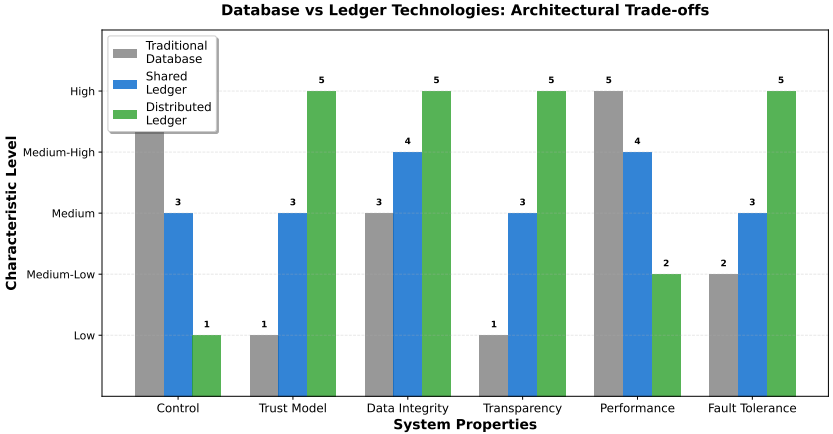
Example: Public blockchain

Benefits:

- No single point of failure
- Transparent, auditable history
- No reconciliation needed

Shared ledgers trust administrators; distributed ledgers achieve consensus through Byzantine fault tolerance protocols

How Do Databases Differ from Distributed Ledgers?



Interpretation: Traditional DBs optimize for performance and central control. Shared Ledgers balance multiple parties' needs. Distributed Ledgers prioritize trustlessness and fault tolerance over speed.

Key Insight: Traditional databases optimize for performance; DLTs trade speed for trustlessness and fault tolerance.

How Do Databases Differ from Distributed Ledgers? – visual summary

The Byzantine Generals Problem (1982)

Historical Context

Leslie Lamport, Robert Shostak, and Marshall Pease formulated this thought experiment to describe the challenge of achieving consensus in distributed systems with potentially faulty or malicious actors.

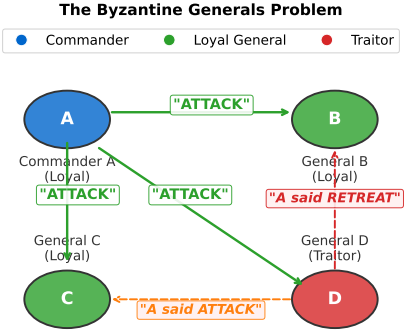
The Scenario:

- Byzantine army surrounds an enemy city
- Army divided into divisions, each led by a general
- Generals must coordinate: **ATTACK** or **RETREAT**
- Communication only via messengers
- Some generals may be traitors (Byzantine faults)

Challenge: How can loyal generals reach consensus when traitors send conflicting messages?

Key point: The Scenario

How Do Byzantine Generals Illustrate the Consensus Challenge?



Problem: How can loyal generals reach consensus when traitors send conflicting messages?

Problem: Traitor D sends conflicting messages to B and C, causing coordination failure.

How Does the Byzantine Generals Problem Work?

Setup: 4 Generals (A, B, C, D), 1 is a traitor

Round 1 - Commander A sends orders:

- To General B: "ATTACK"
- To General C: "ATTACK"
- To General D (traitor): "ATTACK"

Round 2 - Traitor D sends conflicting messages:

- To General B: "A said RETREAT" (lie)
- To General C: "A said ATTACK" (truth)

Result Without Byzantine Fault Tolerance:

- General B thinks majority said RETREAT (withdraws alone)
- General C thinks majority said ATTACK (attacks alone)
- Army coordination fails, both divisions defeated

Key point: Setup

Problem Check-In: Distributed Ledger Technology

Recall Our Problem

How do we reach agreement in unreliable networks?

What We've Learned So Far

- Distributed systems face the Byzantine Generals Problem
- Consensus protocols enable agreement despite faulty nodes
- Both concepts address trust in unreliable networks

Still to Address

- How do we achieve finality? What are the trade-offs?
- Can we tolerate more than 1/3 Byzantine nodes?

Think About

- Based on what you've seen, how would *you* solve this problem?
- What trade-offs do you expect?

Pause and reflect: How does what we've learned so far address "How do we reach agreement in unreliable..."?

How Many Honest Nodes Are Needed for Byzantine Fault Tolerance?

Formal Result

Consensus is achievable if and only if at least $\frac{2}{3}$ of the actors are honest. Formally: $n \geq 3f + 1$ where n = total nodes, f = faulty nodes.

BFT Solution Requirements:

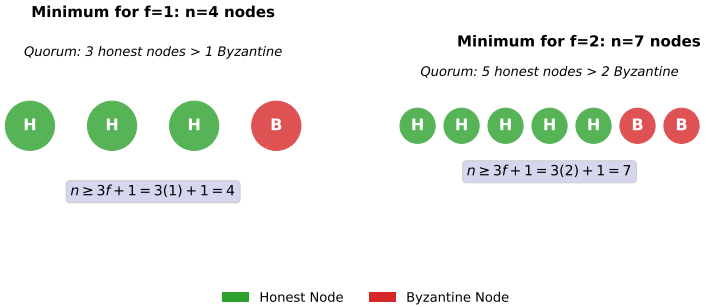
1. **Agreement**: All honest nodes decide on the same value
2. **Validity**: If all honest nodes propose value v, then v is decided
3. **Termination**: All honest nodes eventually decide

Blockchain Context:

- Bitcoin's Proof-of-Work solves Byzantine Generals Problem
- Nakamoto Consensus: Longest chain = majority decision
- Tolerates up to 49% malicious hash power (in practice, 51% attack possible)
- Practical BFT (PBFT) used in permissioned blockchains (Hyperledger)

Byzantine fault tolerance provides mathematical guarantee: consensus with $< 1/3$ malicious nodes, critical for trustless agreement

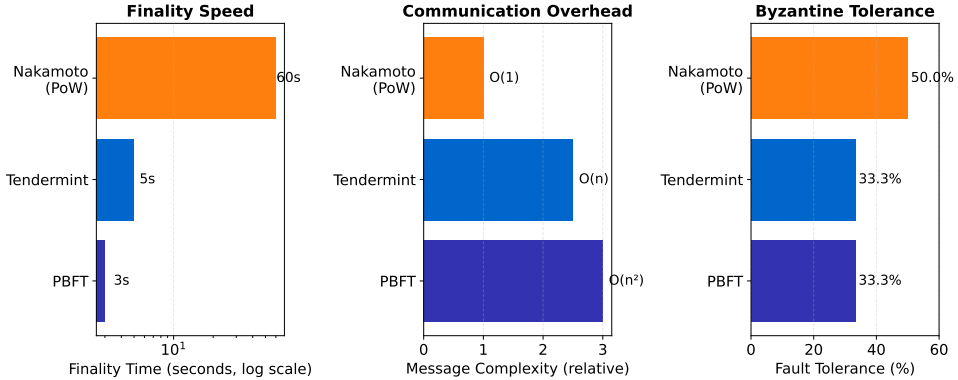
Why Do We Need n $\geq$ 3f+1 Nodes for Byzantine Fault Tolerance?



Key Insight: Need at least 3f+1 total nodes to tolerate f Byzantine failures - quorum of honest nodes must outnumber Byzantine nodes.

Why Do We Need n 3f+1 Nodes for Byzantine Fault Tolerance? – visual summary

How Do Different Consensus Protocols Compare?



Trade-off: PBFT offers fast finality but high message overhead; Nakamoto consensus is simpler but slower.

What Types of Faults Can Occur in Distributed Systems?

Crash Faults (Simpler)

- Node fails and stops responding
- Predictable behavior
- Detectable by timeout
- Example: Server power loss

Handling:

- Majority voting
- Heartbeat monitoring
- Leader election

Example: Apache Zookeeper, etcd (use Paxos/Raft)

Byzantine Faults (Complex)

- Node sends incorrect/conflicting information
- Malicious or software bugs
- Unpredictable behavior
- Example: Hacked node

Handling:

- Cryptographic proofs
- Economic incentives
- Supermajority consensus ($> 2/3$)

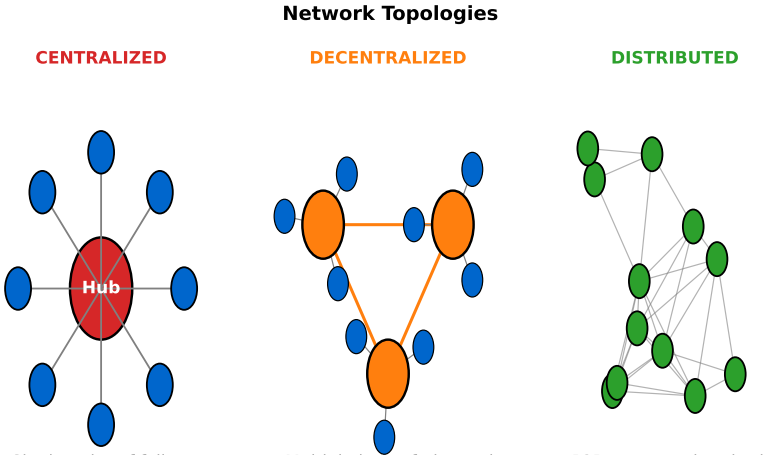
Example: Bitcoin (PoW), Tendermint (BFT)

Key Insight: Public blockchains assume Byzantine environment; private blockchains may assume only crash faults

---

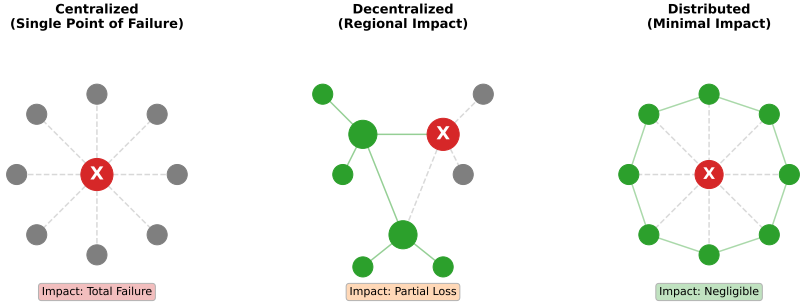
Compare the approaches shown above

How Do Network Topologies Compare?



Key Trade-off: Centralized is fast but fragile; Distributed is resilient but slower.

How Does Network Topology Affect Resilience to Node Failures?



Key Insight: Single node failure has catastrophic impact in centralized networks, negligible impact in distributed mesh networks.

How Does Network Topology Affect Resilience to Node Failures? – visual summary

How Do Centralized Networks Work?

Architecture:

- Single central node (hub)
- All communication flows through hub
- Clients connect to central server
- Hub controls access and data

Advantages:

- Fast and efficient
- Easy to manage and update
- Low coordination overhead

Disadvantages:

- Single point of failure
- Censorship possible
- Privacy concerns (hub sees all data)

Examples: Traditional banking systems, Facebook, Gmail

Centralized networks solve coordination efficiently but cannot provide Byzantine fault tolerance without trusted authority

How Do Decentralized Networks Work?

Architecture:

- Multiple hubs (federated model)
- Each hub serves subset of nodes
- Hubs communicate with each other
- No single central authority

Advantages:

- More resilient than centralized
- Regional autonomy
- Balanced between efficiency and resilience

Disadvantages:

- Hubs still represent points of failure
- Coordination between hubs needed
- Potential for fragmentation

Examples: Email (SMTP), Tor network, Mastodon (federated social media)

Key point: Architecture

How Do Distributed Networks Work?

Architecture:

- Peer-to-peer (P2P) topology
- No hubs or central points
- Every node equal (or nearly equal)
- Direct node-to-node communication

Advantages:

- Maximum resilience (no single point of failure)
- Censorship resistant
- Highly fault-tolerant

Disadvantages:

- Slower coordination (consensus overhead)
- Complex to manage and upgrade
- Higher resource requirements per node

Examples: Bitcoin, Ethereum, BitTorrent, IPFS

Key point: Architecture

What Are the Key Trade-offs Between Network Topologies?

Property	Centralized	Decentralized	Distributed
Control	Single entity	Multiple entities	All participants
Speed	Very fast	Fast	Slower
Resilience	Low (SPOF)	Medium	High
Censorship Resistance	None	Moderate	Strong
Coordination Overhead	Minimal	Moderate	High
Scalability	Easy (vertical)	Moderate	Hard (horizontal)
Trust Required	High	Medium	Low
Examples	Banks, Facebook	Email, Mastodon	Bitcoin, IPFS

Blockchain networks typically use fully distributed topology for maximum trustlessness

Compare the approaches shown above

Continued

The Solution: Distributed Consensus

[COMIC: Distributed vs centralized control]

Placeholder for comic illustration:

Left: King on throne (Centralized – single point of failure)

Right: Circle of equal nodes holding ledger copies

Caption: "When everyone keeps the books, no one can cheat"

What We Learned

- BFT: $> 2/3$ honest nodes required
- Network topology affects resilience
- DLT = consensus without central authority

The Trade-off

- Centralized: Fast but fragile
- Distributed: Resilient but slower

DLT transforms the Byzantine problem from military metaphor to working technology

Problem Solved? Distributed Ledger Technology

The Original Problem

How do we reach agreement in unreliable networks?

How Distributed Ledger Technology Solves It

- Permissioned vs permissionless architectures allow different trust assumptions
- Mathematical guarantee that $> 2/3$ honest nodes achieve consensus
- Peer-to-peer networks eliminate single points of failure

Remaining Limitations

- Cannot simultaneously maximize consistency, availability, and partition tolerance
- Distributed consensus is slower than centralized systems

Open Questions

- What is the optimal fault tolerance ratio (f/n) for real-world networks?
- Risk: Sybil attacks, eclipse attacks

DLT partially solves "agreement in unreliable networks" but introduces new trade-offs (speed vs decentralization)

The Economics of Byzantine Fault Tolerance

Incentive Structure

- Reaching consensus despite Byzantine (arbitrary) failures
- Validators stake assets; misbehavior results in slashing penalties
- Honest validators earn rewards; Byzantine actors lose their stake

Economic Security

- Slashing penalties ¿ potential gain from attack
- Staking creates "skin in the game"

Key Economic Question

Who Pays, Who Earns?

Byzantine actors pay slashing penalties; honest validators earn block rewards and fees

Design Principle

$$\text{Slash Amount} > \text{Attack Gain}$$

Byzantine tolerance through economic incentives: Make honesty profitable, misbehavior costly

Design Space: Byzantine Consensus Mechanisms

Alternatives Considered

1. Tolerates f failures with $3f + 1$ nodes, but $O(n^2)$ messages
2. Probabilistic finality, but simpler protocol

Trade-offs Made

- Immediate finality vs probabilistic finality
- Communication complexity vs simplicity

Design Questions

- What's acceptable: deterministic or probabilistic finality?
- How many rounds of communication are tolerable?
- Permissioned (known validators) or permissionless?

Key Insight

Consensus Trade-offs

Fast finality requires more communication overhead; simple protocols sacrifice determinism.

Design question: Immediate finality vs probabilistic finality - what's acceptable for your use case?

Failure Modes

Failure Modes: Byzantine Fault Tolerance

Critical Assumption

- At most 1/3 of nodes are Byzantine ($f < n/3$)
- More than 1/3 Byzantine nodes violates safety

Root Cause

- Network partitions can violate Byzantine assumptions temporarily
- Coordinated validator takeover
- Economic attack vectors (bribery)

Historical Context

- **2016 The DAO:** Smart contract exploit, not consensus failure, but led to contentious fork
- Consensus held; social layer split

Early Warning Signs

- ! Validator centralization (geographic, entity)
- ! Network partition events
- ! Sustained high confirmation delays

Warning: Network partitions can violate Byzantine assumptions temporarily - monitor validator diversity

Key Takeaways

What You Should Remember:

1. **DLT** is broader than blockchain; includes DAGs, Holochain, etc.
2. **Byzantine Generals Problem**: Achieving consensus with potentially malicious actors requires $> 2/3$ honest nodes
3. **Network Topologies**:
   - Centralized: fast but fragile
   - Decentralized: balanced
   - Distributed: resilient but slower
4. Public blockchains assume Byzantine faults; private may assume only crash faults

Next: L02b covers block structure, Merkle trees, and node types

Next Lesson: L02b – DLT Architecture

Key point: What You Should Remember

Discussion Questions

Consider and discuss:

1. **DLT Selection**: When would a DAG (like IOTA) be preferable to a blockchain?
2. **Byzantine Assumptions**: Are public blockchains too conservative in assuming Byzantine faults?
3. **Network Design**: Why do most cryptocurrencies choose distributed over decentralized topologies?

Key point: Consider and discuss

Thank you

Questions?

Continue to L02b: DLT Architecture

Quiz

Quiz Questions (1/2)

Q1. Which statement correctly distinguishes DLT from blockchain?
A) All DLTs are blockchains B) All blockchains are DLTs C) DLT and blockchain are synonyms D) Blockchains cannot use consensus
Answer: B – All blockchains are DLTs, but not all DLTs are blockchains (e.g., DAGs like IOTA).

Q2. In the Byzantine Generals Problem, what minimum fraction of nodes must be honest for consensus?
A) 1/2 B) 2/3 C) 3/4 D) 100%
Answer: B – The formal requirement is $n \geq 3f + 1$, meaning at least 2/3 honest nodes.

Q3. Which network topology has NO single point of failure?
A) Centralized B) Decentralized C) Distributed D) Federated
Answer: C – Distributed (P2P) networks have no central hubs, maximizing resilience.

Q4. Which fault type is more challenging for distributed systems to handle?
A) Crash faults B) Network partitions C) Byzantine faults D) Timeout errors
Answer: C – Byzantine faults involve malicious/conflicting behavior, harder than simple crashes.

Q5. What is the main advantage of centralized networks over distributed ones?
A) Censorship resistance B) Speed and efficiency C) No single point of failure D) Maximum decentralization
Answer: B – Centralized networks are faster/more efficient but have single point of failure.

Quiz Questions (2/2)

Q6. Which DLT architecture uses a Directed Acyclic Graph instead of blocks?
A) Bitcoin B) IOTA Tangle C) Hyperledger Fabric D) Cardano
Answer: B – IOTA uses DAG (Tangle) where transactions reference previous transactions.

Q7. Who formulated the Byzantine Generals Problem in 1982?
A) Satoshi Nakamoto B) Vitalik Buterin C) Lamport, Shostak, and Pease D) Nick Szabo
Answer: C – Lamport, Shostak, and Pease published the foundational paper in 1982.

Q8. What consensus algorithm is commonly used for crash fault tolerance (not Byzantine)?
A) Proof-of-Work B) PBFT C) Paxos/Raft D) Proof-of-Stake
Answer: C – Paxos and Raft handle crash faults; PBFT and PoW handle Byzantine faults.

Q9. Which is an example of a decentralized (not distributed) network?
A) Bitcoin B) Email (SMTP) C) IPFS D) Ethereum
Answer: B – Email uses federated servers (decentralized hubs), not full P2P distribution.

Q10. What is the key difference between shared ledgers and distributed ledgers?
A) Shared ledgers are faster B) Distributed ledgers require no central authority C) Shared ledgers use consensus D) Distributed ledgers need permission
Answer: B – Distributed ledgers achieve consensus without central authority; shared ledgers rely on one.