UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Dependable Distributed Systems – 5880V/UE

Part 6. Blockchain and Consensus Protocols – 2021-10-12

Prof. Dr. Hans P. Reiser | WS 2021/22

UNIVERSITÄT PASSAU

Overview

Ledger

- Permanent recording of economic transactions
  - documenting contracts, payments, momements of assets, etc.
- Containing date, nature and monetory value of a transaction
- Traditional physical ledgers. . .

(CC0, source https://en.wikipedia.org/wiki/Ledger#/media/File:Ledger.png

- . . . and digital ledgers (digital file/collection of files/database)

Centralized and distributed ledgers

- Centralized shared ledger
  - Ledger maintained by a single entity
  - All parties trusted that entity
  - Feasible within one organisation
  - Controlling entity with malicious intent can cause serious damage
  - Controlling entity can shut down, preventing transaction processing

Centralized and distributed ledgers

- Centralized shared ledger
  - Ledger maintained by a single entity
  - All parties trusted that entity
  - Feasible within one organisation
  - Controlling entity with malicious intent can cause serious damage
  - Controlling entity can shut down, preventing transaction processing
- Distributed ledger
  - Ledger shared (replicated) across mulitple participants
  - Changes to ledger reflected everywhere
  - Mechanisms for ensuring consistency are required
  - Eliminates the need for a trusted central entity

Trusted Third Parties

Nick Szabo

- Inventor of smart contracts (1997)
- Inventor of “bit gold” (1998)

Nick Szabo, 2001: Trusted Third Parties are Security Holes

“The best ‘TTP’ of all is one that does not exist, but the necessity for which has been eliminated by the protocol design, or which has been automated and distributed amongst the parties to a protocol. ”

(`https://nakamotoinstitute.org/trusted-third-parties/`)

Layered view of a “blockchain”

- **Application layer:** many *potentially* interesting use cases
- **Transaction layer:** from simple “coins” to “smart contracts”
- **Distributed ledger technology (DLT):**
  - Replicated ledger: append-only history of all transactions, replicated across participants
  - Consensus: validation of transactions, ensures consistency
- **Cryptography:** Ensuring integrity of ledger, identity of participants, authenticity (and privacy) of transactions

Application layer: use cases

(source: `https://www.ibm.com/blockchain/industries` [2019-01-08])

Automotive

Drive innovative mobility services, supply chain traceability and more secure financial transactions.

Banking and financial services

Bring trust, simplicity and enhanced customer experience to financial services.

Government

Ensure data stewardship to protect citizen information, maintain trust and ensure the accuracy of public records.

Healthcare

Streamline clinical data across organizations and enable patients to control their medical data to increase the quality of care.

Insurance

Revolutionize the trust that powers insurance with an immutable foundation of transparency and shared purpose.

Media and entertainment

Build an ecosystem of trust around digital content usage — music, movies, television, advertising, loyalty points and more.

Overview

Blockchain as a state machine

Remember state machine:

- Has a state s
- Has inputs (operations) that atomically transform state: $s' = o(s)$
- May produce an output r for each operation (based on state/input)

Blockchain as a state machine

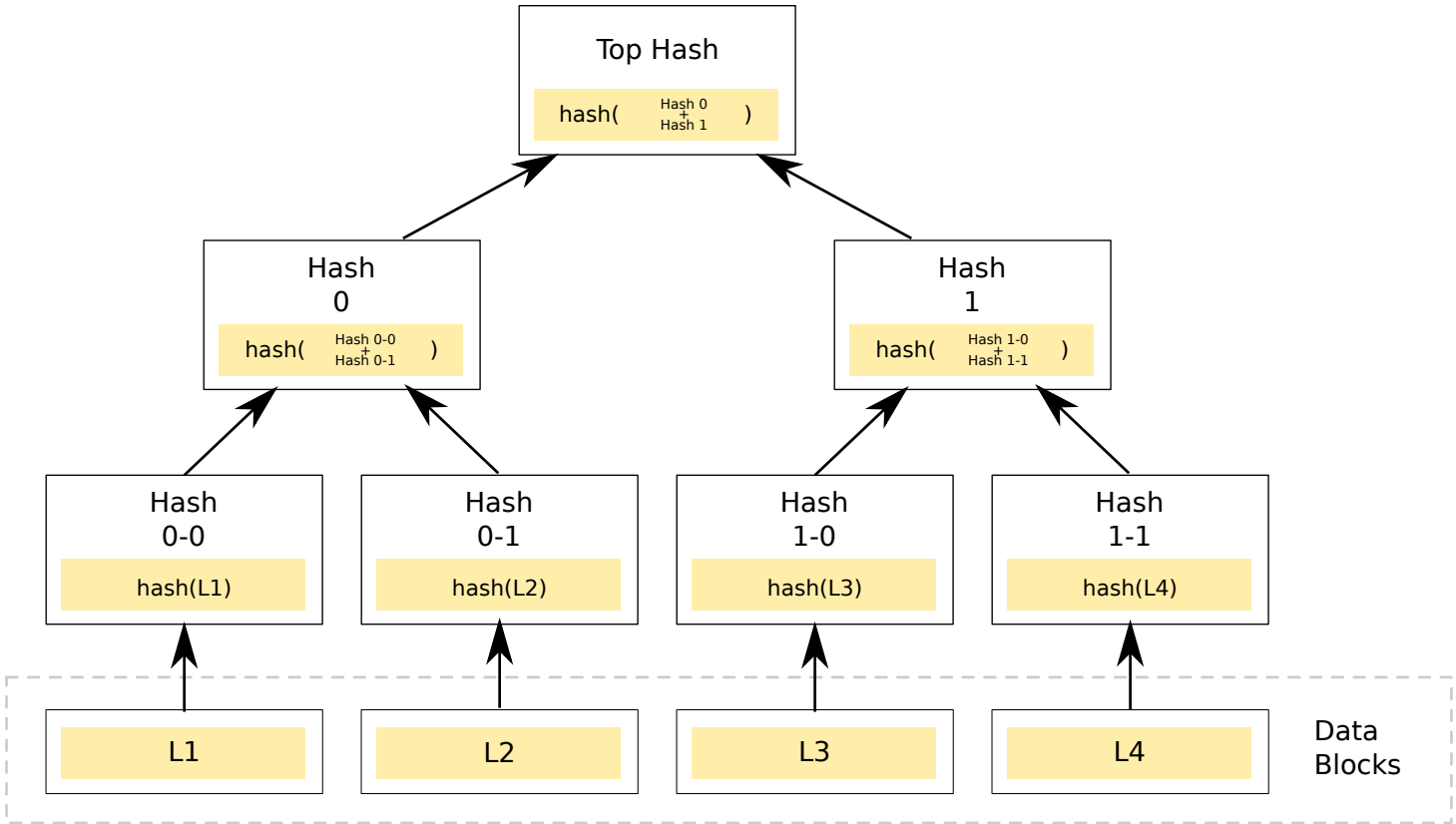
- State: log of transactions
- State machine operation: append a block of transactions tx^t to the log
- Hash chain: for each block t:
  $h_t := \text{Hash}([tx_1^t, tx_2^t, ...], h_{t-1}, t)$

- h_t depends on *all transactions* up to block t
- Wanted: efficient verification that tx_i^t is part of blockchain without having to know *all transactions*
  - Validity of block t can be verified based on knowledge of list of h_i up to t
  - Bitcoin: "block header" required for verification: 80 bytes (previous hash, transaction hash, mining nonce–see later), 1 block per 10 minutes
    $\rightarrow$ 4.2MB/year

Merkle tree

Definition (from Wikipedia): “. . . a hash tree or Merkle tree is a tree in which every leaf node is labelled with the hash of a data block and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes. ”

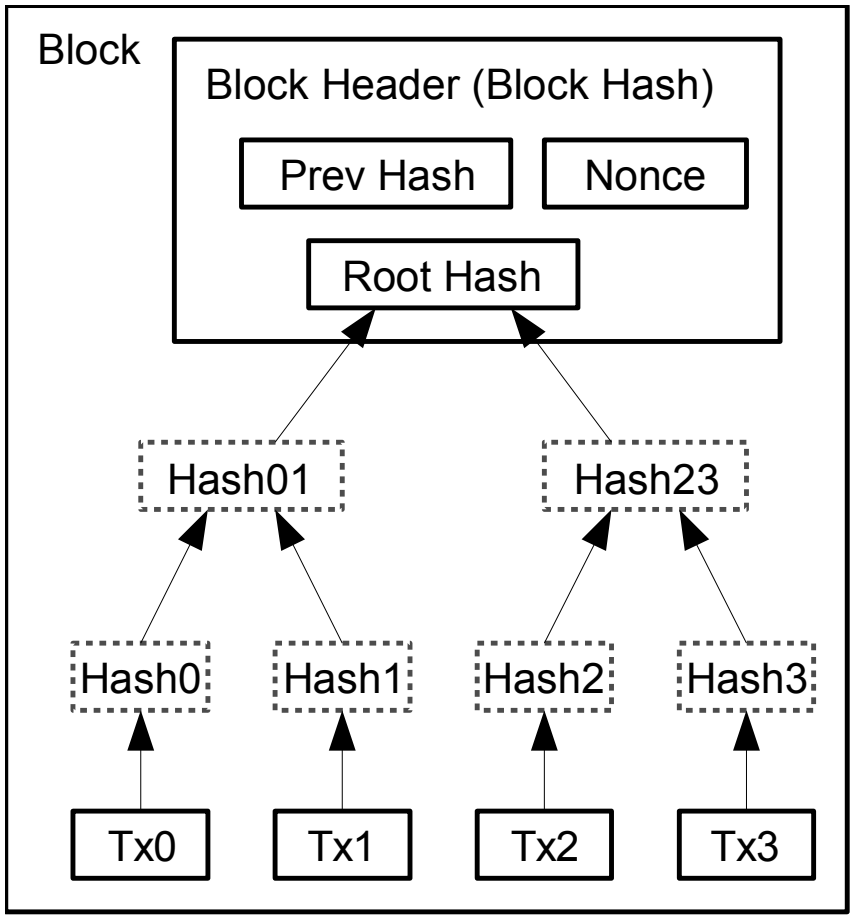
→ US patent, Ralph Merkle, 1979



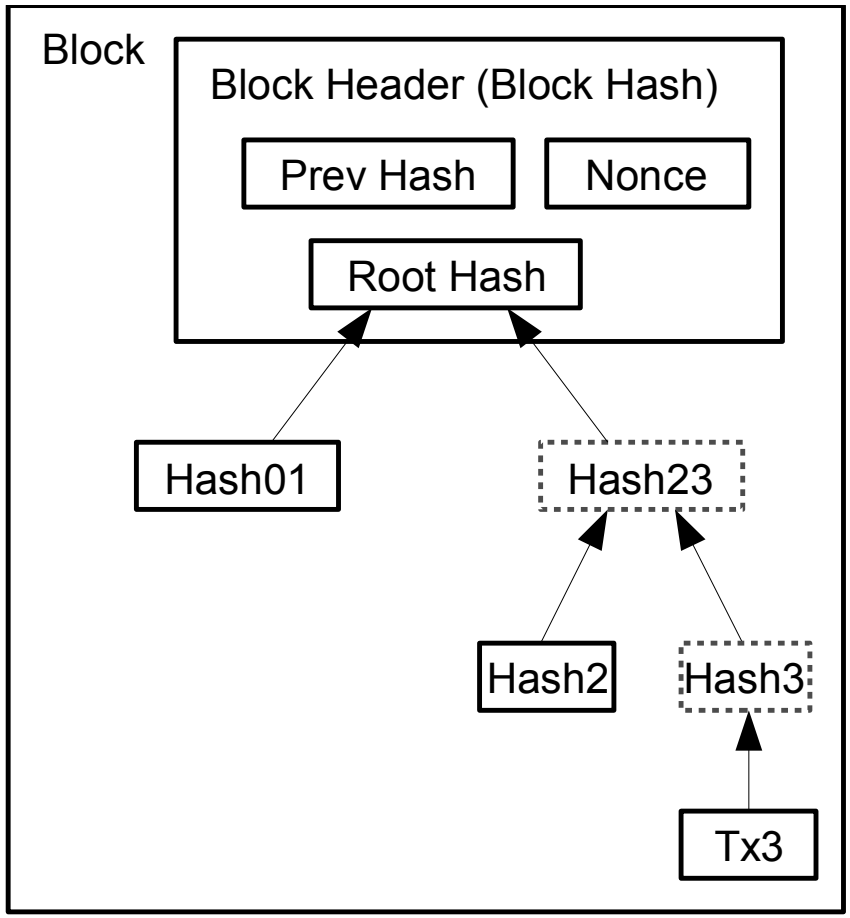
Blockchain and Merkle trees

- Potentially many transactions in one block
- Lightweight client wants to verify if one transaction is part of a block
- Simple construction of $[tx_1^t, tx_2^t, ...]$ is a hash of all transactions
  - Verification requires knowledge of all transactions in the block
- Merkle tree representation of $[tx_1^t, tx_2^t, ...]$:
  - Verification requires knowledge of only one transaction (leaf node) and hashes of „branches“ along the way from leaf to root

Blockchain and Merkle trees



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

(source: Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”)

Example: Bitcoin

- Bitcoin: unforgeable bitstring created by “mining”
- Owner of a bitcoin defined by digital signature key
- Bitcoin transaction: transfer of (fraction of) bitcoin from old owner to new owner
  - Transaction digitally signed by old owner
  - Only *valid* transaction will be included in blocks
- Validation of transactions: valid if
  - Signer has received bitcoin before, and
  - Signer has not yet spent the bitcoin

Example: Ethereum

- Ethereum transaction: Smart contracts written in Turing-complete instruction set (EVM: Ethereum Virtual Machine)
- Ether (ETH) as a crypto currency based on Ethereum
- Validation of transactions:
  - Each transaction has a “nonce” (counter for transactions of some account)
  - Transaction with different nonces: All preceding transactions are executed first; if not enough ETH in account, later transaction will be rejected
  - Transaction with same nonces: Multiple transaction with same nonce will be rejected

Consensus in blockchain protocols

- Need to make sure that all “replicas” append same blocks to chain
- Avoid inconsistent transaction
  - A sends bitcoin to B (on some of the ledger replicas)
  - A sends the same bitcoin to C (on some other ledger replicas)
- Main consensus approaches
  - Proof-of-work (PoW) consensus (“permissionless”)
  - BFT consensus (“permissioned”)

Overview

Bitcoin – Nakamoto's PoW consensus

- Nodes prepare blocks
  - List of *valid* transactions
- Nodes have to solve a "hard" problem (proof-of-work, "mining")
  - Usually, fastest node to solve problem wins
  - Solution is disseminated in a peer-to-peer fashion to all nodes
  - Nodes verify validity and append block to their chain
  - Miner gets reward for solving the problem (incentive. . .)

Bitcoin – Nakamoto's PoW consensus

- Nodes prepare blocks
  - List of *valid* transactions
- Nodes have to solve a "hard" problem (proof-of-work, "mining")
  - Usually, fastest node to solve problem wins
  - Solution is disseminated in a peer-to-peer fashion to all nodes
  - Nodes verify validity and append block to their chain
  - Miner gets reward for solving the problem (incentive. . .)
- Does this always guarantee consistency?

Bitcoin – Nakamoto's PoW consensus

- Several miners might solve the problem concurrently for conflicting transactions
- Conflicting blocks are disseminated ("forks")
- Solution: "longest chain wins"
  - If a node receives a longer chain, it replaces its local chain with the longer chain
  - With high probability, forks will not last forever
    (probability of fork with k blocks exponentially small in k)
  - Bitcoin transaction considered confirmed if 6 blocks have been appended

Hashcash proof-of-work function (1)

(based on Adam Back, “A partial hash collision based postage scheme”, 1997)

(note: PoW not an entirely new concept, see Cynthia Dwork and Moni Naor: “Pricing via Processing or Combatting Junk mail”, Crypto conference, 1992.)

- Relies on *pre-image resistance* of a secure hash function
  - given y and a hash function $H(x)$, it is difficult to find x such that $y = H(x)$
  - brute-force running time: $O(2^n)$, n: hash size
  - Bitcoin uses two iterations of SHA256 ($n = 256$)
- Hashcash challenge
  - Find x, such that $H(x)/2^{n-k} = 0$ (integer division)
  - In simple words: Find x, such that the upper k bit of $H(x)$ are 0.
  - k: “work factor”
    e.g., $k = 20$: on average 2^{20} operations required to find a suitable x

Hashcash proof-of-work function (2)

- Challenge: find x, such that $H(x)/2^{n-k} = 0 \ldots$
- Linking challenge to some purpose (block)?

Hashcash proof-of-work function (2)

- Challenge: find x, such that $H(x)/2^{n-k} = 0 \ldots$
- Linking challenge to some purpose (block)?
- Replace with "find c such that $H(s, c) \ldots$"
  - s: service string (purpose): block (header) to add to blockchain
  - c: some random counter

Hashcash proof-of-work function (3)

- Incentive for “mining”?

Hashcash proof-of-work function (3)

- Incentive for “mining”?

- Why should someone participate in mining?
- Bitcoin: first transaction in block: special transaction:
  new coin owned by block creator
- Alternative: transaction fees (miner collects fee that spenders may include in any Bitcoin transaction)
  (see also `https://bitcoinfees.info`)

Hashcash proof-of-work function (4)

- Hashcash difficulty coarse-grained (increment $k \rightarrow$ twice as difficult)
- More fine-grained control of difficulty?

Hashcash proof-of-work function (4)

- Hashcash difficulty coarse-grained (increment $k \rightarrow$ twice as difficult)
- More fine-grained control of difficulty?
  - Replace "find x such that $H(x)/2^{n-k} = 0$
  - by "find x such that $H(x) < 2^{n-k}$
    - equivalent if k is an integer
    - k can be a fractional number
  - Bitcoin: 2^{n-k} is called the "target"
    - recalculated every 2016 blocks (about two weeks)
    - basically represented as (24-bit base) $*$ $2^{(8\text{-bit exponent})}$

Bitcoin – Nakamoto's PoW consensus

- PoW discussion
  - Anyone can participate ("one CPU, one vote")
    - "permissionless" blockchains
  - Economic incentive to participate
  - Disadvantage: huge electricity consumption (Forbes 2018 estimation: Bitcoin world-wide similar to power consumption of Switzerland)
  - Relatively slow transactions (Bitcoin: average 1 block/10 minutes)

Goal of decentralization in reality

- Assumption in PoW blockchains: many participants, thus very difficult for an adversary to control more than 1/2 of all nodes(*).
- `https://arewedecentralizedyet.com`
  (unfortunately no longer available in its usual form, but see archive.org)
- `http://web.archive.org/web/20181126060148/https://arewedecentralizedyet.com/`

(*) Note on “more than 1/2 of all nodes’

Common believe: Adversary can do harm if he controls >50% of the mining power

- True

Common believe: Adversary controlling less than 50% of mining power can do no harm

- Wrong
- see Eyal and Sirer: *Majority is not Enough: Bitcoin Mining is Vulnerable*
  - In Bitcoin, “selfish mining” possible if controlling 25% of mining power
  - Can be improved to $1/3$ by protocol enhancements

Intel's Sawtooth ledger

"Proof of elapsed timed" (PoET) consensus

- Work replaced with waiting for a random amount of time
- Requires secure random number generation and attested proof of elapsed time
- PoET program executed in trusted execution environment (Intel SGX)

Overview

BFT consensus in blockchains

- BFT consensus suitable for ordering transactions in a replicated system (remember previous lectures)
- Requirement: Known set of participants ($\Rightarrow$ consortium consensus)
  - Requires group membership management
- Scalability challenge: Typically $O(n^2)$ communication complexity for n participants
  - Suitable for 10s or maybe 100s of nodes
  - Not suitable for 1000s of nodes

Some existing scalable BFT protocols

	ByzCoin	FastBFT	Stellar	HoneyBadgerBFT	Algorand	Gosig
Scalability (evaluated with)	1004	199	currently running ca 100	104	**up to 500k**	up to 10k
Throughput (transactions/s)	700 (n=1004)	370 (n=199)	1000 (n ca. 100)	1200 (n=104)	<1000	4000 (n=140)
Latency	30s	**< 1s (1 Gbps LAN)**	few seconds	100s	1 minute	<1 minute
Synchrony	weakly synchronous	weakly synchronous	asynchronous, but progress depends on synchrony	**asynchronous**	weakly synchronous	asynchronous, but provable liveness only under weak synchrony
Consensus determinism	deterministic	deterministic	deterministic	probabilistic	probabilistic	probabilistic
Approaches for scaling consensus	comm. tree + collective signatures	hardware-based TEE + secret sharing, tree topology	federal Byzantine agreement with hierarchical structure	novel ACS reduction with threshold crypto, efficient RBC with erasure codes	committee (cryptographic sortition) + gossip	multi-signatures + gossip

BFT consensus in blockchains

More on that in next presentation