UNIVERSITY of NICOSIA

Week 2 – Session 3

Databases vs. Blockchain Architectures

BLOC 512: Blockchain Systems and Architectures

Session Objectives

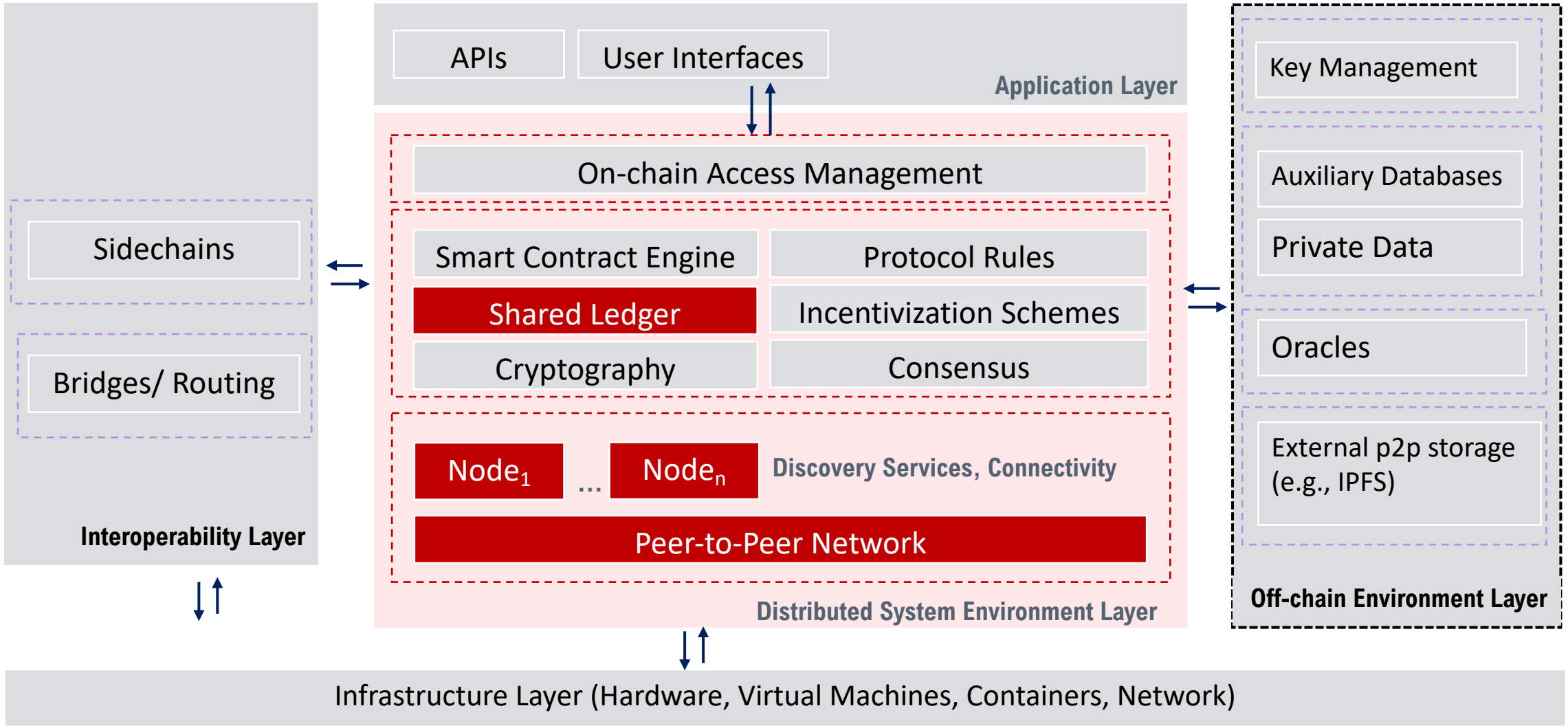
- The uprise of Bitcoin with the idea of blockchains has opened many interesting and challenging research topics in the areas of databases and distributed systems.
- The fundamental component of such systems is the proposal of a replicated data structure (i.e., the blockchain) that is shared among all participants. This data structure ensures data integrity and guarantees a consistent view of all user transactions to all users.
- In this session we will discuss several aspects on databases architectures and distributed systems that influenced recent innovations in DTLs and blockchain-based systems.

Agenda

1. Data Sharing Models
2. The Architecture of Trust
3. Databases vs. Blockchains
4. Deployments
5. Conclusions
6. Self-Assessment Exercises and Further Readings

Flashback

DLT Layers & Architectural Components



Data Sharing Models

Data Sharing

- From the literature we can distinguish different mechanisms for sharing data considering the following systems:
  1. Single System (File-storage)
  2. Centralized Databases
  3. Distributed Databases
  4. Linked data
  5. Peer-to-Peer Storage Systems
  6. Decentralized ledger-based Systems (e.g., blockchain-based + incentives)

Data Sharing: Decentralized ledger-based systems

- We have seen blockchain-based DLTs that are characterized as a p2p distributed ledger in which records called blocks are linked and secured using a cryptographic hash.
- By design these systems are:
  - Decentralized, secure, immutable (governed by consensus)
  - and fault tolerant

- However, can DLTs adapted in design to accommodate:
  - Decentralized storage of large volumes of data? (Data Sharing)
  - Used for distributed processing of large volumes of data? (Distributed Processing)
  - Efficient data look-up capabilities? (e.g., query capabilities?)

Data Sharing: Blockchain-based Systems

- As we noted through the course, blockchains are considered as a distributed data structure and a governance system (i.e., the protocol, governed by consensus)
  - Data Organization
    - A reversed linked list of blocks (i.e., the ledger)
    - The ledger provides an ordered list of events
    - Blocks: indicate the data definition (i.e., schema of the data)
  - Read/Write
    - Data are transparent for everyone to read (some restrictions in case of private settings)
    - The protocol dictates **how** the data are made persistent on the data structure
    - Append only structure
  - Data Integrity
    - Consensus: making sure that all users see exactly the same state of the data
  - Data Replication
    - Distributed data structure replicated by all peers

How about Data Sharding?

- Data sharding is a scalling solution used by traditional database management systems.
- Sharding is a database architecture pattern related to ***horizontal partitioning*** — the practice of separating one table's rows into multiple different tables, known as **partitions**.
- Each partition has the same schema and columns, but also entirely different rows. Likewise, the data held in each is unique and independent of the data held in other partitions.

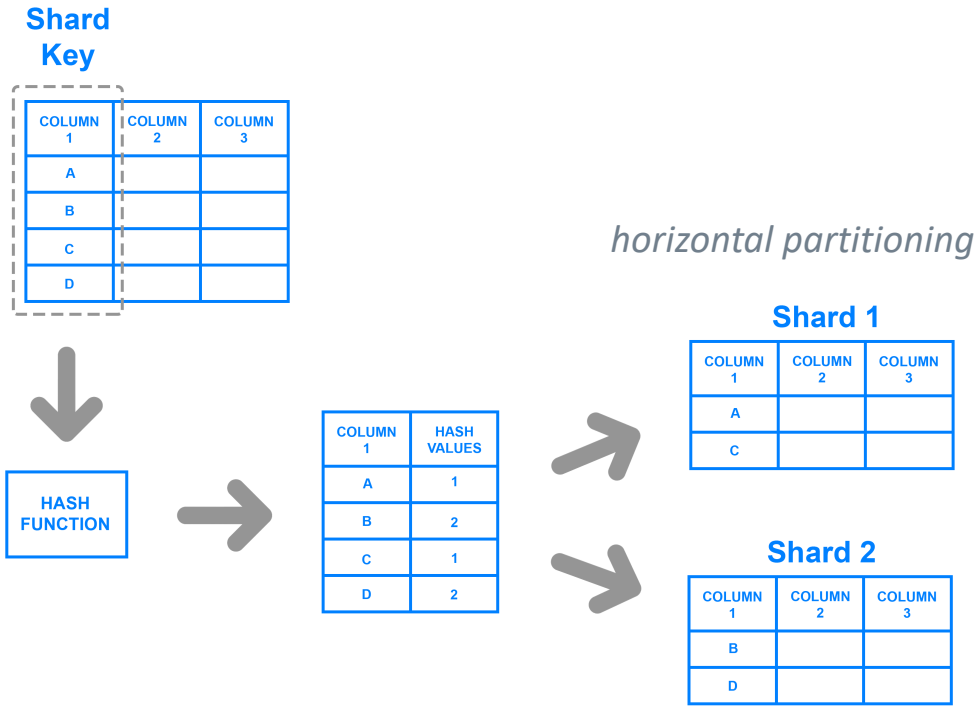


Image Source: Link

How about Data Sharding? (Cont.)

- This concept has been introduced recently to blockchains to harness concurrency across shards.
- Two key challenges for blockchains:
  - (i) how to form a shard, and
  - (ii) how to ensure atomicity for cross-shard transactions.
- A shard formation protocol determines which nodes and data go to which shard. The security of blockchains depends on the assumption that the number of failures is below a certain threshold.
- Sharded blockchains face additional challenges in ensuring ***atomicity*** because the coordinator cannot be trusted under the Byzantine failure model
- To overcome this, Eth2 introduces a separate chain running Casper consensus known as the Beacon Chain that aims to coordinates cross-shard transactions.
- Interesting Polkadot vs Ethereum 2.0 sharding concepts (here)

The Architecture of Trust

Introduction

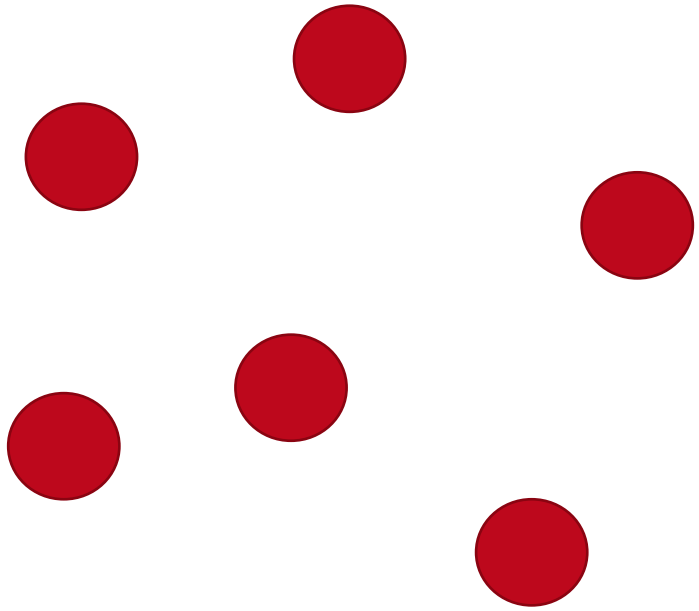
- One of the main challenges Bitcoin addresses is to maintain a consistent view of the replicated data structure (i.e., the blockchain). It does so in a secure and fault-tolerant manner assuming a highly adversarial environment in the presence of malicious actors.
- Challenges:
  - How to provide reliability of services in such an environment?
    - In a permissioned (private) environment the set of participants are known *a priori* whereas in a permissionless (public) environment the topology of the network changes dynamically
    - Participants are free to join or leave the network at any time.
  - How are users incentivized to support and secure the network?
  - How to support distributed transactions? and proof of ownership?
  - How to ensure that the ledger is immutable since everyone holds a copy?

- To address these challenges, Bitcoin builds on foundations developed from diverse fields:

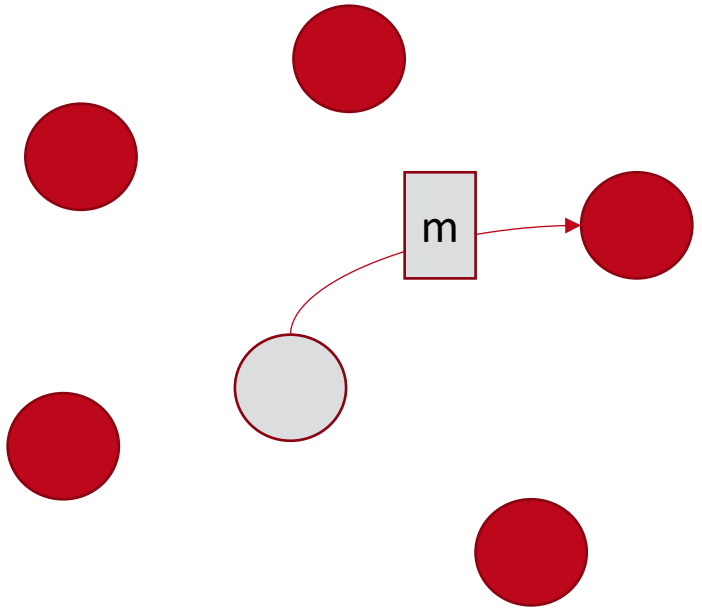
Cryptography | Distributed Systems | Data Management

Re-establishing the Trust Layer

- Let's assume a decentralized network of peers



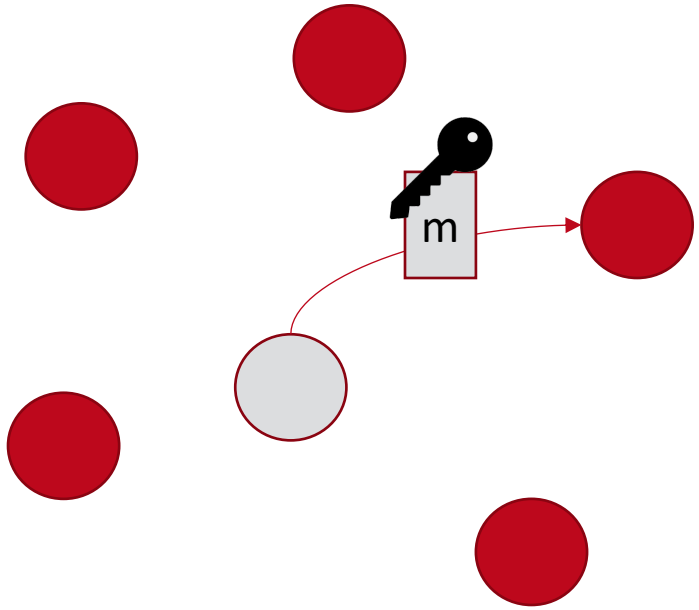
- Nodes are untrusted and unknown to each other. How can we enable them to exchange information (in the form of messages) in a ***trusted*** manner?



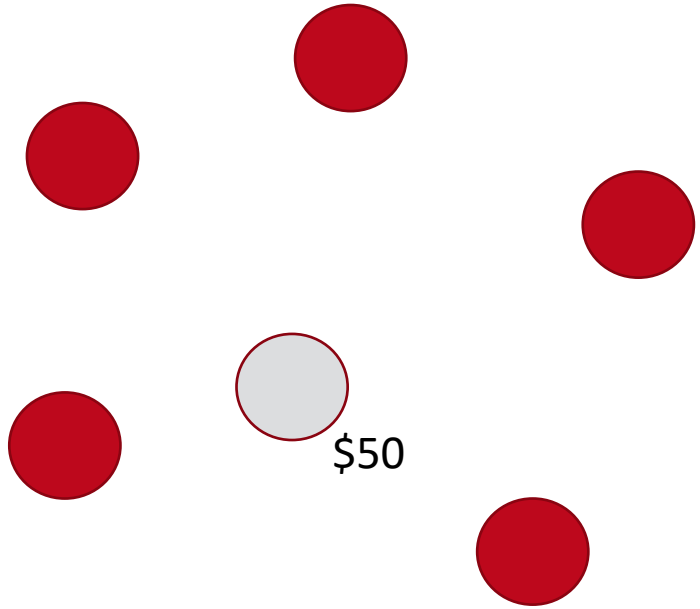
Do we need a blockchain to do that?

Re-establishing the Trust Layer

- Ok, let us introduce some cryptography to sign the message with a private key.

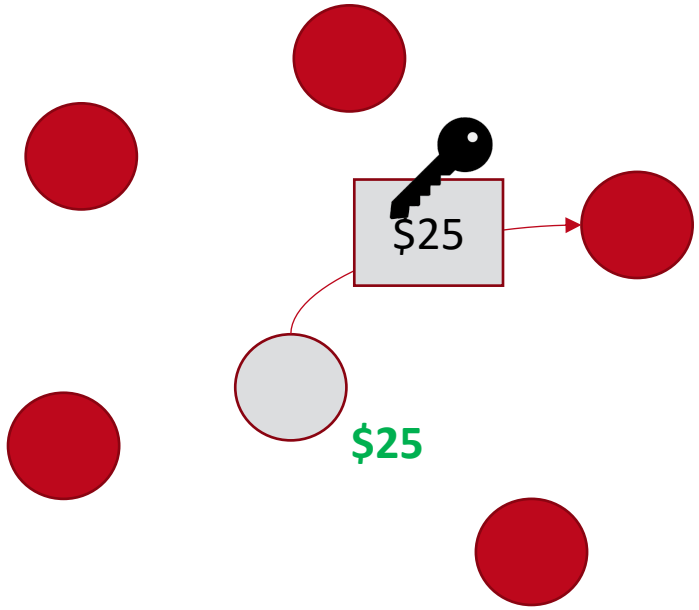


- The recipient of the message can then verify the signature using the public key of the sender.
- Once the message has been proven to be signed then we know that it was not modified.
- Assume that instead of some message we would like to send some amount of funds to some node?

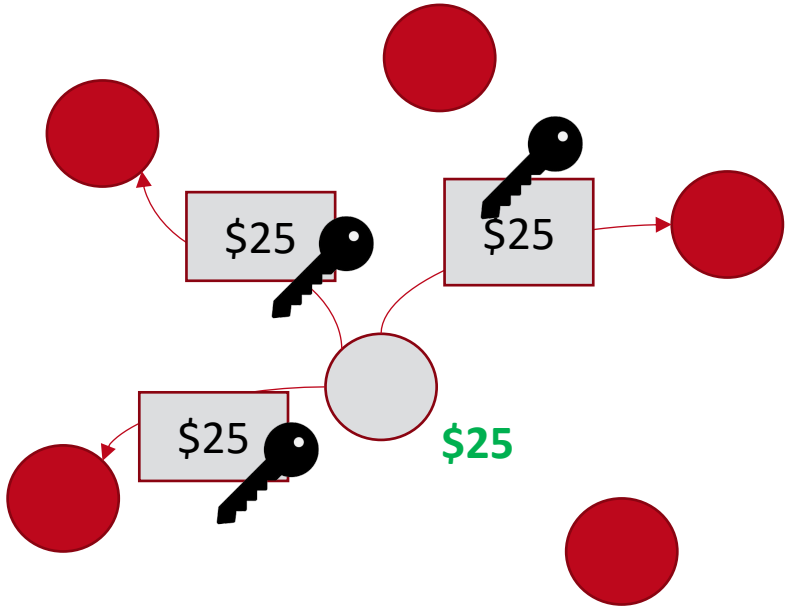


Re-establishing the Trust Layer

- We sign the transfer with our private key, to also confirm its existence



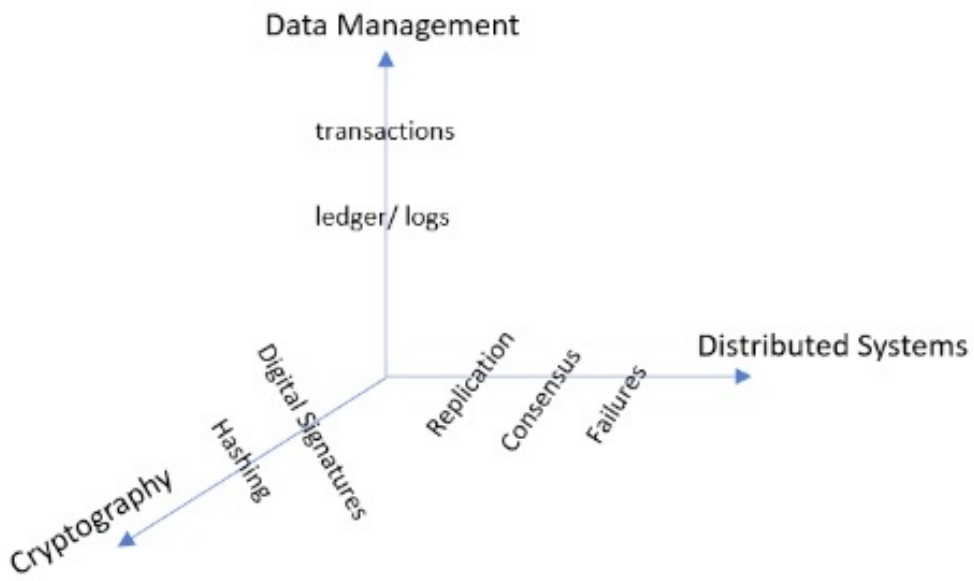
- Consider this: What prevents us from double-spending the funds?
- How can we prevent such disputes from happening?



We need a blockchain to do that

Building a Trust Machine

- Paper [1] summarizes the space of techniques from the various research fields used to implement DLT/blockchains.



Cryptography
- Use of cryptographic techniques, hashing, public-key infrastructure, Merkle trees, and timestamping techniques.

P2P Network
- Network for peer discovery and data sharing

Validity/ Protocol Rules
- A set of rules to confirm the validity of transactions, how the ledger evolves

Ledger
- The data structure consisting of transactions organized into some structure e.g., blockchain or DAG

Consensus
- Fault-tolerance support, determining the order of transactions

[1] Maiyya, S., Zakhary, V., Amiri, M. J., Agrawal, D., & El Abbadi, A. (2019, June). Database and distributed computing foundations of blockchains. In Proceedings of the 2019 International Conference on Management of Data (pp. 2036-2041).

A Trust Layer

- What is trust?

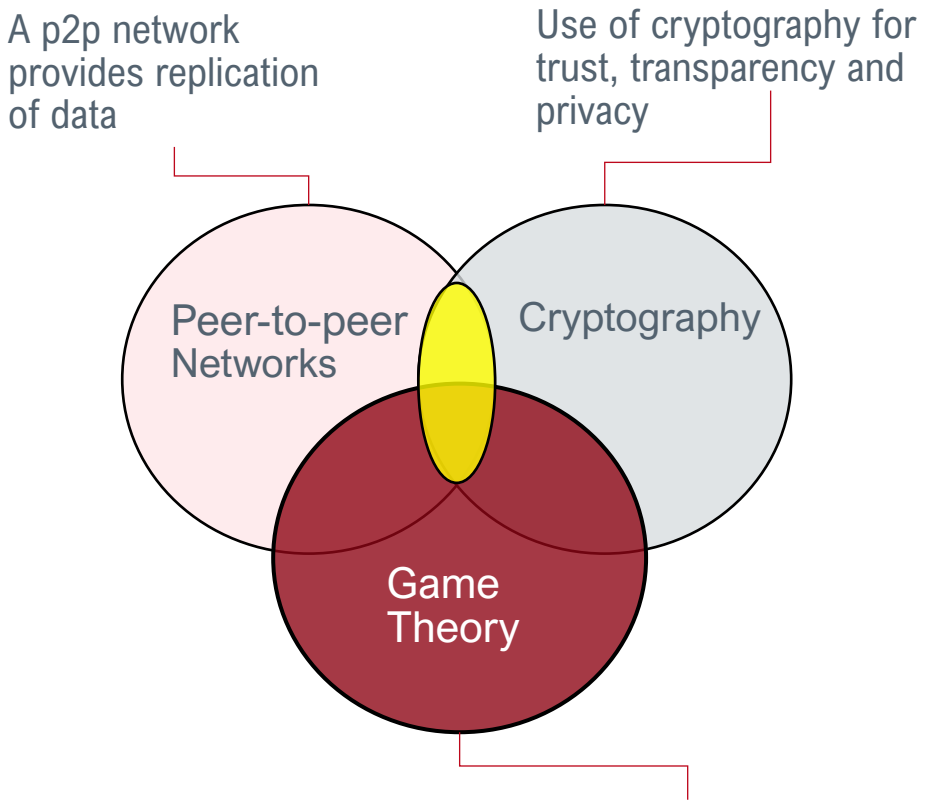
  "Trust refers to the willingness of a party to be vulnerable to the

  actions of another party based on the expectation that the other party

  will perform an action important to the trustor, irrespective

  of the ability to monitor or control that other party [1]"

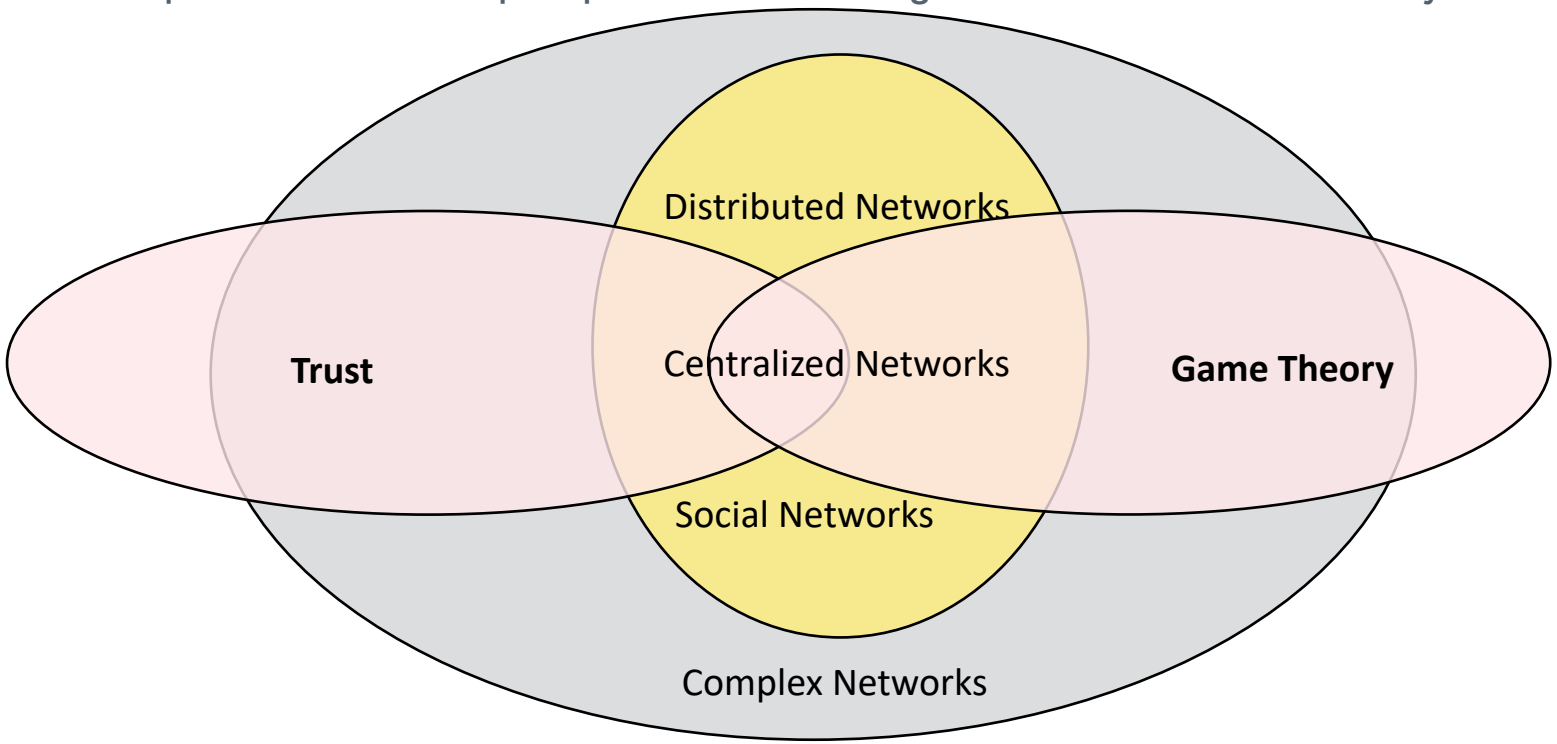
- Nakamoto proposed, in part, a proposal for reconsidering ***trust*** and redefined "the politics" in achieving agreements between untrusted parties



[1] Schoorman, F. D., Mayer, R. C., & Davis, J. H. (2007). An integrative model of organizational trust: Past, present, and future.

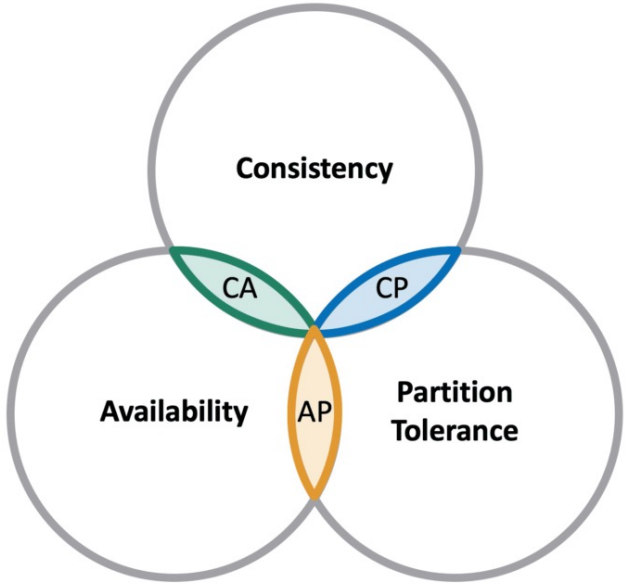
Building a Trust Machine (Cont.)

- But wait there is more….
- Blockchains proposed a complex network for establishing trust. How do we position DLT/blockchains in the general picture of building a trust machine?
- Blockchains encapsulate Satoshi's perspective of building trust over the Internet Layer.



The CAP Theorem

- CAP theorem states that a distributed data store cannot simultaneously be Consistent, Available and Partition tolerant.
  - **Consistency:** What you read and write sequentially is what is expected.
  - **Availability:** the whole system remains reliable; every non-failing node always returns a response.
  - **Partition Tolerant:** The system continues to function and uphold its consistency/availability guarantees despite network partitions.
- *In blockchains, consistency is sacrificed in favor of availability and partition tolerance.*
- *To correctly implement a CP system, a way of achieving distributed consensus is needed – e.g., Raft and Paxos.*
- A detailed proof of this Theorem is outside the scope of this course (more details here)



Source here

DLTs as Distributed Systems

- It all goes back to the CAP theorem from Distributed Systems
- DLTs/blockchain-based systems deploy a p2p network for maintaining the following key principles (considering the case of Bitcoin):
  - Data redundancy - each node has a copy of the data structure
  - Check of transaction requirements
  - Organizing and recording transactions in sequentially ordered blocks
  - The creation and distribution of the native cryptocurrency is controlled by a consensus algorithm
  - All transactions are based on cryptographic proofs with the use of public-key cryptography
  - Programmability is embedded with the of a transaction scripting language or a smart contract engine e.g., Ethereum Virtual Machine on Ethereum
  - Embedded incentivization schemes that provide economic (or other) incentives for maintaining the health of the network.

Databases, Distributed Databases …

- The notion of databases has a different design nature than building a trust machine.
- **Suitability:**
  - Blockchains are suitable when the applications are running in untrusted, hostile environments, whereas databases are suitable when performance is more important than security.
  - On the other hand, databases have been designed for storing and processing large volumes of data whereas blockchain are not suitable for data achieving and retrieval.
  - Databases employ optimization algorithms, indexing, sharding, concurrency and other techniques to facilitate a high throughput of transactions per second.
- **Governance:**
  - The main assumption of traditional Database systems is that data are kept under the control of a single possible distributed entity (which can be considered centralized)
  - Whereas the main assumption of blockchains is that participants are unknown to each other, potentially untrusted, and not controlled by the same entity.

- There is however a perspective on blockchains that could be ***evolved*** by introducing several foundations from databases.

Databases vs. Blockchains

Databases vs. Distributed ledgers

Property	Databases	Distributed Ledgers
Authority	- mostly controlled by the database administrator and are centralized in nature - even at a distributed or a federated setting databases are controlled centrally be their internal organizations	- no centralized approach by design, there is no need for a central authority. - exception for private DLTs that may use some form of centralization - consortium DLTs have a variant decentralization degree that is controlled by some governance model
Architecture	- Mostly utilizes a client-server architecture	- architecture builds over a p2p network
Data Handling	- supports Create, Read, Update, and Delete operations aka CRUD	- an append only ledger that support only Read and Write operations
Integrity	- corrupted or malicious actors can alter the data - a single point of failure	- inherently resistant to data modification - uses cryptographic hashes and timekeeping mechanisms to create a proof of history for the data - almost impossible to tamper with

Databases vs. Distributed ledgers (Cont.)

Property	Databases	Distributed Ledgers
Transparency	Administrator is responsible for creating user views on parts of the data. Decides which data remain private and which data are made public	By nature public permissionless DLTs are transparent. For consortium DLTs transparency is offered towards the consortium members only. For private DLTs transparency is restricted
Cost	Mature technology that can be easily deployed in various settings	More complex to deploy and maintain
Performance	Highly scalable and fast	Depends on the verification and consensus protocol, thus comparatively restrictive in terms of throughput and scale. Blockchains introduce a performance penalty because of its verification methods

To Sum-up …

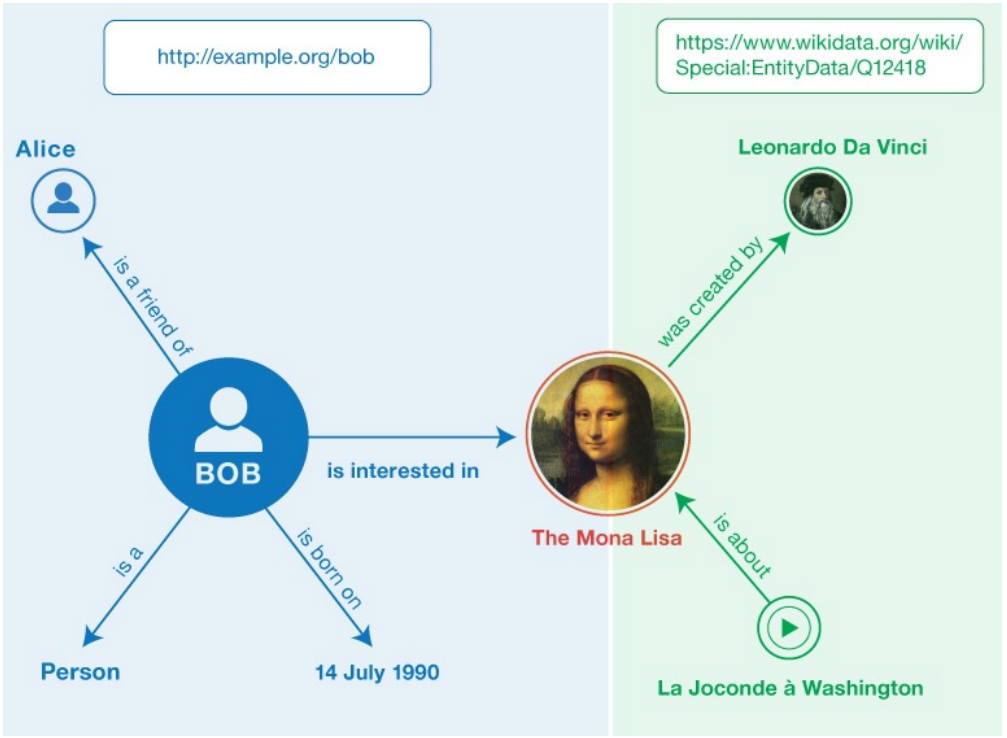
- Main difference of the two is how the concept of authority and control is realized! Blockchains are designed with disintermediation and contest driven decentralization compared to Databases that are highly centralized even when distributed. Without an administrator a database cannot function at all.
- What about their operations?
  - Blockchains have been designed to provide only Read/Write operations whereas a Database provides CRUD operations.
- What about Transparency?
  - Blockchains provide a data structure with integrity in mind (each block is linked with previous with the use of cryptographic hashes)
  - Immutability in Databases depends on some central authority and administration
- What about their architectures?
  - A databases is mostly based on a client-server (2-tier) architecture where a central server acts as the processing unit of the information and the client request some data from the server over the network. We will examine Database Deployments later.
  - On the other hand, DLTs/blockchains are based on a peer-to-peer network (a kind of a Distributed System). Each peer is identified with cryptography and communicates using a secure cryptographic protocol. There is no a centralized node to coordinate the state of the data which is orchestrated by a consensus algorithm.

Types of Databases: An Overview

- **From Hierarchical File Storage to Relational**
  - Databases started as flat file hierarchical systems which provided information gathering and storage. The proposal of the Relational model enabled more complex ways for organizing and querying data. In brief the model proposes that data elements are stored in a Table form. Tables contains columns and rows. The columns represent the type of the data (called fields/attributes) and the rows define the actual records stored in a database
- **NoSQL/Non-Relational Databases:** proposed as alternative to Relational Databases. Main difference is that NoSQL databases can be schema-agnostic, allowing unstructured and semi-structured data to be stored and manipulated - can take a variety of forms:
  - **Key-value stores:** are simple database management systems that store only key-value pairs and provide basic functionality for retrieving the value associated with some key e.g., Redis and Amazon DynamoDB.
  - **Wide column stores:** are schema-agnostic systems that enable users to store data in a multi-dimensional key-value store. These solutions are designed with the goal of scaling to manage petabytes of data man servers in a massive, distributed system e.g., Cassandra, Scylla, and Hbase.
  - **Document stores:** are schema-free systems that store data in the form of JSON documents e.g., MongoDB and Couchbase.
  - **Graph databases:** represent data as a network of related nodes or objects in order to facilitate data visualizations and graph analytics e.g., Neo4J
  - **Search engines:** store data using schema-free JSON documents, similar to document stores, aim to make unstructured or semi-structured data easily accessible e.g., Elasticsearch

Types of Databases: An Overview (Cont.)

- **RDF Triplestores**
  - Is a kind of a graph database that uses the Resource Description Framework (RDF) model for data publishing and interlinking according to the Semantic Web standards. Triplestores are a kind of NoSQL database that store data in “triples” rather than the traditional relational structure.
  - A triple is a data entry composed of subject-predicate-object (s, p, o)
  - The RDF triples allow data elements to be annotated with semantic metadata from Ontologies so that the data can hold semantic facts e.g., Jena, Stardog, AllegroGraph



Source: https://www.w3.org/TR/rdf11-primer

Deployments

Deployments of Database Management Systems

- Database Management Systems (DBMS) are classified in the following architectures:
  1. Single-tier Architecture
  2. Two-tier Architecture
  3. Three-tier Architecture

- The architecture refers to the representation of the DBMS design.
- The components of the DBMS architecture can be divided in various layers or tiers.
- Tiers refer to the layers of the architecture based on the complexity of the user application and how the data are stored and queried.

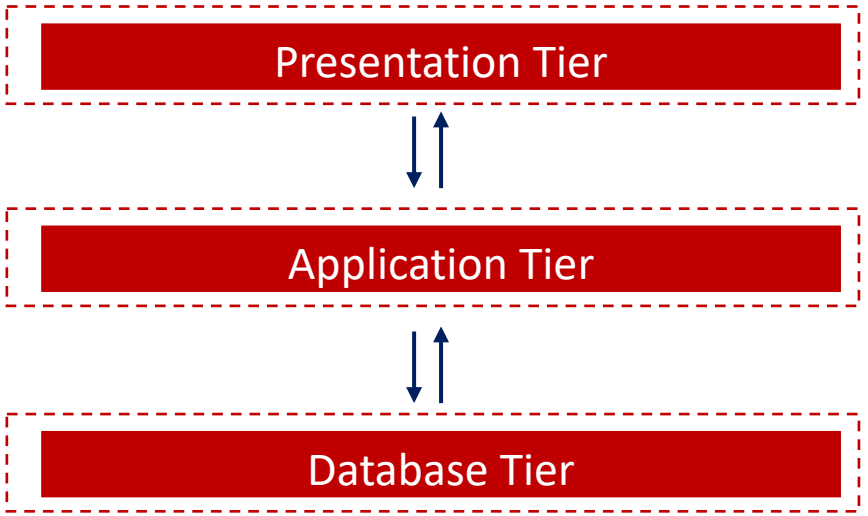


Fig. Deployment Tiers of a DBMS

Centralized Database Architecture

- We present an overview of a Database Management System (DBMS) used later to guide our comparisons on how DLTs can be deployed or become decentralized.
- A DBMS is a software system that comprises of different component to:
  - Define, manipulate
  - Retrieve, and manage

  data that are stored in a database.

- **Highly Centralized:** The client, sever and database are on the same computer. This does not require a network connection to perform actions to the databases
- **Authority:** Blockchains have been designed to work in a decentralized manner whereas database deployments are highly centralized.
- **Transparency:** The admin decides on user views.

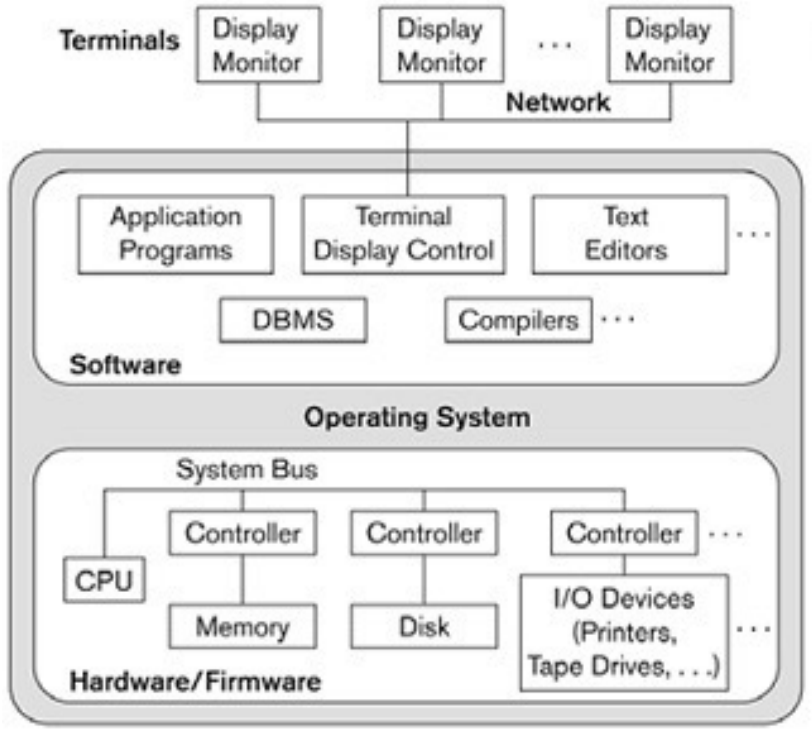
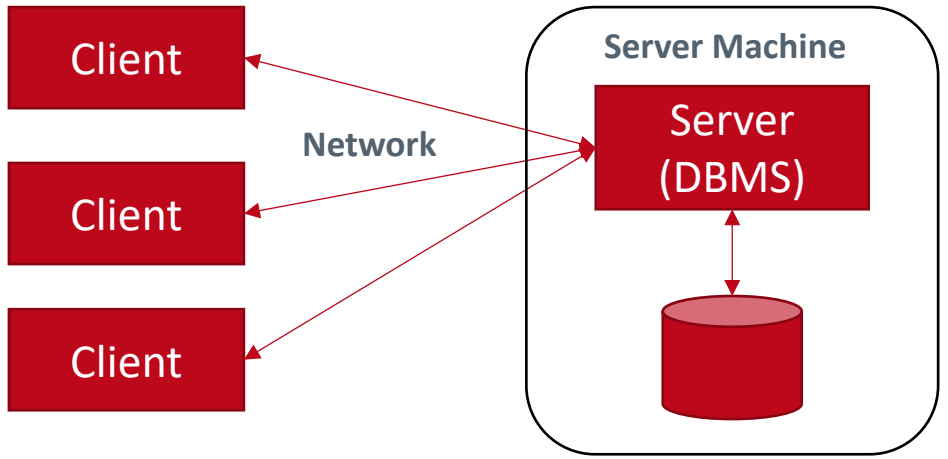


Fig. Physical centralized architecture

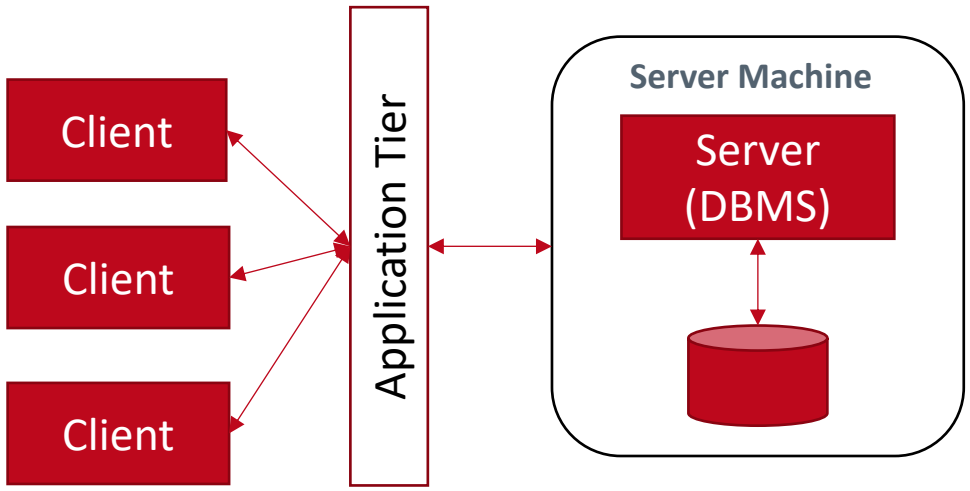
Client-Server (2-tier) Architecture

- In this architecture the Database system is present at a sever machine which is responsible to deliver the data and manage resources
  - **Client:** a piece of software or application that takes an input and sends request to the server.
  - **Server:** a piece of software that receives and processes requests from clients
- This architecture has one or more clients that are connected to a central server.
- Whenever a client requests access to the database via the central server using a query language e.g., SQL, the server perform the request on the database and returns the result back to the client.



3-tier Architecture

- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data from the database. It is the most widely used architecture to design a DBMS.
  - **Presentation Tier (Client):** Users/Clients interact with this layer. The users are not aware about any existence of the database beyond this layer.
  - **Application Tier:** This is where the application is stored (e.g., Web Server) and the programs that access the database. This layer sits in the middle and acts as a mediator between the client and the database.
  - **Database Tier:** This is where the the database resides along with its query processing languages (on some Server Machine)
- Although DBMS have the capability to scale into multiple servers with various techniques still their deployment is highly centralized. All layers need to be trusted, middle layer is acting as an intermediary.



Scaling Databases: The need for distribution!

Client-server vs P2P

- P2P networks are a category of a ***distributed system*** that is instantiated for the exchange of data among multiple peers.
- P2P networks do not depend to some central server for the exchange of data.
- Peers are connected to each other for the purpose of sharing data and communicate directly to each other without the need of accessing some central server.
- P2P networks have been crucial for the development of DLTs including blockchain-based ledgers.
- DLTs have been evolving the idea of p2p networks with:
  - incentivization schemes
  - protocol rules governed by consensus

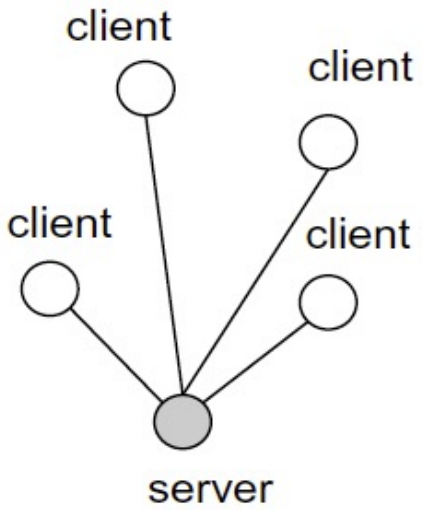


Fig. Client-server model

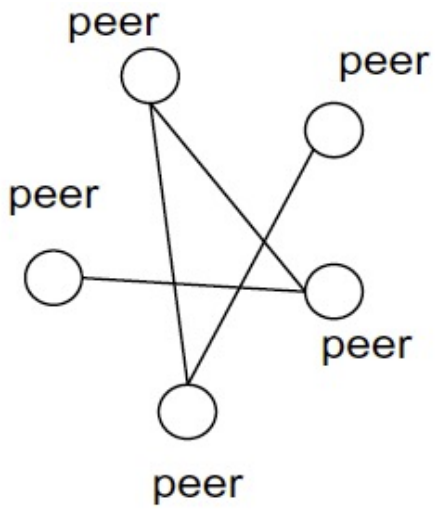


Fig. Peer-to-Peer model

Characteristics of P2P networks

Symmetric Role

- Such networks provide a democratic degree on user roles. Each peer can act both as a client or as a server. Peers are capable of handling a dual role and responsibility in the network.

Distribution

- Peers are likely to be distributed to many different locations and not centralized under some network.

Decentralized

- In the most ideal case, there is no centralized control for managing these systems.

Scalable

- Such networks can be highly scalable, since are not tightly coupled with the computational resources of some server machine.

Heterogeneous

- Peers are considered heterogeneous; in terms of the hardware and the operating system used by the peers that participate to such networks.

Dynamism

- Typically, such networks are highly dynamic. This means that the topology of the network can change rapidly due to network or byzantine faults.

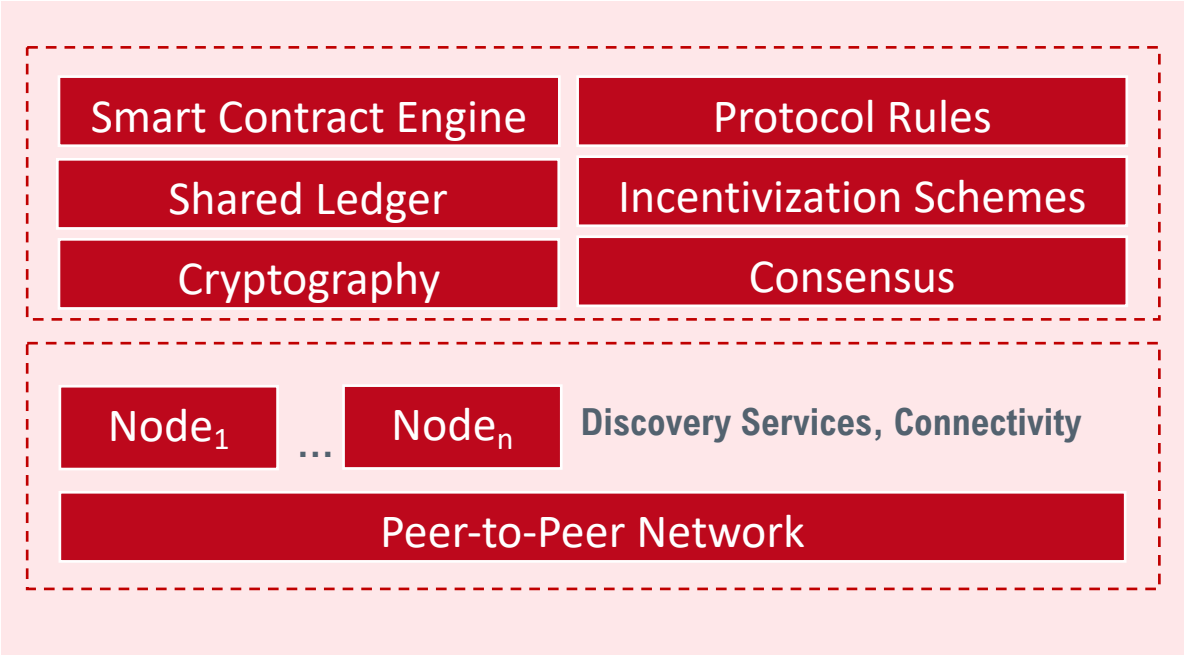
Applications of P2P Networks

- The infrastructure provided by p2p networks can be instantiated to support a wide spectrum of applications:
  - **File sharing:** p2p networks provide a serve-less network for sharing data. Such networks allow peers to download files from other network participants, and also enable peers to select a set of files from their local storage to be shared to the network with other participants e.g., Gnutella, BitTorrent, Freenet, and Napster.
  - **Video Streaming:** Another interesting application of p2p networks is to enable video streaming among peers. In such a network each peer can start a streaming process e.g., Zatoo, PPlive, Tribler
  - **Cloud Computing:** In this setting p2p networks are utilized to increase the scalability for clouds.
  - **Grid Computing:** In this setting p2p networks can be used as an infrastructure for sharing computational resources.
  - **P2P Databases:** In this setting p2p networks can be used as an infrastructure for supporting server-less and distributed databases e.g., OrbitDB and Barrel.
  - **Search Engines:** A computing paradigm introduced as an alternative to centralized search engines for querying and ranking Web content e.g., YaCy
  - **P2P Storage:** To be used as the infrastructure for supporting storage and retrieval operations for remote files among several peers e.g., IPFS

Applications of P2P Networks

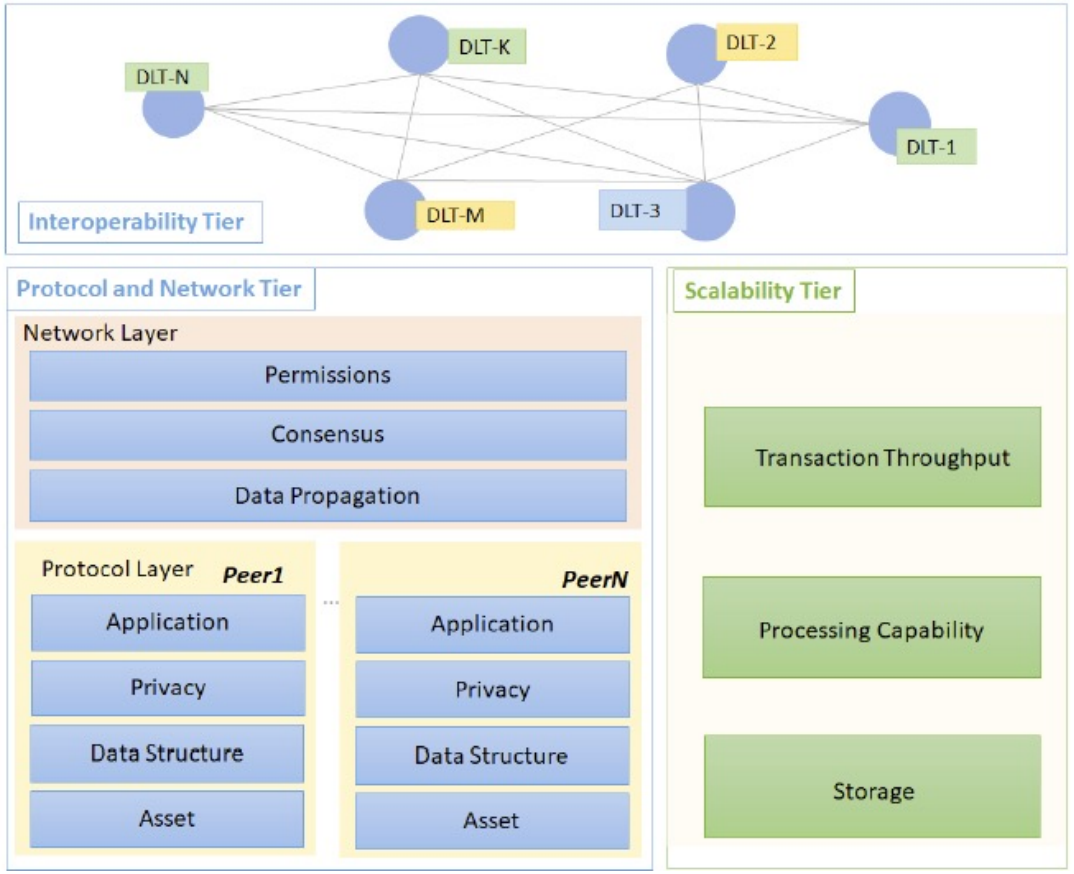
- **Anonymized Communication:** Utilizing cryptography and a p2p network to enable users to communicate in an anonymous way e.g., Tor
- **Hosting Services:** P2P networks are instantiated to deliver Web content e.g., WebRTC
- **Cryptocurrencies:** P2P networks to facilitate the distribution and settlement of financial transactions

Bitcoin was the first instantiation of a **p2p network** instantiated to facilitate (**through an incentivization scheme**) the integrity of a distributed and decentralized data structure (i.e., the blockchain) to many ***untrusted*** parties under ***certain conditions*** embedded as protocol rules



Three-tier architecture for dApps development

- DLTs/blockchains provide an infrastructure for developing decentralized applications (dApps) with no central authority for registering, sharing, and synchronizing data.
- Similar to the 3-tier architecture of databases a three-tier based architecture for DLT applications is proposed [1] for the development of dApps.



[1] Antal, C., Cioara, T., Anghel, I., Antal, M., & Salomie, I. (2021). Distributed ledger technology review and decentralized applications development guidelines. *Future Internet*, *13*(3), 62.

Methods of Decentralization

Towards Decentralization …

- Moving from centralized systems, where a single authority controls the systems, to a more distributed architecture.
- With a Distributed System data and computation are spread across multiple nodes in the network. However, with a Distributed System there still exists a central authority that governs the entire system; whereas, in a fully decentralized system, no such authority exists.
- A decentralized system is a type of network where participants (e.g., nodes) are not dependent on a single master node or peer; instead, in such systems control is distributed among many nodes.
  - Decentralization could be achieved with alternative governance models
  - Consensus can offer such a model where users agree on something via the algorithm without the need for a central, trusted third party, intermediary, or service provider.
- **Change of values**
  - **Disintermediation:** decentralization is achieved by disintermediation of central power. For Smart Contracts refer also to the idea of Decentralized Autonomous Organizations.
  - **Contest-driven decentralization:** Nakamoto consensus aims to achieve decentralization through competition among participants to be selected for the provision of services.

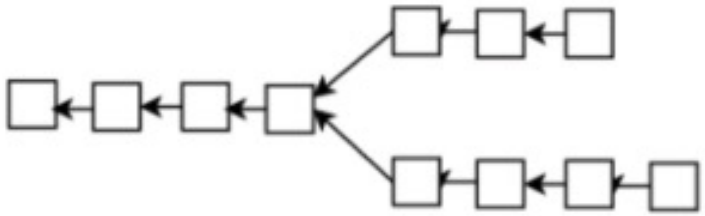
Alternative Data Structures

Intro to alternative Data Structures

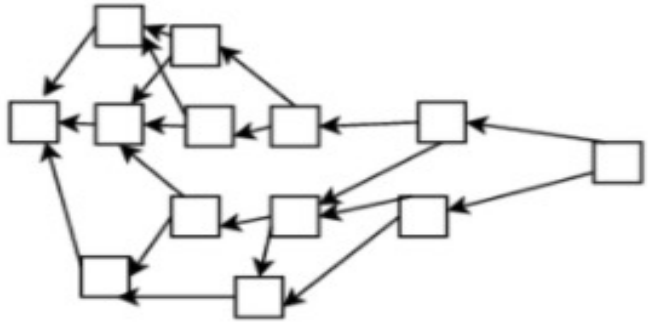
- As we discussed in previous sessions, blockchains are a kind of a distributed ledger but not all distributed ledgers use a blockchain as their underlying data structure.
- The main concept with DLTs is that they are considered as trust machines established in an inherently trustless environment without the need of a central authority. However, there are several ways DLTs can be instantiated for organizing the underlying distributed data structure.
- One major challenge for blockchains are its poor scalability which makes it unable to process large volumes of transactions. Therefore, several alternatives have been proposed in the literature that would be more efficient than a blockchain.

Directed Acyclic Graphs

- From Graph theory a Directed Acyclic Graph (DAG) is a graph that consists of vertices (also called nodes) and edges, with each edge directed from one vertex to another, such that following those directions will never form a closed loop.
  - A property that makes DAGs appealing for creating DLTs is their topological ordering property (meaning that its growth can go from one direction - from earlier blocks to the later)
  - Blocks in this structure are not arranged in a sequential order one after the other like in [a], instead of having just one parent node in this structure nodes can link to more than one. However, Each node in the DAG (called site) represents a ***transaction***, and the connections (direct edges) between transactions represent the ***validations of transactions.***
  - Transactions are submitted to the DAG by nodes, much like on a blockchain.
  - Examples of this concept is IOTA and Obyte



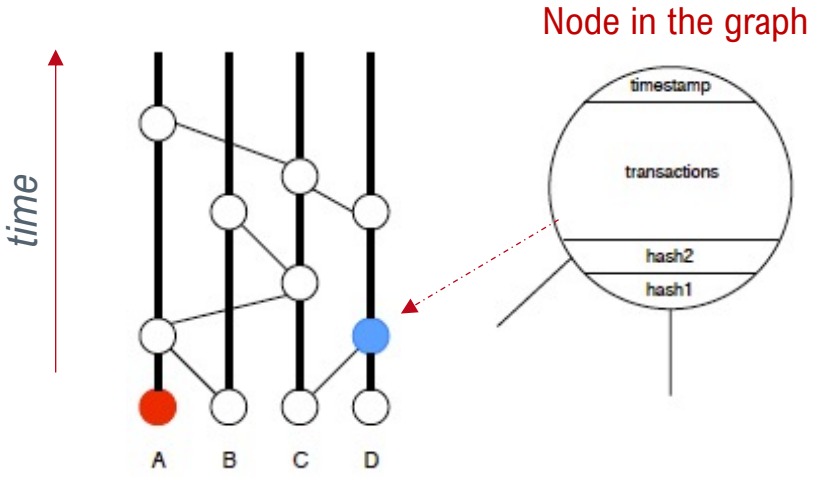
[a] Abstract view of a blockchain



[b] Abstract view of a DAG

time

Directed Acyclic Graph with a Gossip Protocol



[c] Abstract view of a Hashgraph which is also a DAG

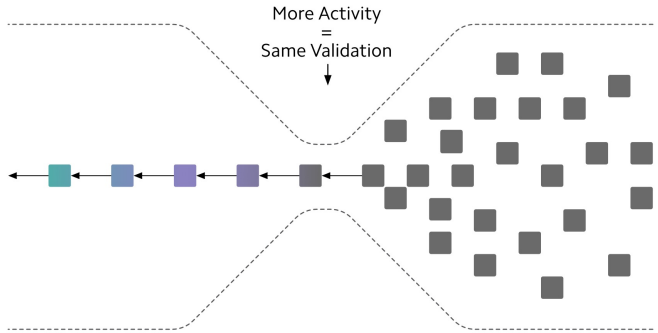
- A hashgraph is an alternative to a blockchain structure
  - Hashgraph uses a DAG as data structure for storing transactions, and a voting algorithm combined with gossip protocol to quickly reach consensus among nodes
  - In real breakthrough of this structure is the consensus protocol which has been proven to help replicate data in a faster fashion compared to blockchain.
- This protocol allows users to:
  - **Submit a transaction:** users can submit an event that contains a new transaction (red vertex)
  - **Gossip about a transaction:** users can randomly choose other users and tell them what they know. For instance, in Fig. [c] the user C has submitted an event, then it gossips to user D about it. This process allows for an exponential diffusion of information about newly submitted transactions.
  - Example of this concept the **Hashgraph** proposal [1]

[1] Baird, L. (2016). The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep.

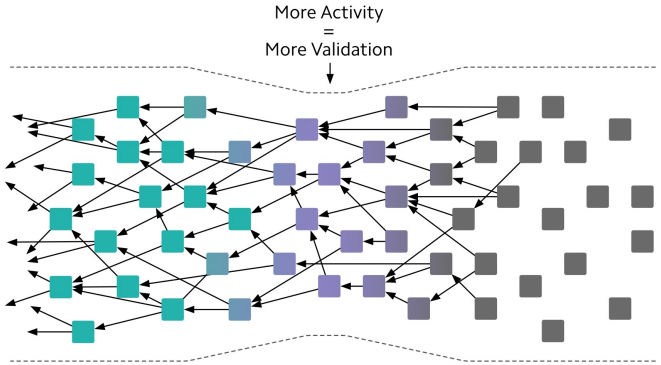
What is the value of DAGs?

However, the potential of DAG is being recognized. Vitalik Buterin praised the concept of directed acyclic graphs admitting that they can reduce the network latency: "…they do have some value, particularly in reducing latency, so basically you can design systems where the latency goes down from something like Ethereum's 14 seconds possibly to 1 second…"

THE BLOCKCHAIN BOTTLENECK



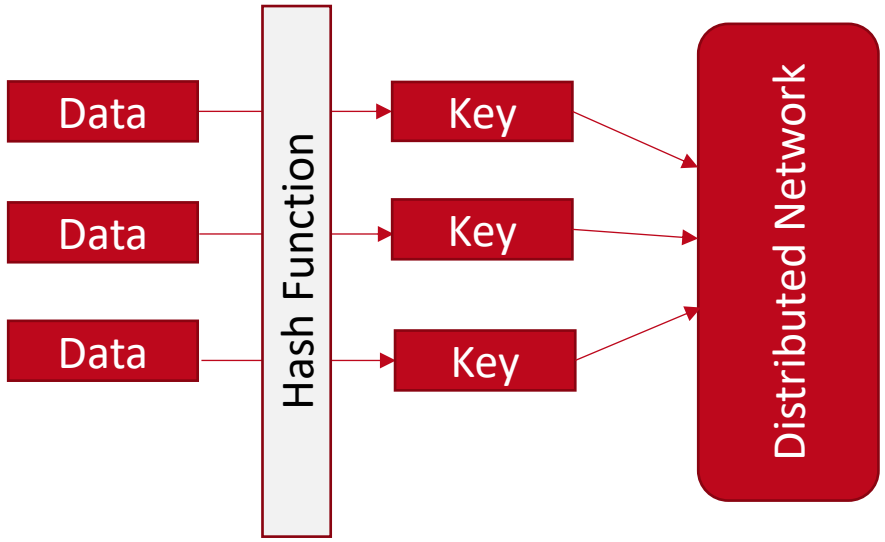
THE IOTA TANGLE SCALES!



Source: https://blog.iota.org/on-the-tangle-white-papers-proofs-airplanes-and-local-modifiers-44683aff8fea

Distributed Hash Tables (DHTs)

- DHTs are tables of key-value pairs stored in overlapping shards among network peers; any participating node can efficiently retrieve the value associated with a given key.
- They provide a highly efficient and generalizable infrastructure for managing data and carrying out routing across large-scale peer-to-peer networks.
- DHTs as such constitute a form of sharded validation (what Ethereum is looking to eventually accomplish with PoS+sharding, known as Ethereum 2.0) where participants store only those shards they happen to validate.
- Notable Examples:
  - Distributed system that uses DHT for file sharing are IPFS and BitTorrent.
  - DLT that makes use of a DHT is Holochain



Fusion of Blockchains and Databases

Examples: BigchainDB

BigchainDB [1] it is a great example of how the databases (i.e., high transaction rate, low latency, indexing & querying of structured data) could be enhanced with blockchain properties (e.g., decentralization, immutability, owner-controlled assets). BigchainDB's state machine is specialized for registering and tracking the ownership of "assets".

Bigchain DB Service Node

Tendermint:

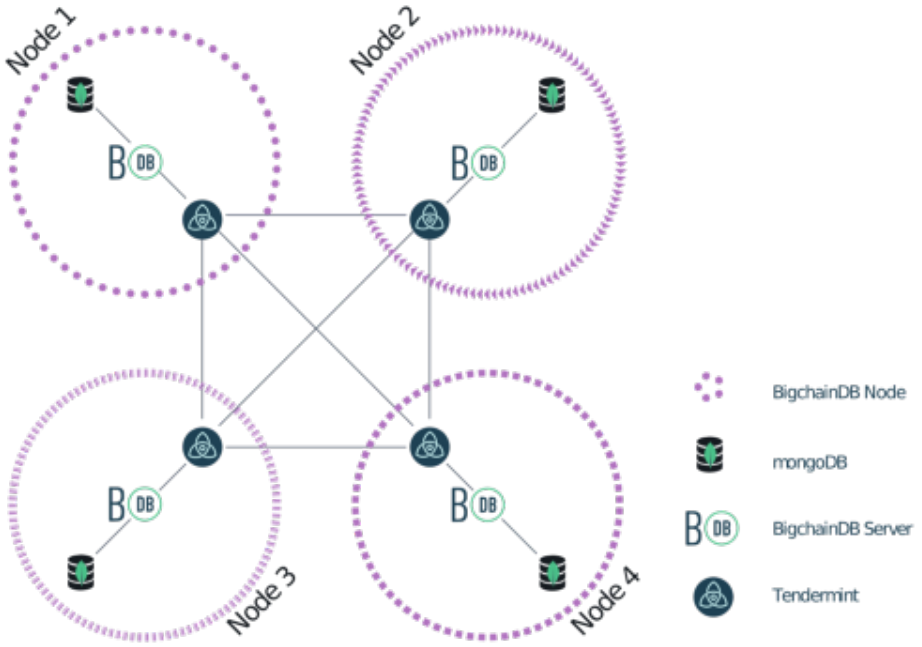
- handles the intra-cluster communication
- Replication, voting, and consensus logic is handled by Tendermint, not by MongoDB or BigchainDB.

MongoDB:

- MongoDB is used to store state and query state. You can think of the MongoDB database as giving BigchainDB a built-in "blockchain explorer.

BigchainDB:

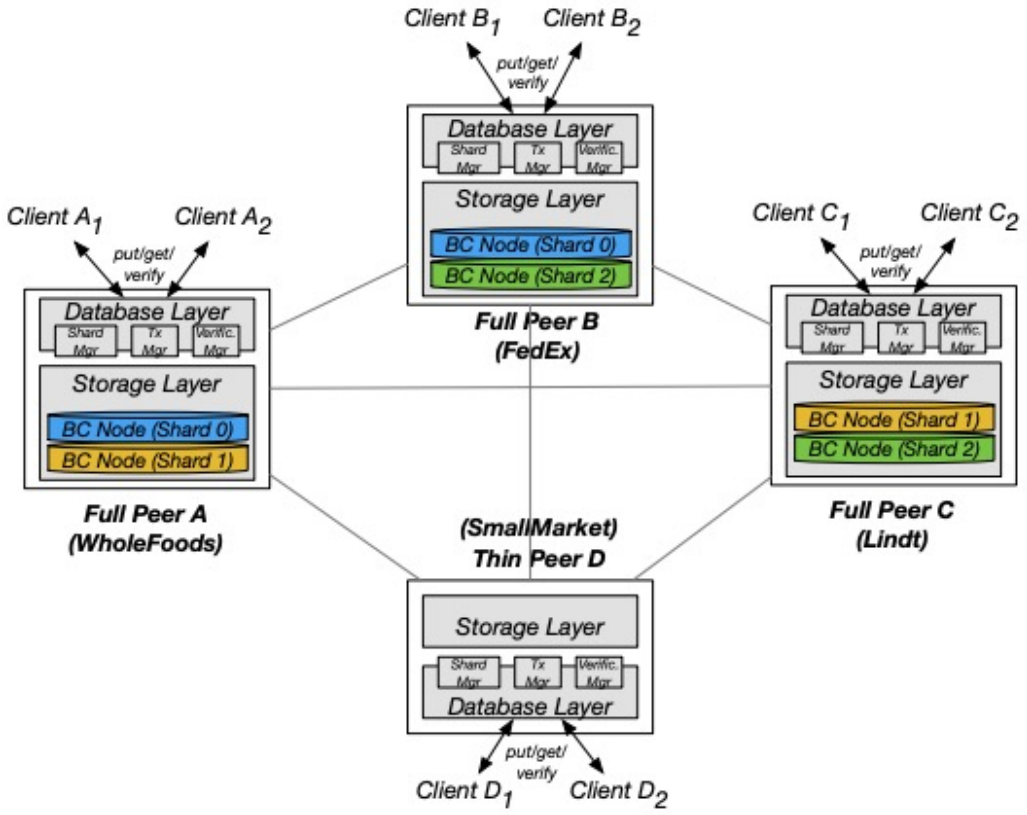
- accept new transactions via an API
- bundles transactions in blocks and validate them
- allows the creation of JSON-like objects called BSON objects (with support to timestamps, dates and binary objects)



[1] BigchainDB 2.0 the blockchain database[Online], Available: https://www.bigchaindb.com/whitepaper/

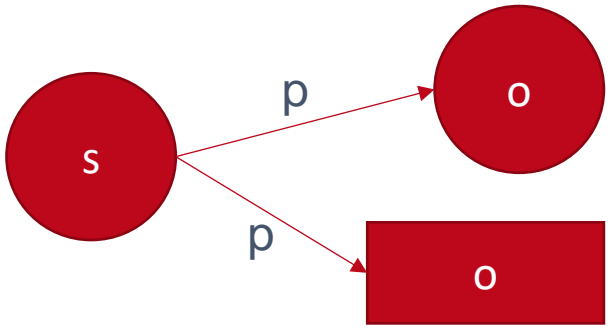
Examples: BlockchaiDB

- BlockchainDB introduces:
  - a blockchain as a storage layer
  - a database layer on top that extends blockchains with data management techniques like Sharding
  - a query interface
- The key idea of BlockchainDB is that data is not replicated to all peers to avoid the high overhead of blockchain consensus. Instead, shared tables are partitioned (i.e., sharded), thereby each shard is implemented as a separate blockchain network [1].



[1] El-Hindi, M., Binnig, C., Arasu, A., Kossmann, D., & Ramamurthy, R. (2019). BlockchainDB: A shared database on blockchains. Proceedings of the VLDB Endowment, 12(11), 1597-1609.

Examples: GraphChain



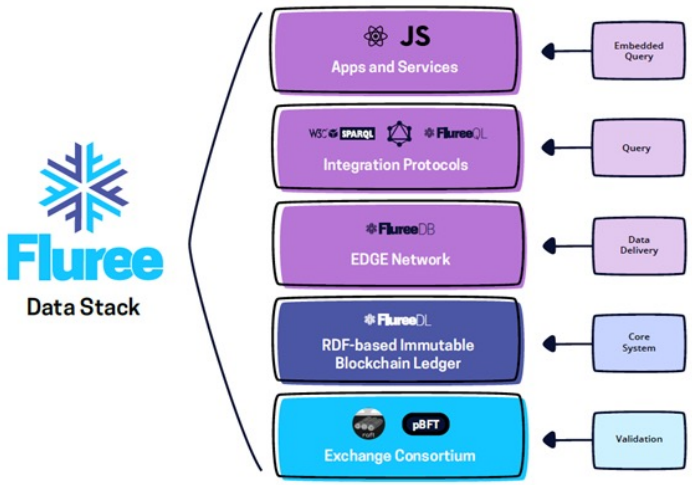
Vertices:
- Resources represented as URIs
- or predicate/attributes values

Edges:
- Predicate/attributes represented as URIs
- Represent relationships

- It is one of the first projects to show the synergy between blockchain and Semantic Web technologies.
- Graphchain proposes a framework for on-chain data management with the use of RDF graphs.
- RDF Graph: is an unordered set of RDF triples (subject, predicate, objects) and a named RDF graph is an RDF graph which is assigned a name in the form of a URI
- More specifically, GraphChain proposes:
  - a mechanism for linking named RDF graphs that are specified by meta-data with the use of Ontologies
  - a set of generic mechanisms for calculating the digest of named RDF graphs
  - a set of network mechanisms responsible for the distribution of named RDF graphs among distributed peers.

Sopek, M., Gradzki, P., Kosowski, W., Kuziski, D., Trójczak, R., & Trypuz, R. (2018, April). GraphChain: a distributed database with explicit semantics and chained RDF graphs. In Companion Proceedings of the The Web Conference 2018 (pp. 1171-1178).

Examples: Fluree



In the era of Web3 apps Fluree is an open source semantic graph database that ensures data integrity, and has features that enable secure data sharing. Fluree's ledger could run privately or as part of a federated network and it is used as an immutable time-ordered ledger, providing a data-centric architecture.

Fluree provides trust and traceability over every piece of data, combining transactions into immutable time-stamped blocks. Traceability of changes are backed with digital signatures. In addition to the blockchain capabilities offered, Fluree encapsulates semantic standards that link, leverage and connect information creating a rich semantic web of data. It uses RDF with standardized JSON-LD, alongside with SPARQL, and GraphQL query support. Regarding privacy and security permissions, Fluree embeds meta-data alongside RDF data at the source.

https://github.com/fluree/

Conclusions

Conclusions

- In this session we have revied various data sharing models in an attempt to understand the various mechanisms for sharing data. We are building the case that many techniques from the Data Management literature have been utilized in the design of DLTs/blockchains.
- We have then considered the design elements that let to a new definition of trust. Nakamoto proposed, in part, a proposal for reconsidering trust and redefined “the politics” in achieving agreements between untrusted parties.
- A comparison of Databases vs. Blockchains is discussed highlighting their distinctive properties and design considerations. Furthermore, we discussed the idea of P2P networks as a solution to scaling and distributing data.
- As we discussed in previous sessions, blockchains are a kind of a distributed ledger but not all distributed ledgers use a blockchain as their underlying data structure.
- Lastly, we conclude this session with a discussion on notable examples that show how blockchains can be blended with blockchains.

Glossary

Glossary

Append-only immutable ledger: a relatively simple database in which data elements can be added but can never be deleted or modified.
Merkle Tree: A Merkle tree is a hash tree in which successive pairs of hashes are themselves hashed until a single root hash is obtained. It is used for integrity purposes.
RDF Graph: is an unordered set of RDF triples and a named RDF graph is an RDF graph which is assigned a name in the form of a URI.
Sharding: is a database architecture pattern related to horizontal partitioning — the practice of separating one table's rows into multiple different tables, known as partitions.

Self-Assessment Exercises and Further Readings

Self-Assessment Exercises

- It seems that there is some confusion as to what a blockchain is and its dichotomy with a database. Although both are considered as ways to store information, they are not interchangeable due to their differences in design.
- Consider the main differences or unique features of each.

You are welcome to share your thoughts on the forums!

Further Reading

- Sánchez Galiano, S. (2019). *Comparison of underlying data structures for distributed ledgers* (Master's thesis, Uniandes).
- Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., & Saxena, P. (2016, October). A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 17-30).
- [Mohan, 2018] Mohan, C. "Blockchains and databases: A new era in distributed computing." In 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 1739-1740. IEEE, 2018.
- Maiyya, S., Zakhary, V., Amiri, M. J., Agrawal, D., & El Abbadi, A. (2019, June). Database and distributed computing foundations of blockchains. In Proceedings of the 2019 International Conference on Management of Data (pp. 2036-2041).

UNIVERSITY *of* NICOSIA

Questions?

Contact Us:

Twitter: @mscdigital
Instructor's Email: christodoulou.kl@unic.ac.cy

Course Support:
Mark Wigmans - wigmans.m@unic.ac.cy
Marios Touloupos - touloupos.m@unic.ac.cy

IT & live session support: dl.it@unic.ac.cy