Blockchain 101

Lecture Outline

- Blockchain History
- Distributed Systems Design
- Blockchain defined
- Consensus
- CAP theorem and Blockchain
- Decentralization

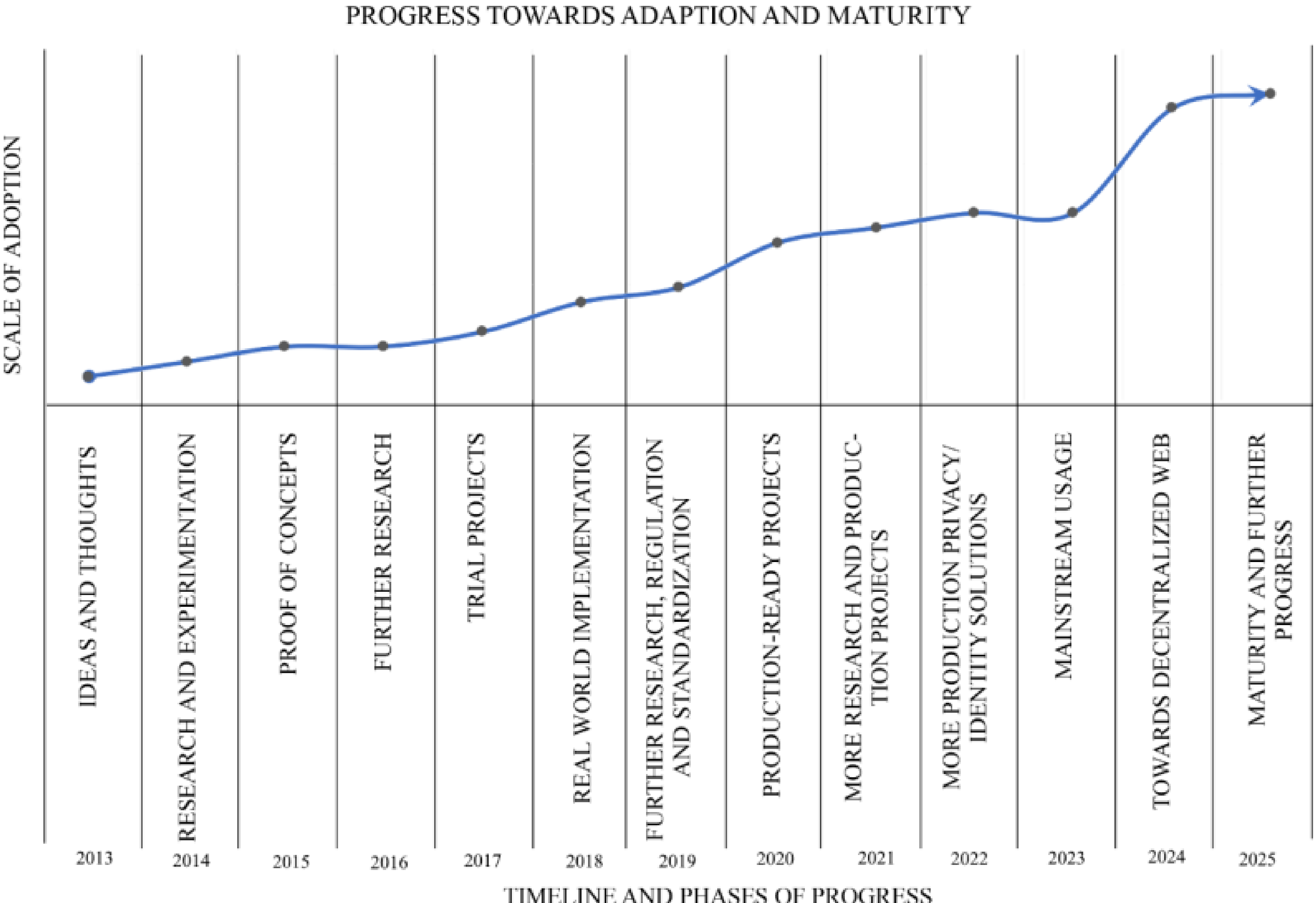
Lecture Outline

- **Blockchain History**
- Distributed Systems Design
- Blockchain defined
- Consensus
- CAP theorem and Blockchain
- Decentralization

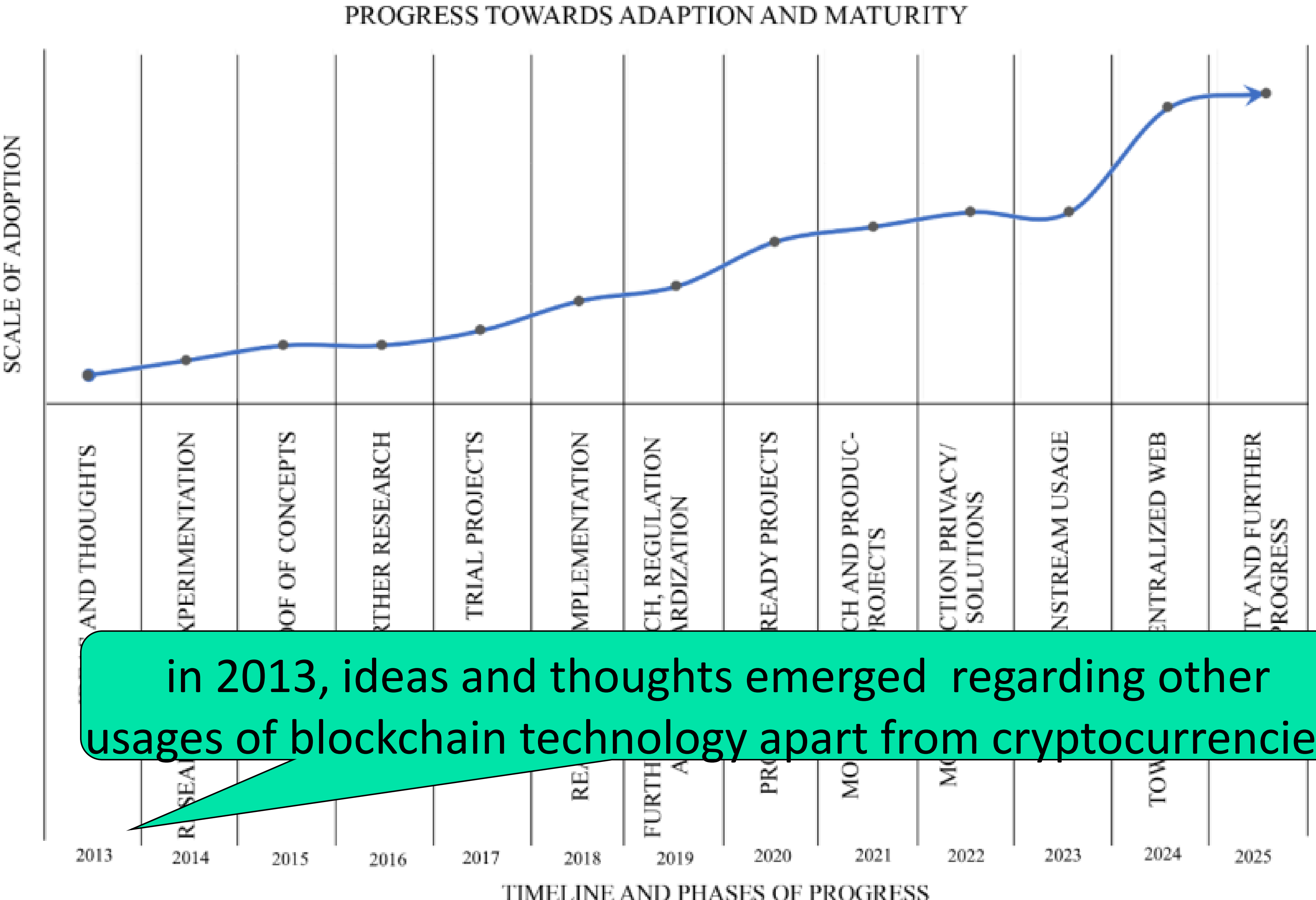
Blockchain History

- With the invention of Bitcoin in 2008, the world was introduced to a new concept, which revolutionized the whole of society
- It was something that promised to have an impact upon every industry.
- This new concept was blockchain; the underlying technology that underpins Bitcoin.
- In 2013, some ideas started to emerge that suggested that blockchain may have the potential for application in areas other than cryptocurrencies.

Blockchain History

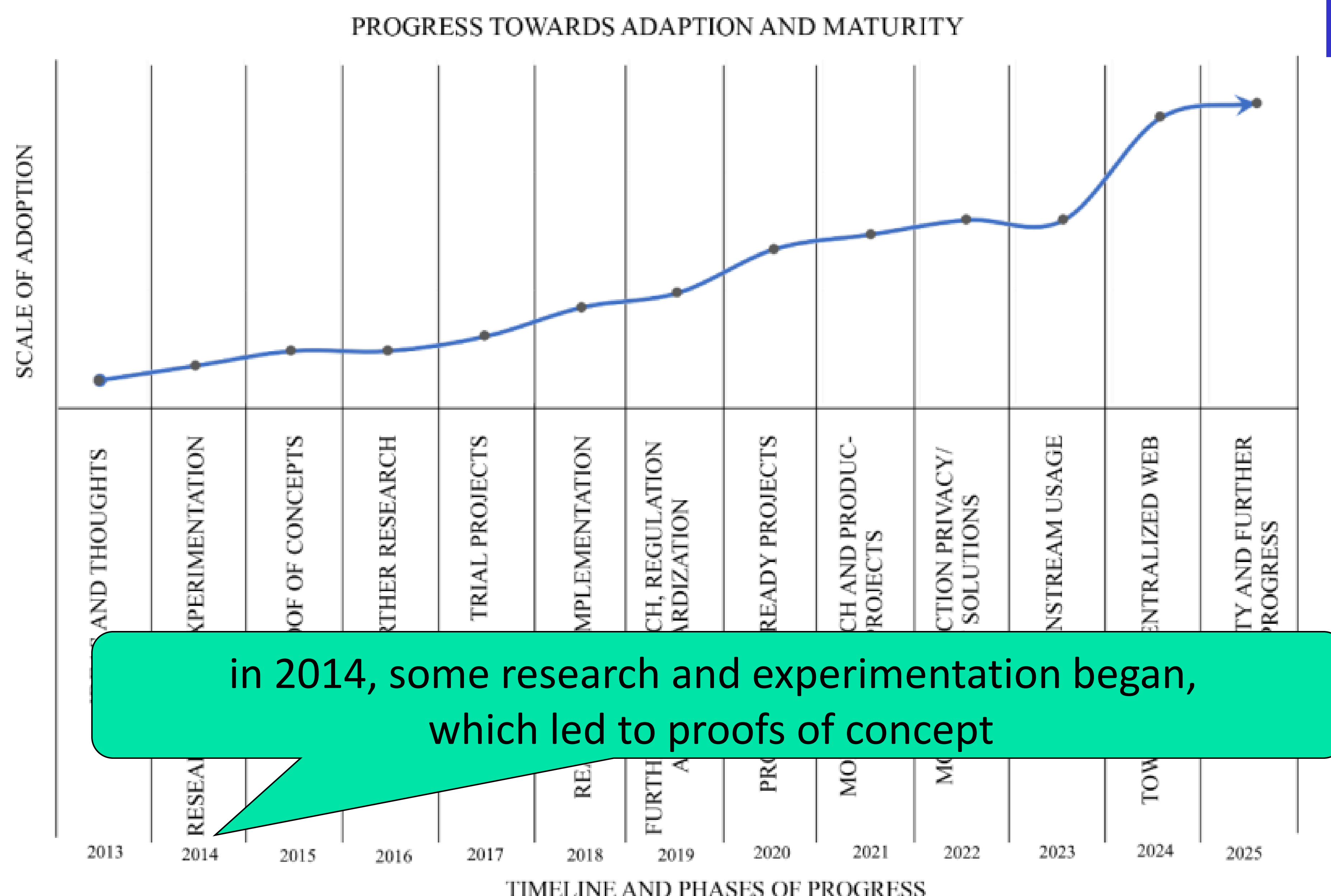


Blockchain History



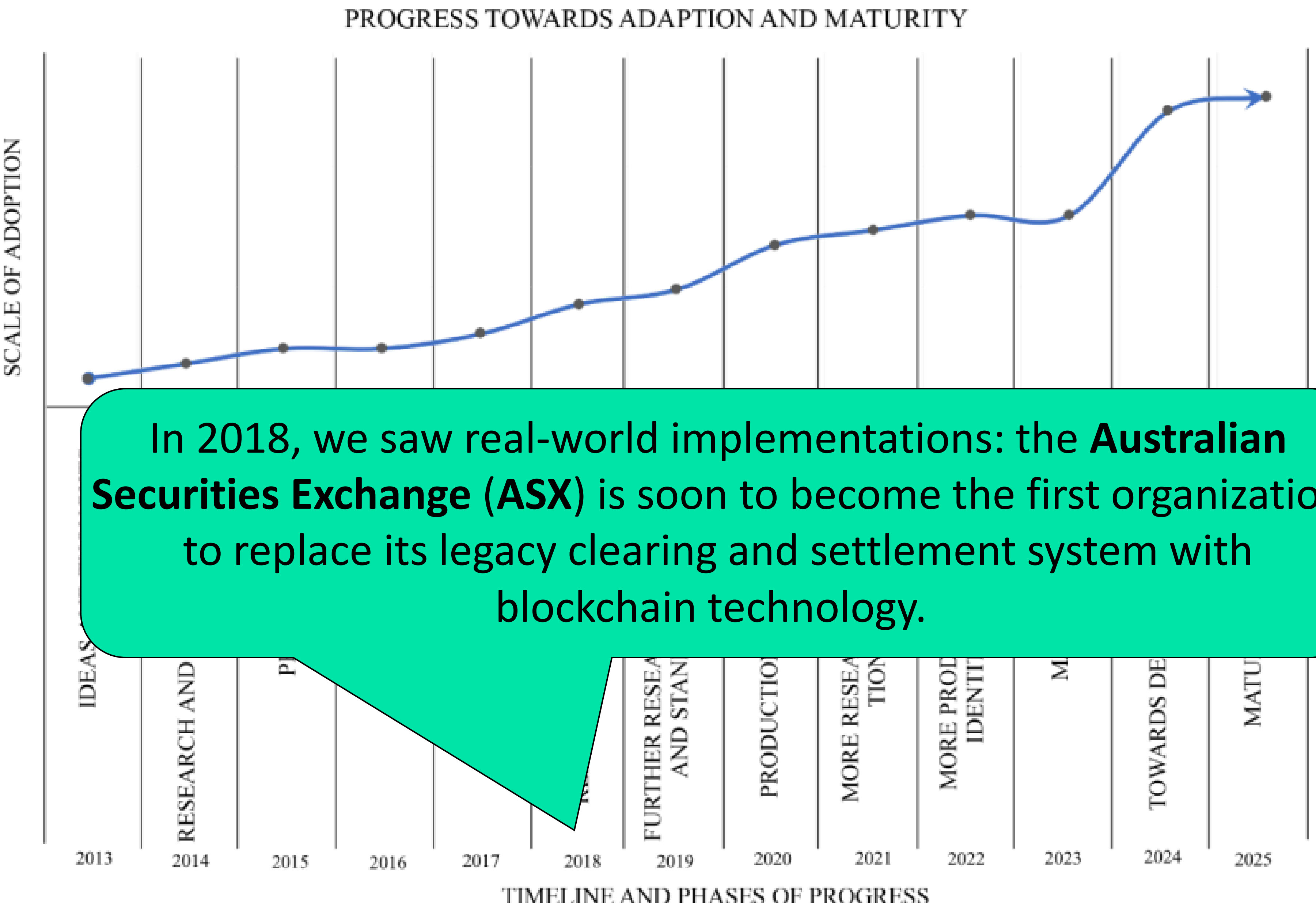
in 2013, ideas and thoughts emerged regarding other usages of blockchain technology apart from cryptocurrencies

Blockchain History



in 2014, some research and experimentation began, which led to proofs of concept

Blockchain History



In 2018, we saw real-world implementations: the **Australian Securities Exchange** (**ASX**) is soon to become the first organization to replace its legacy clearing and settlement system with blockchain technology.

Lecture Outline

- Blockchain History
- **Distributed Systems Design**
- Blockchain defined
- Consensus
- CAP theorem and Blockchain
- Decentralization

Distributed Systems

- Understanding distributed systems is essential to understand blockchain, as blockchain was a distributed system at its core.
- It is a distributed ledger that can be centralized or decentralized.
- It can be thought of as a system that has properties of the both **decentralized** and **distributed** paradigms. It is a decentralized-distributed system (decentralized vs. distributed?)

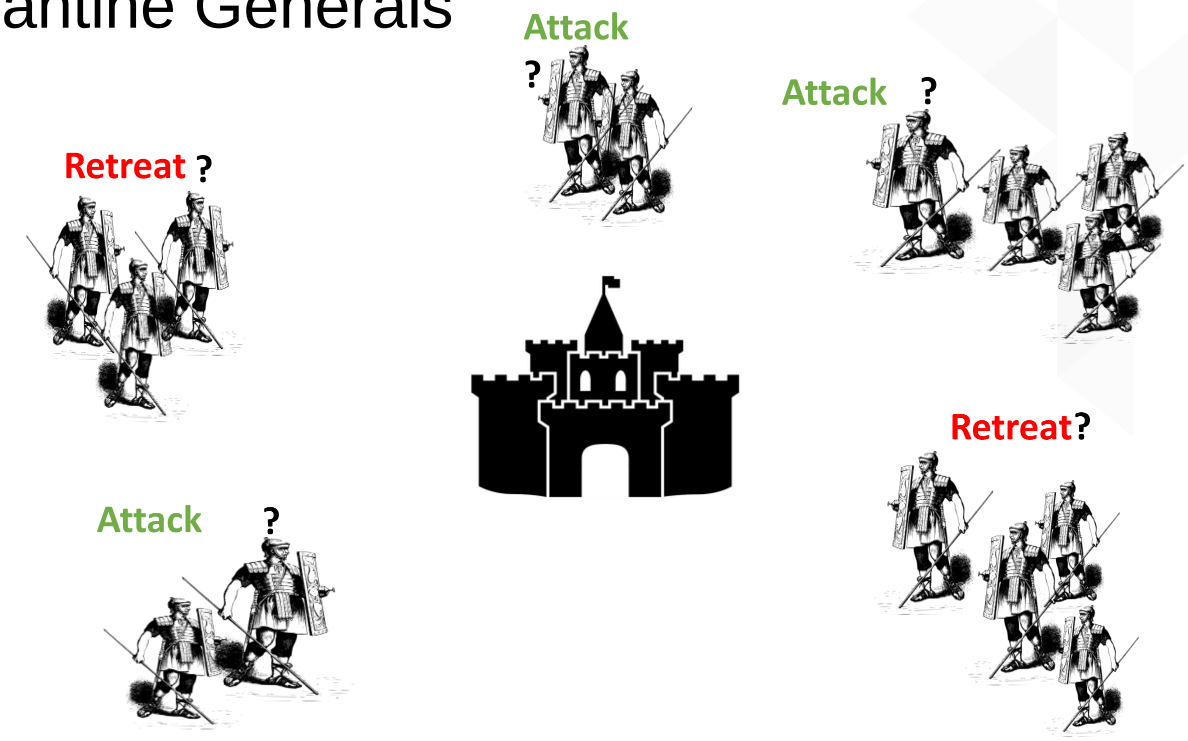
Distributed Systems (Cont.)

- **Distributed systems** are a computing paradigm whereby two or more nodes work with each other in a coordinated fashion to achieve a common outcome.
- It is modeled in such a way that end users see it as a single logical platform.
- A **node** can be defined as an individual player in a distributed system.
- All nodes are capable of sending and receiving messages to and from each other. Nodes can be honest, faulty, or malicious, and they have memory and a processor

Distributed Systems (Cont.)

- A node that exhibits irrational behavior is also known as a *Byzantine node* after the **Byzantine Generals** problem

The Byzantine Generals problem



Attack or retreat?
Consensus required to win

The Byzantine Generals Problem

- A group of army generals who lead different parts of the Byzantine army is planning to attack or retreat from a city, where they only communicate among them is via a messenger.
- They need to agree to strike at the same time in order to win.
- The issue is that one or more generals might be traitors who could send a misleading message.
- There is a need for a viable mechanism that allows for agreement among the generals, even in the presence of the treacherous ones, so that the attack can take place at the same time.

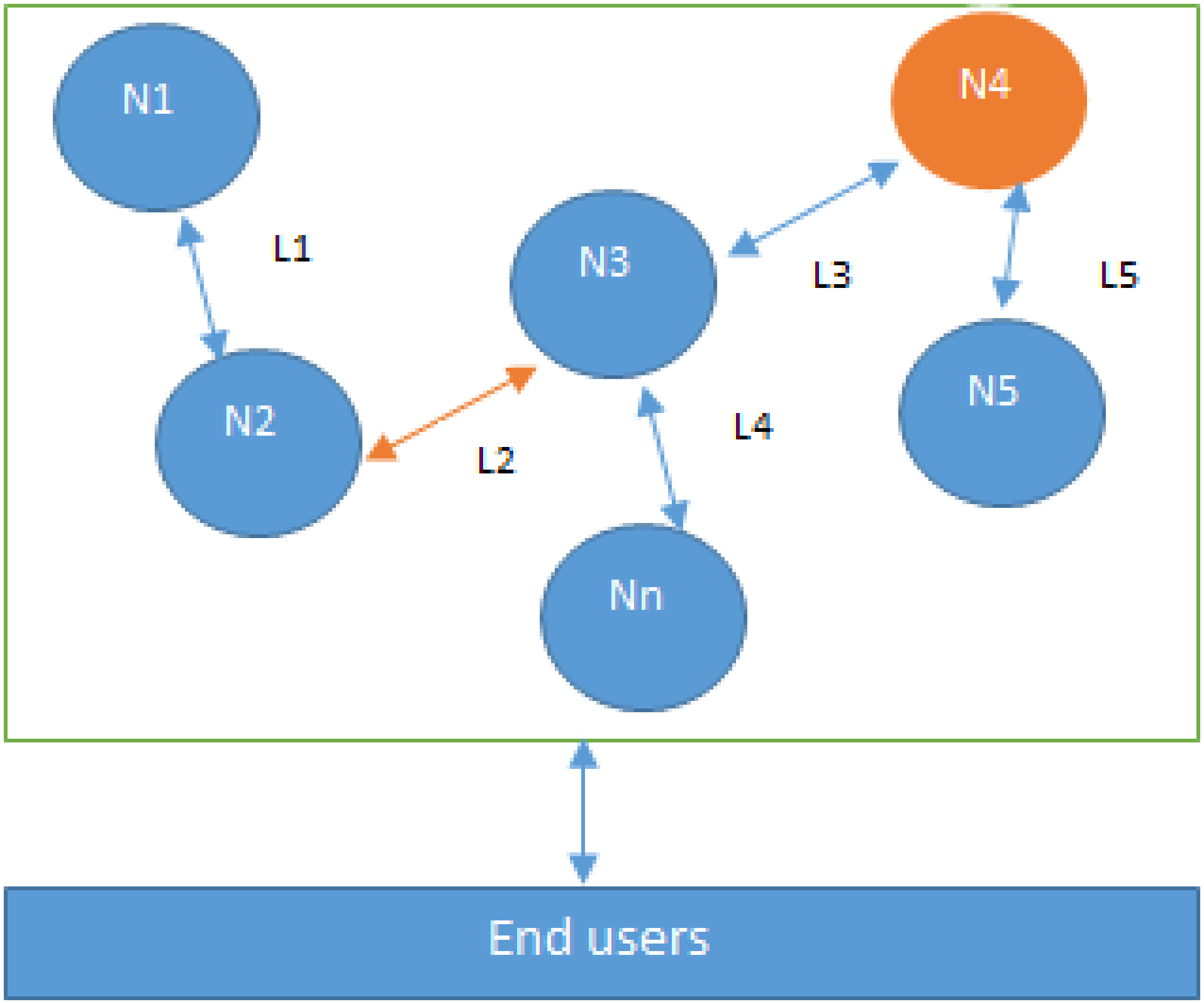
The Byzantine Generals Problem

- In 1999, Castro and Liskov presented the **Practical Byzantine Fault Tolerance** (**PBFT**) algorithm, which solves the consensus problem in the presence of Byzantine faults in asynchronous networks by utilizing the state machine replication protocol.
- PBFT goes through a number of rounds to eventually reach an agreement between nodes on the proposed value.

The Byzantine Generals Problem

- This type of inconsistent behavior of Byzantine nodes can be intentionally malicious, which is detrimental to the operation of the network.
- Any unexpected behavior by a node on the network, whether malicious or not, can be categorized as Byzantine.

Design of a distributed system



N4 is a Byzantine node, L2 is broken or a slow network link

Design of a Distributed System

- The primary challenge of a distributed system design is the coordination between nodes and fault tolerance.
- Even if some (a certain threshold dictated by the consensus protocol) of the nodes become faulty or network links break, the distributed system should be able to tolerate this and continue to work.
- Distributed systems are so challenging to design that a theory known as the **CAP theorem** has been proven

CAP theorem

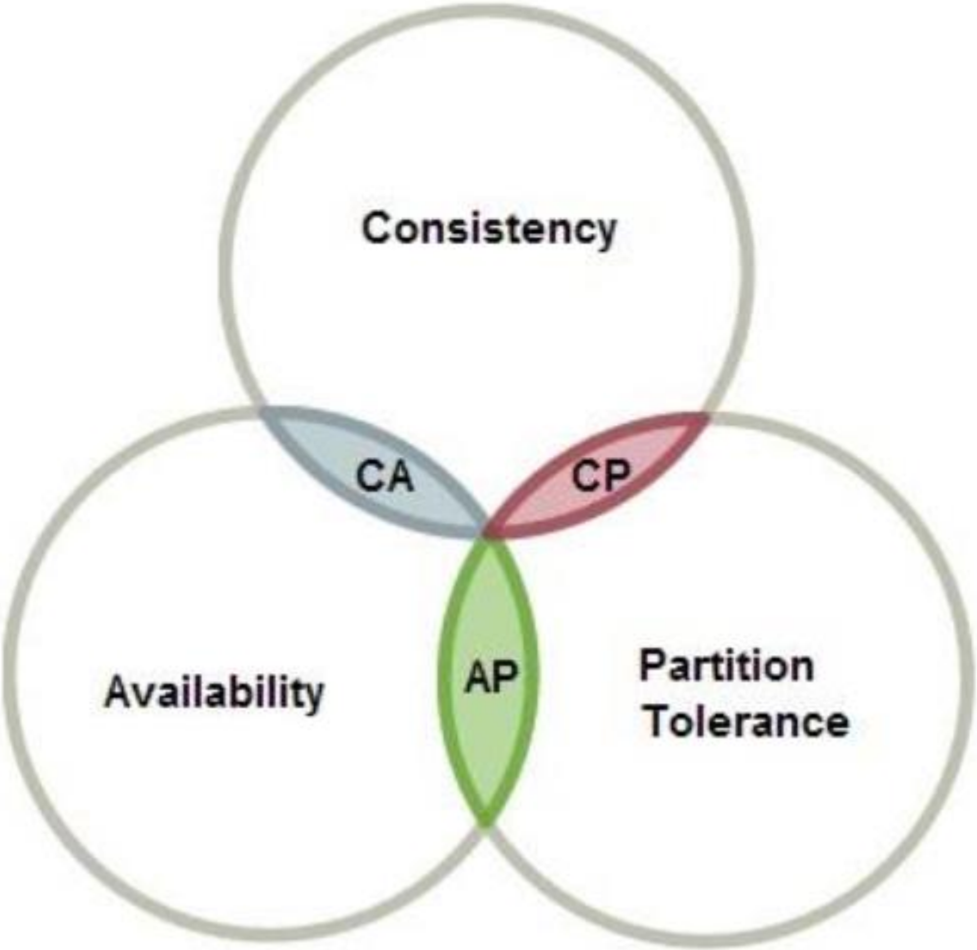
This states that a distributed system cannot have all three of the desired properties simultaneously; that is:

- Consistency: is a property that ensures that all nodes in a distributed system have a single, current, and identical copy of the data
- Availability: means that the nodes in the system are up, accessible for use, and are accepting incoming requests and responding with data without any failures as and when required.
- Partition tolerance: ensures that if a group of nodes is unable to communicate with other nodes due to network failures, the distributed system continues to operate correctly. This can occur due to network and node failures.

CAP theorem

This states that a distributed system cannot have all three of the desired properties simultaneously; that is:

- Consistency
- Availability
- Partition tolerar



CAP Theorem Example

- Let's imagine that there is a distributed system with two nodes. Now, let's apply the three theorem properties on this smallest of possible distributed systems only with two nodes
  - How is consistency achieved?
  - How is availability achieved?
  - How is partition tolerance achieved?
- Now assume that a partition occurs in that network, and nodes can no longer communicate with each other.

CAP Theorem Example

- Now assume that a partition occurs in that network, and nodes can no longer communicate with each other.
- What happens if a new update comes?
  - What is sacrificed?

Types of faults in distributed systems

Fail-stop faults (crash faults)

- Where components crash or cease to operate
- Simpler to deal with

Byzantine faults

- Where components are potentially untrustworthy or malicious
- Difficult to deal with

Design of Distributed Systems

- Even though blockchain can be considered to be both a distributed and decentralized system, there are, however, critical differences between distributed systems and decentralized systems that make both of these systems architecturally different.
- Blockchain are claimed to break the CAP theorem (How true is that?)

Lecture Outline

- Blockchain History
- Distributed Systems Design
- Blockchain defined
- Consensus
- CAP theorem and Blockchain
- Decentralization

Lecture Outline

- Blockchain History
- Distributed Systems Design
- Electronic Cash
- **Blockchain defined**
- Consensus
- CAP theorem and Blockchain
- Decentralization

Defining ‘Blockchain’

Layman's definition: Blockchain is an ever-growing, secure, shared recordkeeping system in which each user of the data holds a copy of the records, which can only be updated if all parties involved in a transaction agree to update.

Technical definition: Blockchain is a peer-to-peer distributed ledger that is cryptographically-secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among peers.

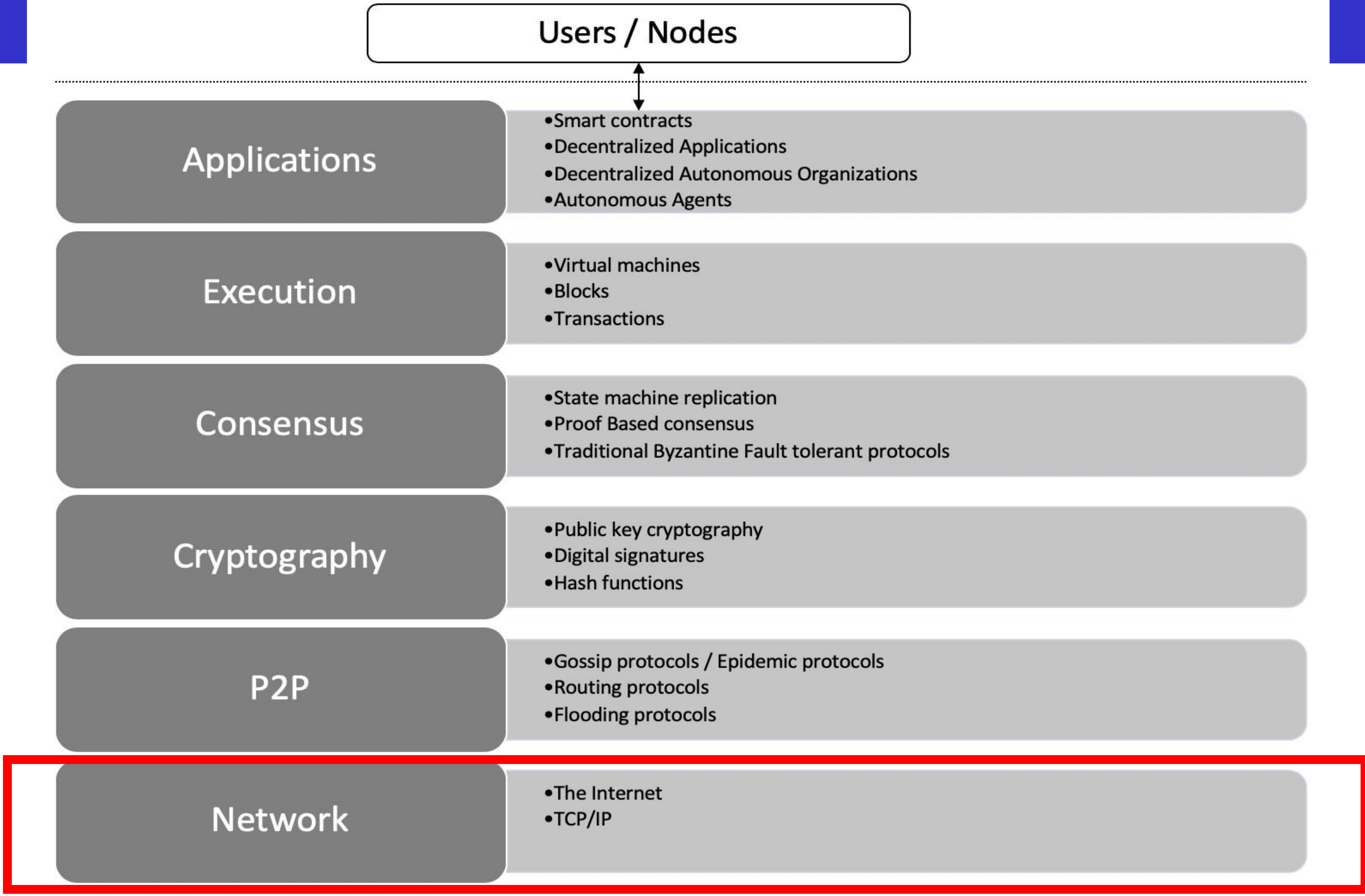
Blockchain definition

- Peer-to-peer: This means that there is no central controller in the network
- Distributed ledger: means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the complete ledger.
- Cryptographically secure: which means that cryptography has been used to provide security services that make this ledger secure against tampering and misuse.

Blockchain definition

- Append only: that blockchain is "append-only," which means that data can only be added to the blockchain in *time-sequential order*
- Updateable via consensus (consensus-driven): In this scenario, no central authority is in control of updating the ledger.
- Instead, any update made to the blockchain is validated against strict criteria defined by the blockchain protocol and added to the blockchain only after a **consensus** has been reached among all participating peers/nodes on the network.

Architectural view of Blockchain



Users / Nodes

Applications

- Smart contracts
- Decentralized Applications
- Decentralized Autonomous Organizations
- Autonomous Agents

Execution

- Virtual machines
- Blocks
- Transactions

Consensus

- State machine replication
- Proof Based consensus
- Traditional Byzantine Fault tolerant protocols

Cryptography

- Public key cryptography
- Digital signatures
- Hash functions

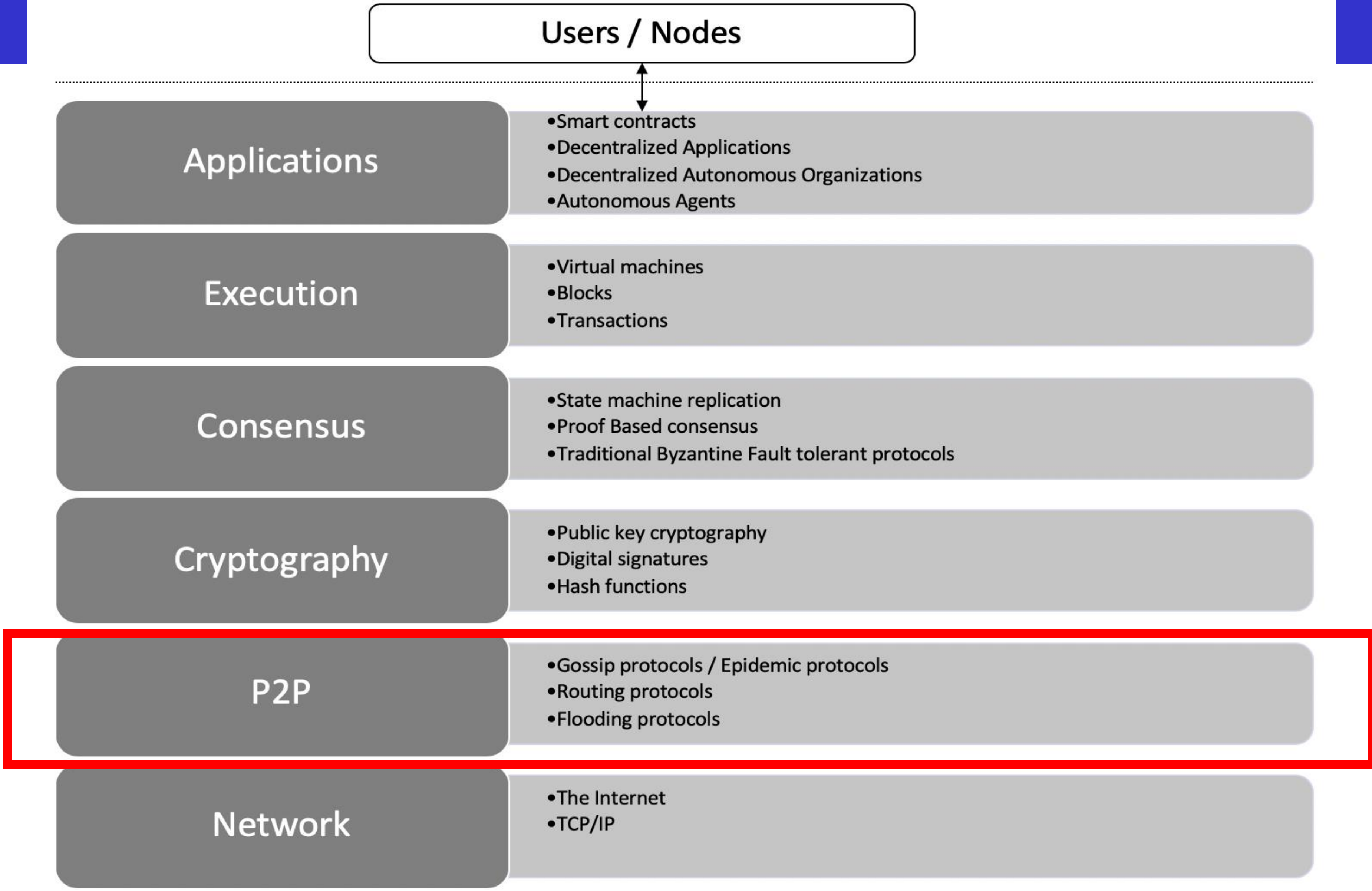
P2P

- Gossip protocols / Epidemic protocols
- Routing protocols
- Flooding protocols

Network

- The Internet
- TCP/IP

Architectural view of Blockchain



Users / Nodes

Applications

- Smart contracts
- Decentralized Applications
- Decentralized Autonomous Organizations
- Autonomous Agents

Execution

- Virtual machines
- Blocks
- Transactions

Consensus

- State machine replication
- Proof Based consensus
- Traditional Byzantine Fault tolerant protocols

Cryptography

- Public key cryptography
- Digital signatures
- Hash functions

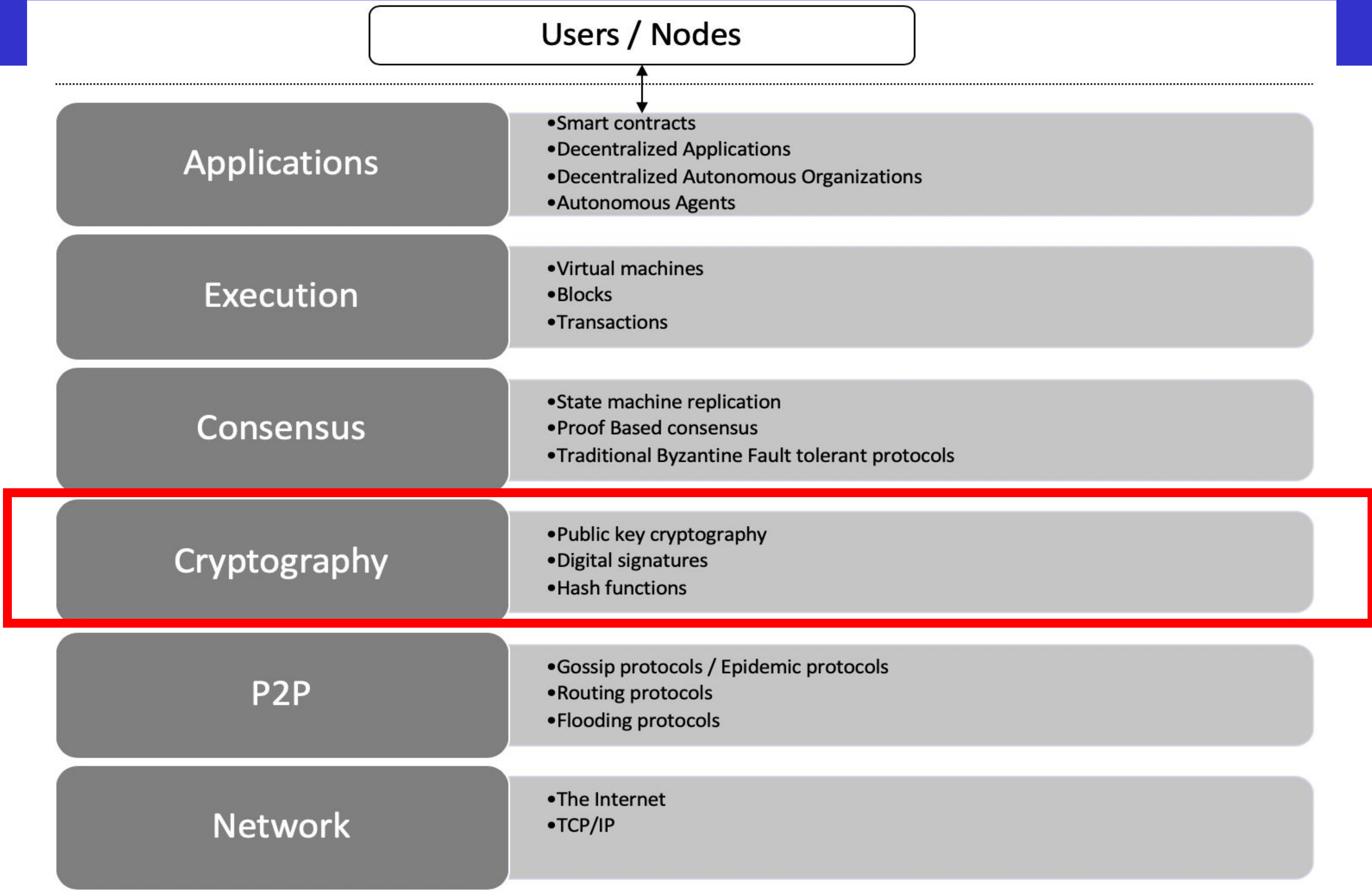
P2P

- Gossip protocols / Epidemic protocols
- Routing protocols
- Flooding protocols

Network

- The Internet
- TCP/IP

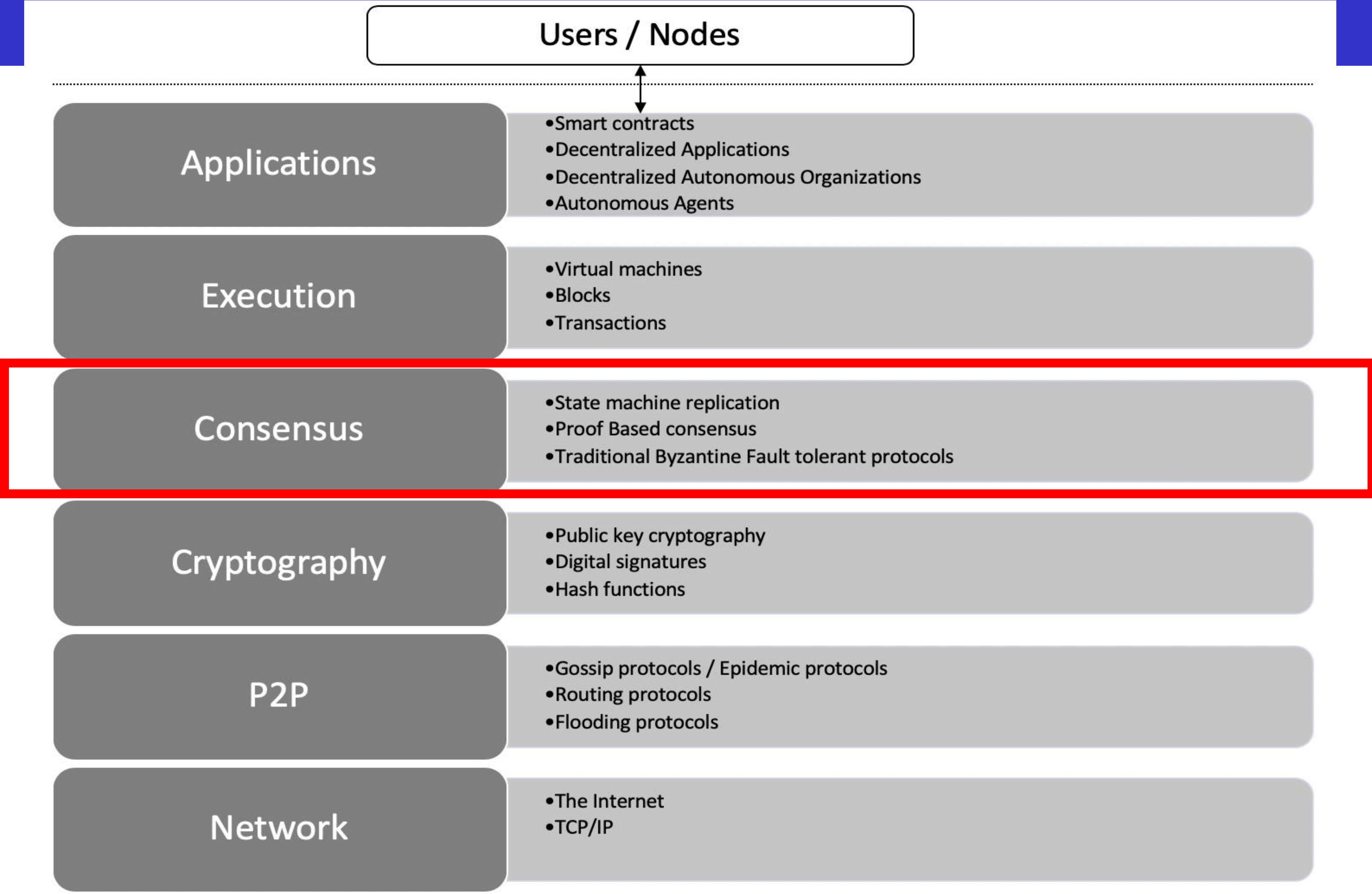
Architectural view of Blockchain



Users / Nodes

Applications	•Smart contracts •Decentralized Applications •Decentralized Autonomous Organizations •Autonomous Agents
Execution	•Virtual machines •Blocks •Transactions
Consensus	•State machine replication •Proof Based consensus •Traditional Byzantine Fault tolerant protocols
Cryptography	•Public key cryptography •Digital signatures •Hash functions
P2P	•Gossip protocols / Epidemic protocols •Routing protocols •Flooding protocols
Network	•The Internet •TCP/IP

Architectural view of Blockchain



Users / Nodes

Applications
- Smart contracts
- Decentralized Applications
- Decentralized Autonomous Organizations
- Autonomous Agents

Execution
- Virtual machines
- Blocks
- Transactions

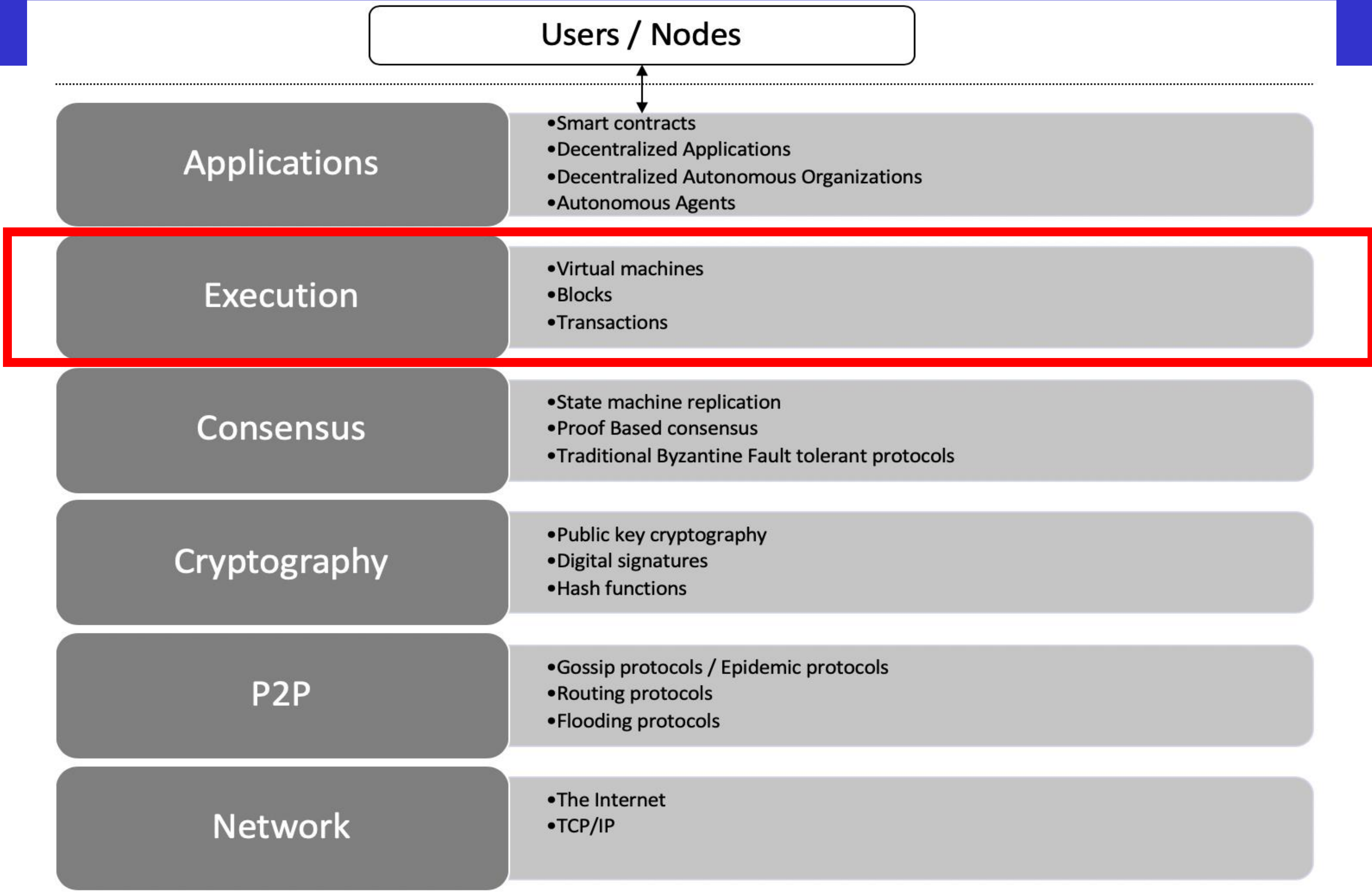
Consensus
- State machine replication
- Proof Based consensus
- Traditional Byzantine Fault tolerant protocols

Cryptography
- Public key cryptography
- Digital signatures
- Hash functions

P2P
- Gossip protocols / Epidemic protocols
- Routing protocols
- Flooding protocols

Network
- The Internet
- TCP/IP

Architectural view of Blockchain



Users / Nodes

Applications

- Smart contracts
- Decentralized Applications
- Decentralized Autonomous Organizations
- Autonomous Agents

Execution

- Virtual machines
- Blocks
- Transactions

Consensus

- State machine replication
- Proof Based consensus
- Traditional Byzantine Fault tolerant protocols

Cryptography

- Public key cryptography
- Digital signatures
- Hash functions

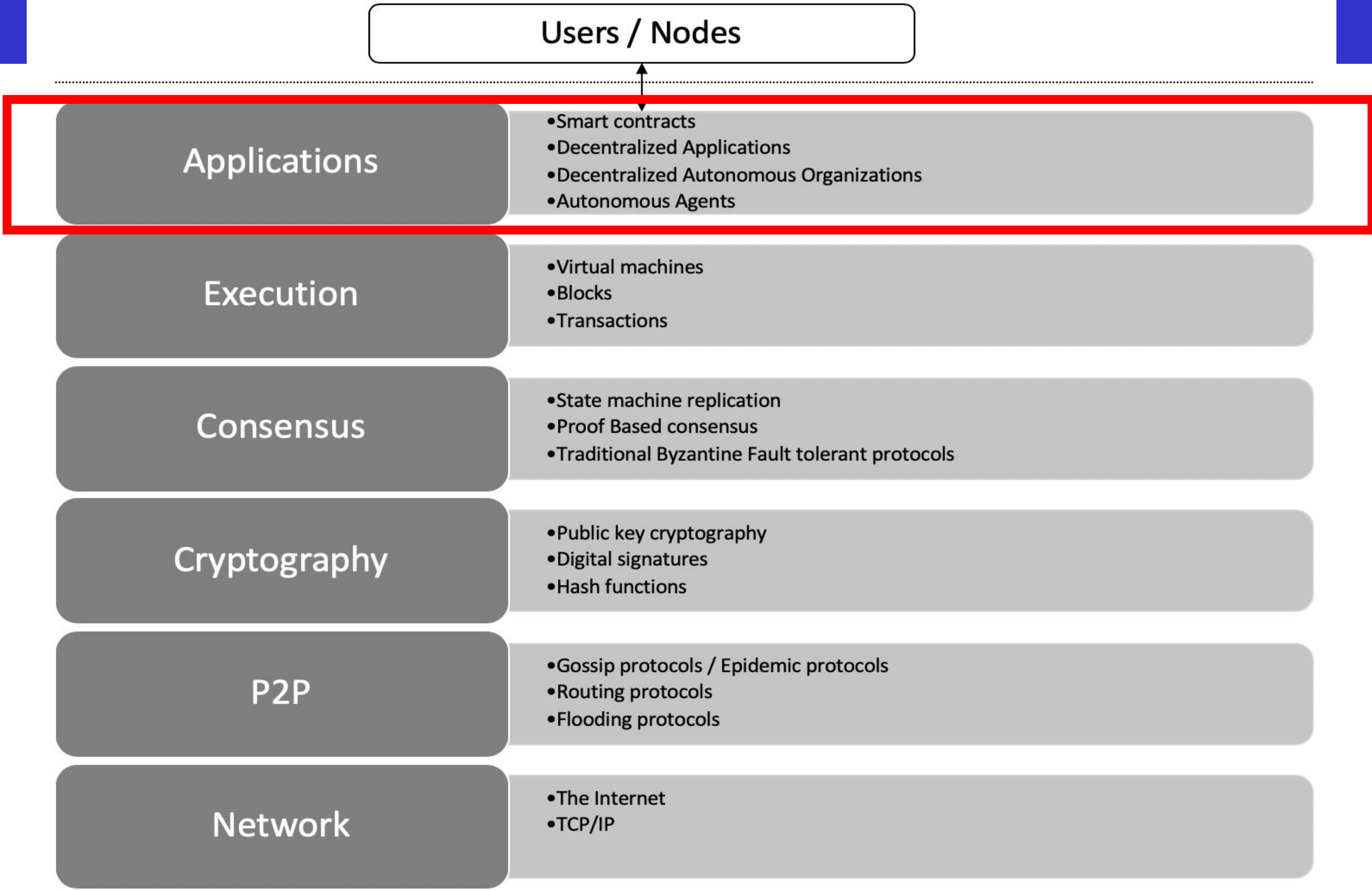
P2P

- Gossip protocols / Epidemic protocols
- Routing protocols
- Flooding protocols

Network

- The Internet
- TCP/IP

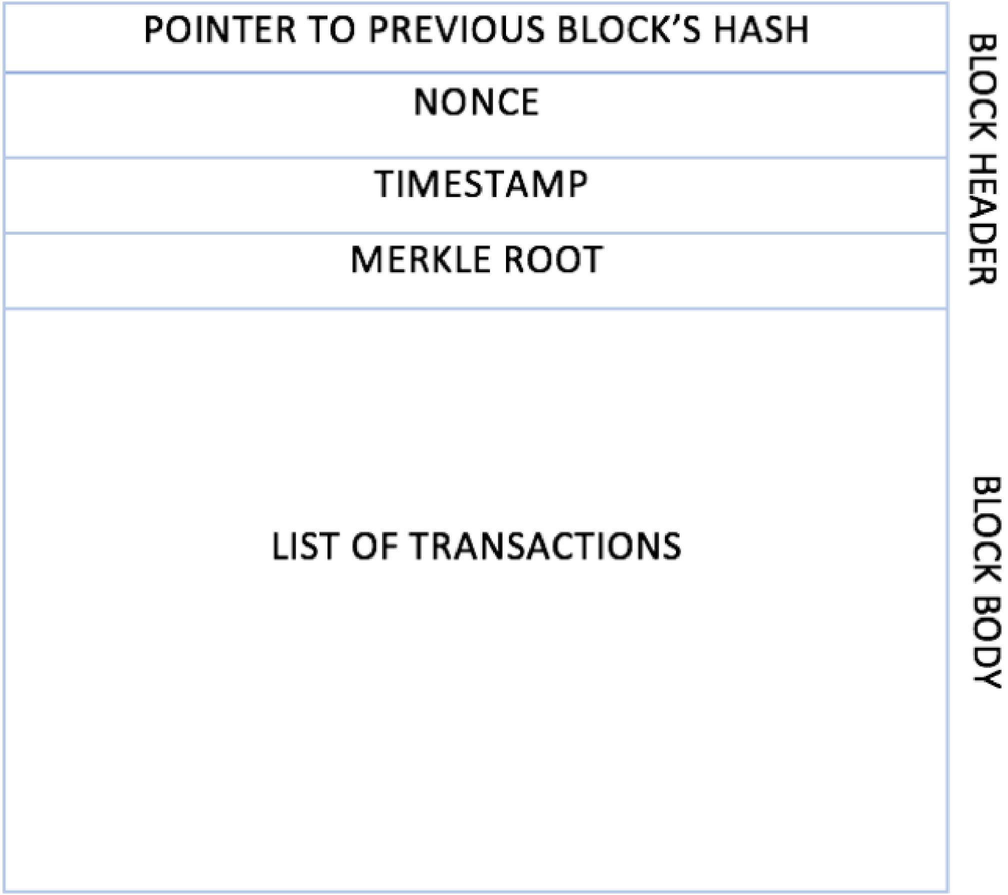
Architectural view of Blockchain



Users / Nodes

Applications	•Smart contracts •Decentralized Applications •Decentralized Autonomous Organizations •Autonomous Agents
Execution	•Virtual machines •Blocks •Transactions
Consensus	•State machine replication •Proof Based consensus •Traditional Byzantine Fault tolerant protocols
Cryptography	•Public key cryptography •Digital signatures •Hash functions
P2P	•Gossip protocols / Epidemic protocols •Routing protocols •Flooding protocols
Network	•The Internet •TCP/IP

The Generic Structure of a Block



POINTER TO PREVIOUS BLOCK'S HASH

NONCE

TIMESTAMP

MERKLE ROOT

LIST OF TRANSACTIONS

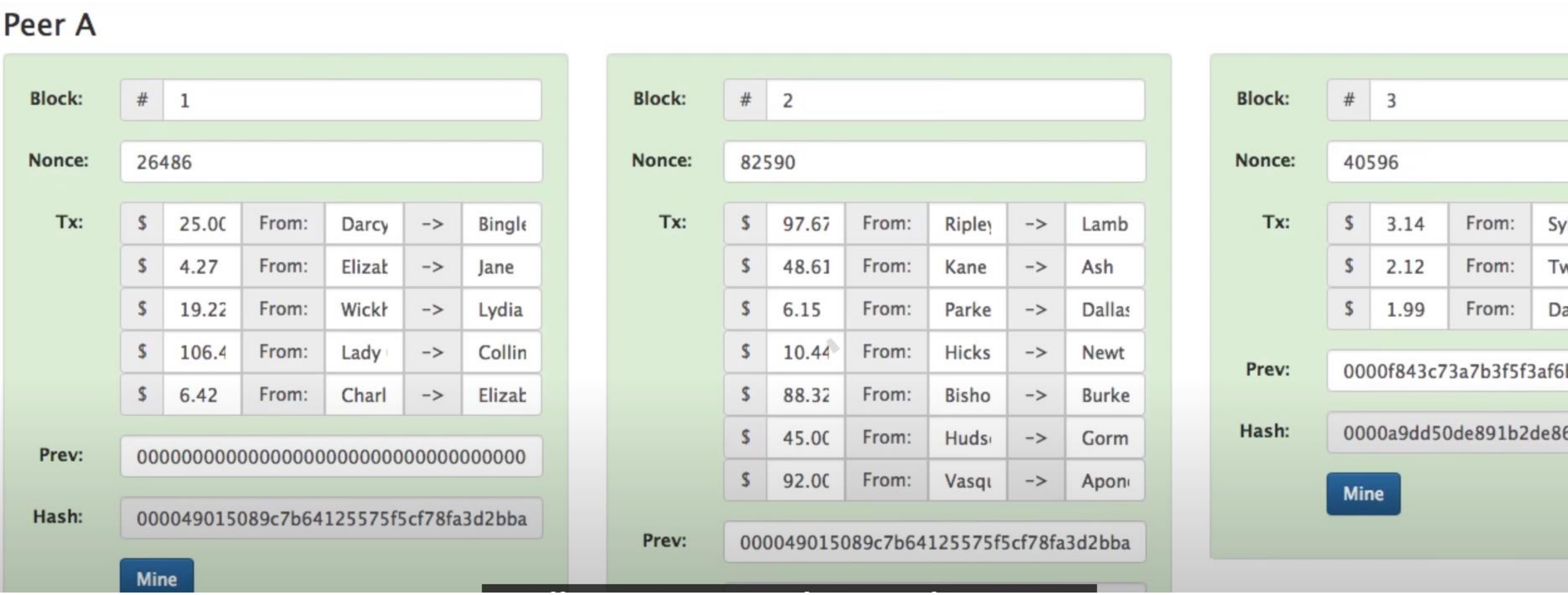
BLOCK HEADER

BLOCK BODY

The Generic Structure of a Block

- **Address**: Addresses are unique identifiers used in a blockchain transaction to denote senders and recipients. An address is usually a public key or derived from a public key.
- **Transaction**: A transaction is the fundamental unit of a blockchain. A transaction represents a transfer of value from one address to another.

The Generic Structure of a Block



Peer A

Block: # 1

Nonce: 26486

Tx:

$	25.0	From:	Darcy	->	Bingl
$	4.27	From:	Eliza	->	Jane
$	19.2	From:	Wick	->	Lydia
$	106.4	From:	Lady	->	Collin
$	6.42	From:	Charl	->	Eliza

Prev: 0000000000000000000000000000000000

Hash: 000049015089c7b64125575f5cf78fa3d2bba

Mine

Block: # 2

Nonce: 82590

Tx:

$	97.6	From:	Riple	->	Lamb
$	48.6	From:	Kane	->	Ash
$	6.15	From:	Parke	->	Dalla
$	10.44	From:	Hicks	->	Newt
$	88.3	From:	Bisho	->	Burke
$	45.0	From:	Huds	->	Gorm
$	92.0	From:	Vasq	->	Apon

Prev: 000049015089c7b64125575f5cf78fa3d2bba

Block: # 3

Nonce: 40596

Tx:

$	3.14	From:	Sy
$	2.12	From:	Tw
$	1.99	From:	Da

Prev: 0000f843c73a7b3f5f3af6

Hash: 0000a9dd50de891b2de8

Mine

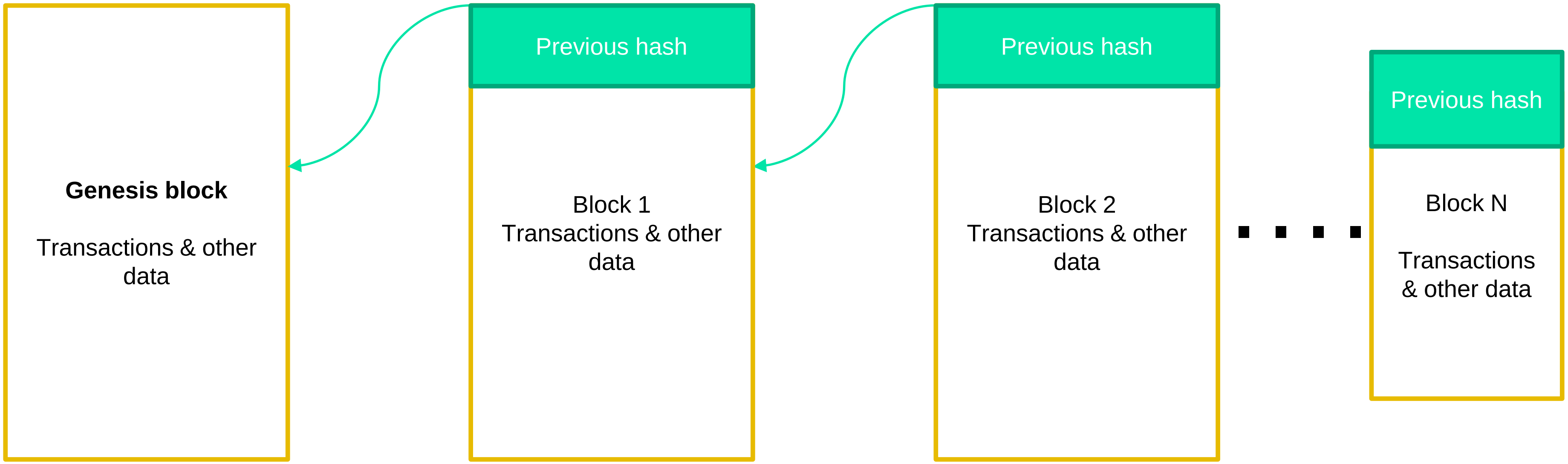
The Generic Structure of a Block

- **Block**: A block is composed of a block header and a selection of transactions bundled together and organized logically. A block contain several elements, which we introduce as follows:
  - A **nonce** is a number that is generated and used only once. A nonce is used extensively in many cryptographic operations to provide replay protection, authentication, and encryption. In blockchain, it' used in PoW consensus algorithms and for transaction replay protection. A block also includes the nonce value.
  - A **timestamp** is the creation time of the block.

The Generic Structure of a Block

- **Block**: A block is composed of a block header and a selection of transactions bundled together and organized logically. A block contains several elements, which we introduce as follows:
    - A reference to a previous block is also included in the block unless it is a genesis
      block. This reference is the hash of the header of the previous block.
    - A **genesis block** is the first block in the blockchain that is hardcoded at the time the blockchain was first started. The structure of a block is also dependent on the type and design of a blockchain.
    - A **nonce** is a number that is generated and used only once. A nonce is used extensively in many cryptographic operations to provide replay protection, authentication, and encryption. In blockchain, it's used in PoW consensus algorithms and for transaction replay protection. A block also includes the nonce value.
    - A **timestamp** is the creation time of the block.

Generic structure of a blockchain



Genesis block

Transactions & other data

Previous hash

Block 1
Transactions & other data

Previous hash

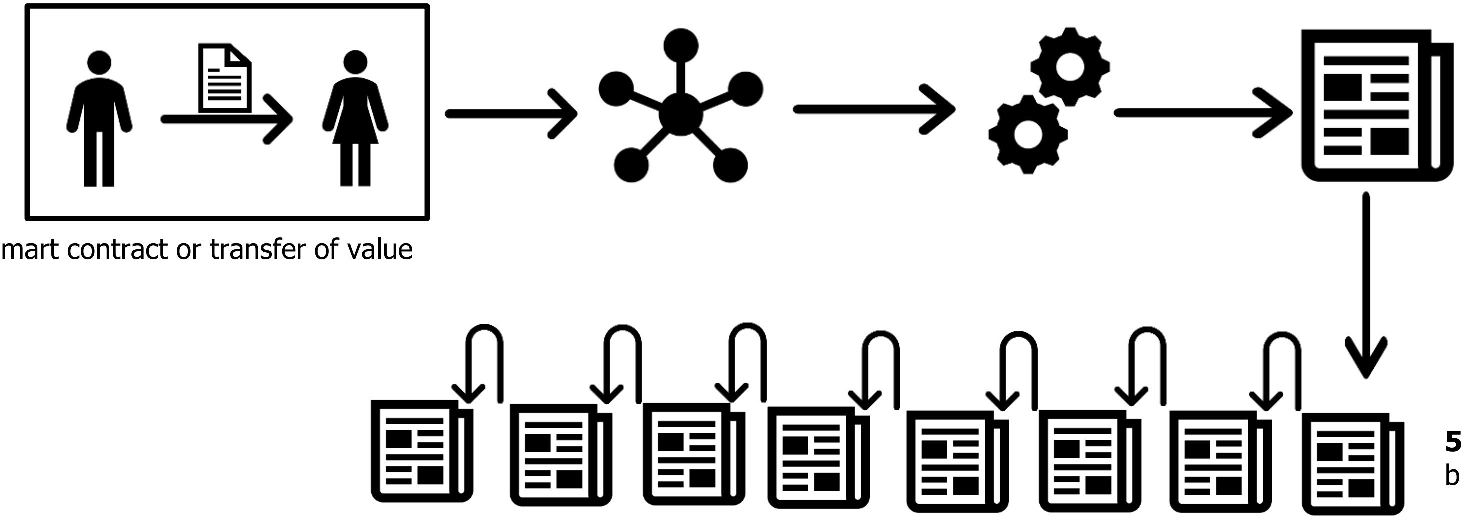
Block 2
Transactions & other data

Previous hash

Block N

Transactions & other data

How a blockchain works



1. User X transacts with User Y

Smart contract or transfer of value

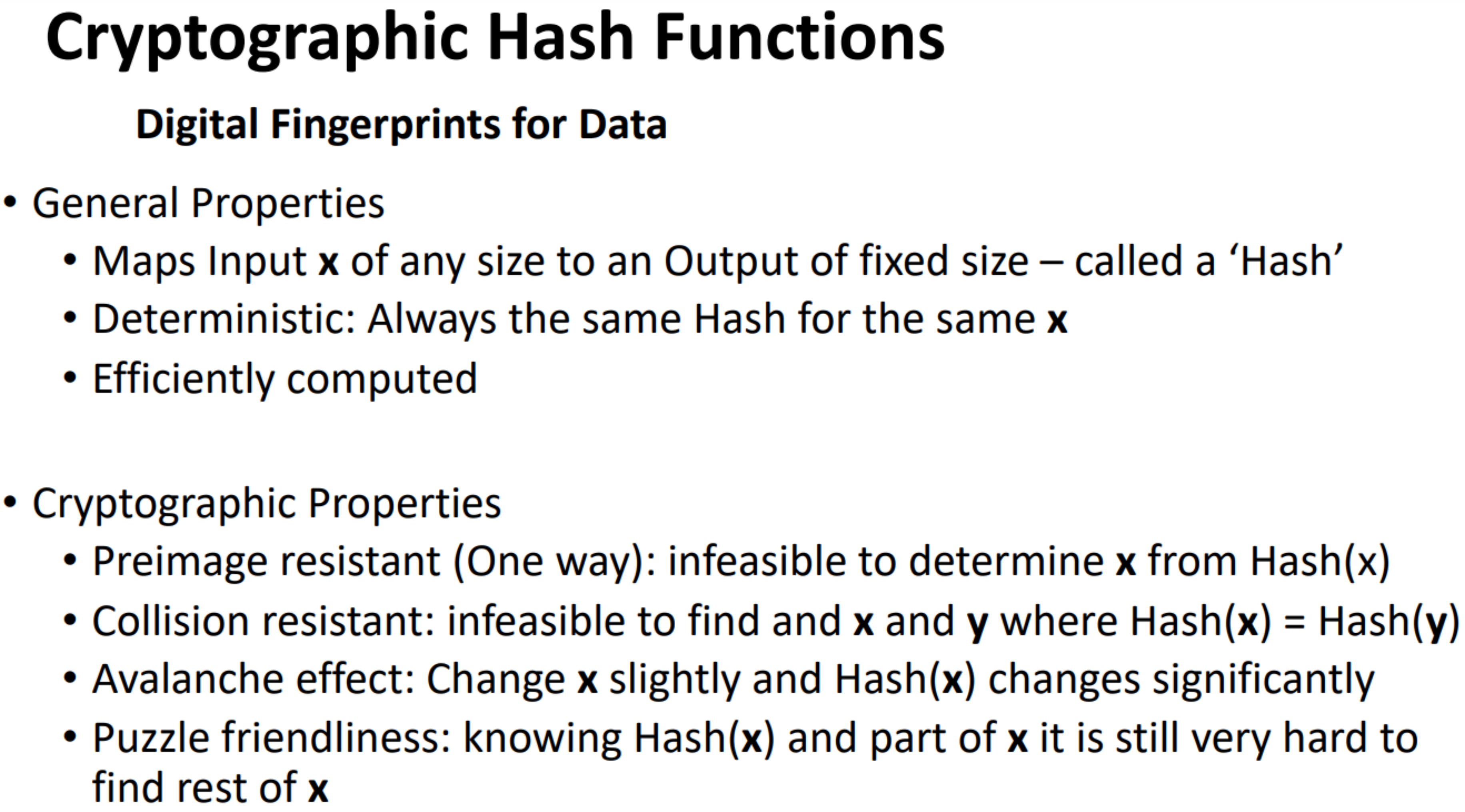
2. Transaction broadcast

3. Find new block (mining)

4. New block found (mined)

5. Add new block to the blockchain

What is a Hash Function



Cryptographic Hash Functions

Digital Fingerprints for Data

- General Properties
  - Maps Input **x** of any size to an Output of fixed size – called a ‘Hash’
  - Deterministic: Always the same Hash for the same **x**
  - Efficiently computed

- Cryptographic Properties
  - Preimage resistant (One way): infeasible to determine **x** from Hash(x)
  - Collision resistant: infeasible to find and **x** and **y** where Hash(**x**) = Hash(**y**)
  - Avalanche effect: Change **x** slightly and Hash(**x**) changes significantly
  - Puzzle friendliness: knowing Hash(**x**) and part of **x** it is still very hard to find rest of **x**

Timestamped Append-only Log - Blockchain

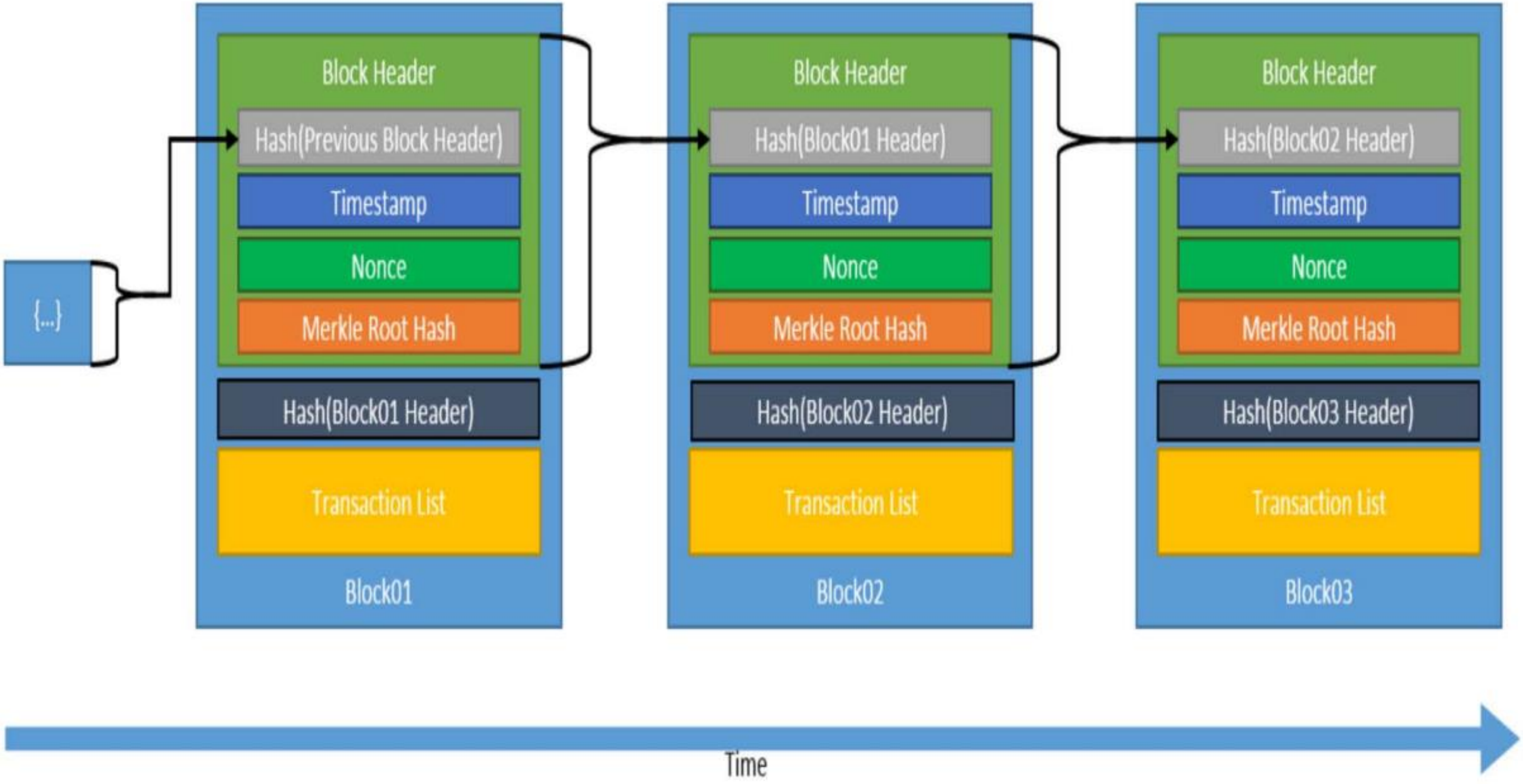


Image is in the public domain by National Institute of Standards and Technology.

- Block header:
  - Version
  - Previous block hash
  - Merkle root hash(hash for the current block)
  - Timestamp
  - Nonce

Merkle Tree – Binary Data Tree with Hashes

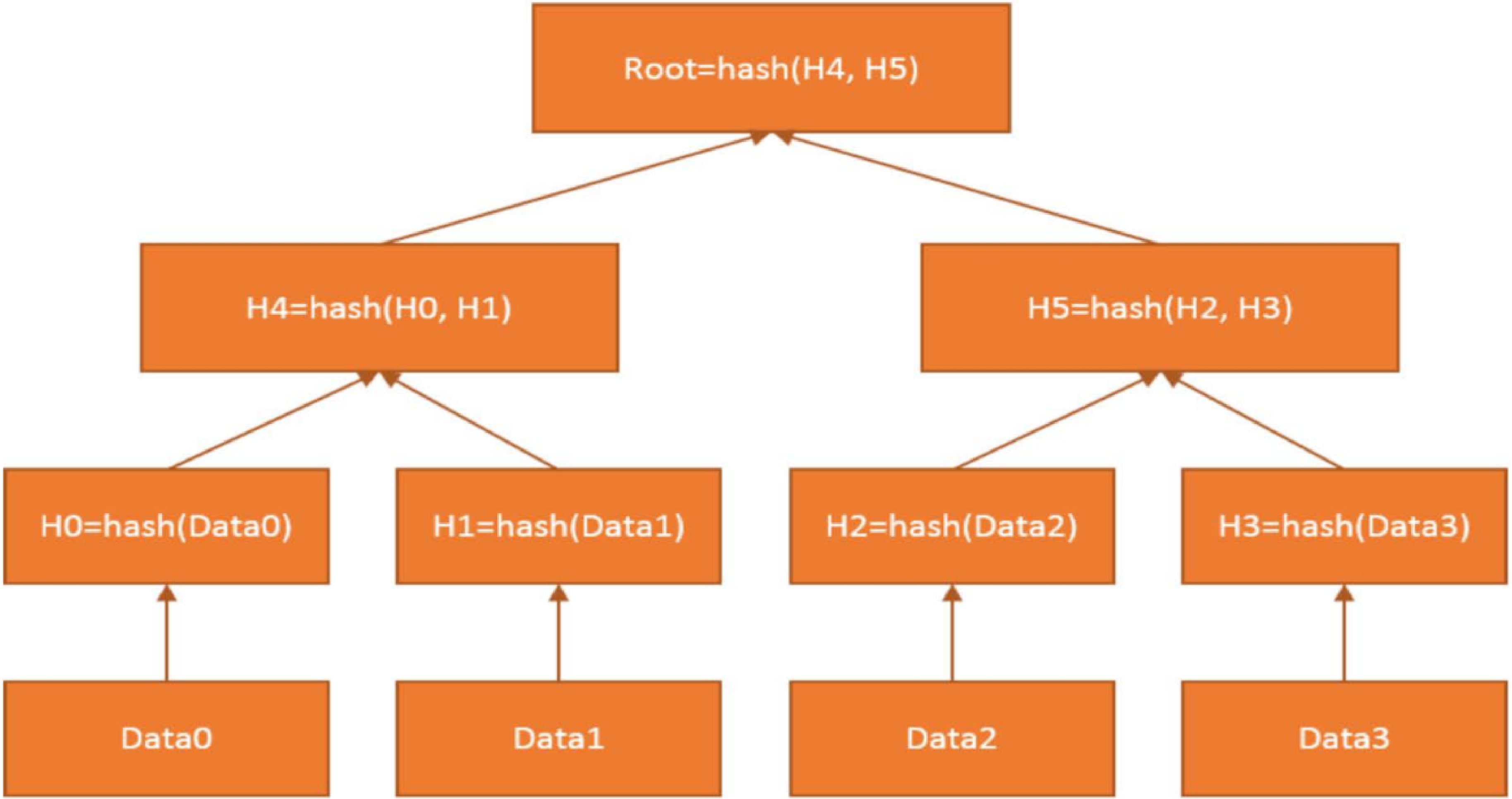
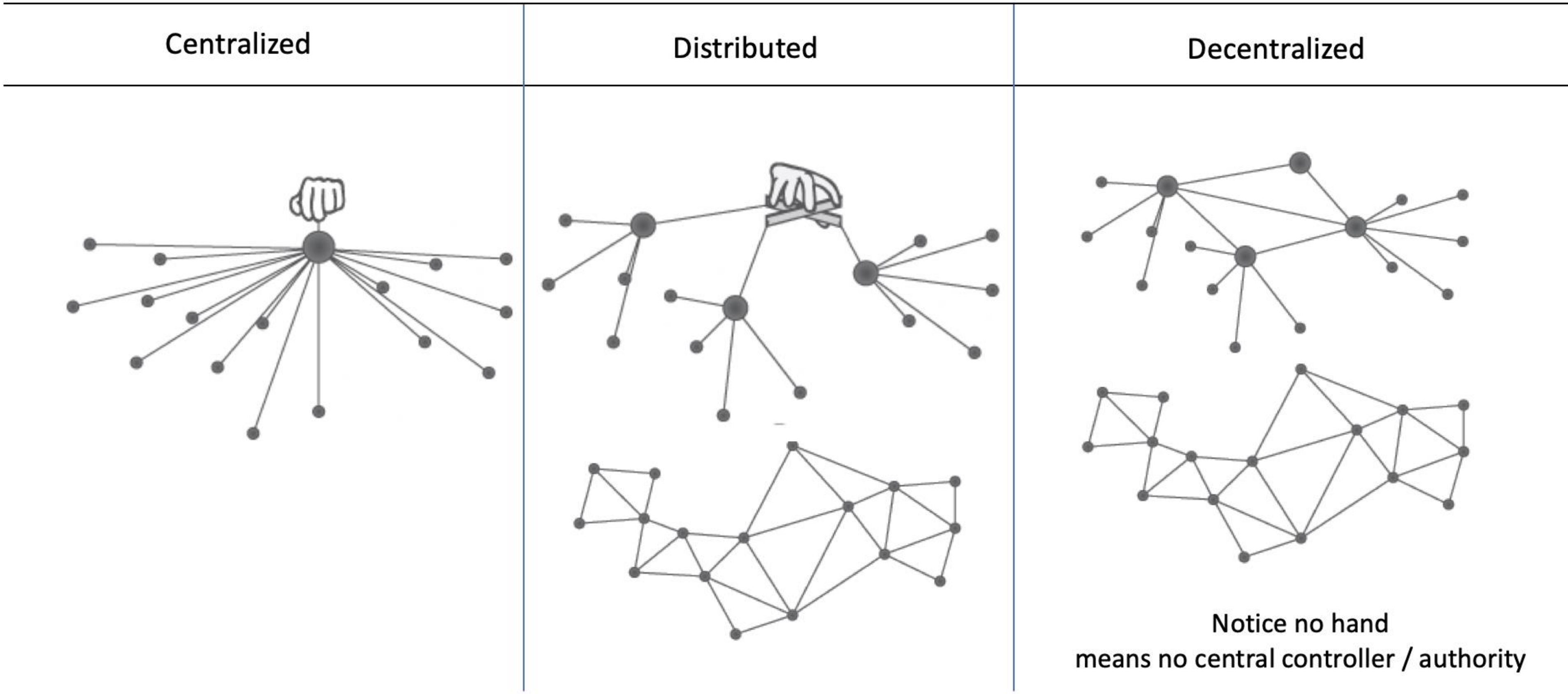


Image is in the public domain by National Institute Standards and Technology.

Decentralization using Blockchain



Centralized

Distributed

Decentralized

Notice no hand
means no central controller / authority

Decentralization using Blockchain

- **Centralized systems** are conventional (client-server) IT systems in which there is a single authority that controls the system, and who is solely in charge of all operations on the system.

- All users of a centralized system are dependent on a single source of service.

Decentralization using Blockchain

- In a **distributed system**, data and computation are spread across multiple nodes in the network . Computation may not happen in parallel and data is replicated across multiple nodes that users view as a single, coherent system

Decentralization using Blockchain

- A **decentralized system** is a type of network where nodes are not dependent on a single master node; instead, control is distributed among many nodes.

Required Reading

- Blockchain 101 – A Visual Demo: https://www.youtube.com/watch?v=_160oMzblY8&t=183s
- Chapter 1 from “Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more, 3rd Edition”. Imran Bashir. Packt Publishing