

7CCSMDLC: Distributed Ledgers & Cryptocurrencies

Lecture 3: Protocols & Consensus



Peter McBurney

Professor of Computer Science
Department of Informatics
King's College London

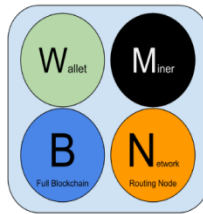
Email: peter.mcburney@kcl.ac.uk
Bush House Central Block North – Office 7.15

2020



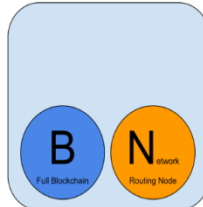
Outline

- Byzantine Fault Tolerance
- CAP Theorem
- Consensus Protocols



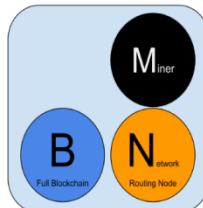
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



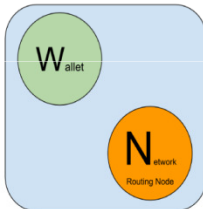
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



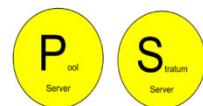
Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



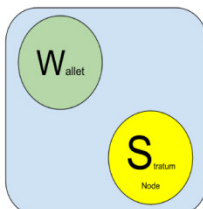
Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.






Byzantine Faults and CAP Theorem



Byzantine Generals

- Generals surround a city
 - Each General can Attack or Retreat
 - Success only if all Attack or all Retreat
- They vote and agree to abide by majority decision
- They are polled and each sends his preference to others
 - 9 Generals: 4 say Attack, 4 say Retreat, 1 is a Traitor
- What happens if the Traitor sends different message to different generals?
 - He sends Attack to the 4 who favour Attack
 - He sends Retreat to the 4 who favour Retreat
 - Each group thinks they have a majority
 - Some attack & some retreat
 - Outcome: Failure

|  |  |  |
|--|--|--|
| Agree on a Strategy | Objective | Agree on Valid Transactions |
| Separated Camps | Spacial Distribution | Distributed Nodes in the Network |
| Loyal Troop and Loyal Generals | The Good Ones | Truthful Nodes |
| Traitors | The Bad Ones | Evil Nodes |
| Corrupt a Message | The Attack | Add an Invalid Transaction to the Blockchain |
| How to Know which Message is True | The Problem | How to know which Transaction is Valid |
| Don't Have | A Solution | Proof of Work |
| Don't Have | Consensus | Blockchain with More Combined Difficulty |



Byzantine faults

- Byzantine faults – faults which appear different to different observers
 - A general sends different messages to different colleagues
 - Attack / Retreat
- A malicious node may send different blocks to different other nodes
- Byzantine Fault Tolerance
 - A system designed to withstand such attacks
- Bitcoin blockchain is Byzantine Fault Tolerant
 - Because it makes it costly to send false messages
 - Nodes have to do Work to have a block successfully included in chain



Databases on ACID

Desirable properties of a database

- Atomicity
 - Each transaction is all-or-nothing.
 - If part of a transaction fails, it all fails
- Consistency
 - Any data written must be valid according to all rules & constraints
 - Reads of same data must be the same
- Isolation
 - Concurrent execution of transactions must lead to same outcome state as if the transactions were done sequentially
- Durability
 - Once a transaction is committed, it remains so.



Challenges for large databases

- Relational databases use SQL (Structured Query Language)
- Alternatives to relational databases have come to be known as NoSQL databases
 - Non-tabular data
 - Flexible schema, disparate data types & formats
 - Distributed and replicated databases
 - Especially for large data holdings
- Examples:
 - Google, Amazon, Facebook, Twitter
- Do not generally support ACID properties
 - Usually Consistency is relaxed
 - Inconsistencies resolved when data is read.

Example: Apache Cassandra

- Free, open-source NoSQL database management system
 - Originally developed by Facebook, then adopted by the Apache S/W Foundation
- For large data holdings held across multiple servers
- For instance: 2 types of Read transaction:
 - **Single Read:** Search is only local
 - **Quorum Read:** Search is wider and value returned is that endorsed by the majority of nodes after a quorum vote
 - These may return inconsistent results.



cassandra



CAP Theorem

- We have a design challenge for distributed databases, expressed by the CAP Theorem:
- Only 2 of the following 3 properties are possible to achieve simultaneously:
 - Consistency of data
 - Availability of data
 - Partition-tolerance
- Partition-tolerance refers to how the data is stored or distributed.
- History
 - Conjectured by Eric Brewer in 2000.
 - Limited form proved by Seth Gilbert & Nancy Lynch.



Design Implications for Distributed Systems

- Faced with the CAP Theorem, designers cannot forgo partition-tolerance, so much choose between Consistency or Availability.
- This is a trade-off and choice will depend on the requirements of the domain,
 - If speed of response is not an issue, then choose C.
 - If response needs to be immediate, then choose A.
- In Bitcoin blockchain, the partition arises due to the P2P network.
- Bitcoin blockchain opts for Availability
 - This may be considered a weakness for a financial system.
 - Although not Consistent, Bitcoin blockchain is **eventually consistent**.



Bitcoin Blockchain

- Has no master or central node to enforce Consistency.
- So needs a consensus algorithm, for nodes to vote on the true state of the database (the blockchain).
- Bitcoin blockchain is open, so cannot use simple majority voting
 - No way to ensure that a majority is present
 - No node has overall view of all other nodes
 - A majority could be manipulated (Sybil attack)
- Bitcoin blockchain is eventually consistent
 - The older a block is, the more work is required to change it
 - The longest internal fork was 24 blocks in length
 - March 2013, caused by a s/w bug
 - No transaction more than 5 hours old has been reversed.



Forks in Bitcoin blockchain

There are two types of fork:

- **Regular operations (or internal) fork:** The temporary existence of competing chains as miners process competing blocks
 - Resolved through consensus protocol.
- **Software fork:** In open-source software projects, a fork is a new software project trajectory that starts from an earlier project.
 - Typically, changes to open source software may require a fork.



Software Forks

Again, there are 2 types of these forks:

- Soft fork: The new version of the software is backwards-compatible
 - In other words, nodes do not need to adopt the new version of the software. In a blockchain, nodes can still recognize new blocks even if they are running the old software version.
 - The network will typically have a mix of nodes running the old & the new software.
- Hard fork: The new version is not backwards compatible.
 - Every nodes needs to adopt the new version of the software to recognize new blocks.

You run a vegetarian-only restaurant and if you change it to be an vegan-only restaurant then you are soft forking it, if you change it to include meat you are hard forking it.
- Andreas Antonopoulos

Bitcoin hard forks involving currency splits

- Bitcoin Cash:

- Forked at Block 478,558
- 1 August 2017



- Bitcoin Gold:

- Forked at Block 491,407
- 24 October 2017



- Bitcoin Diamond:

- Forked at Block 495,866.
- 12 December 2017.





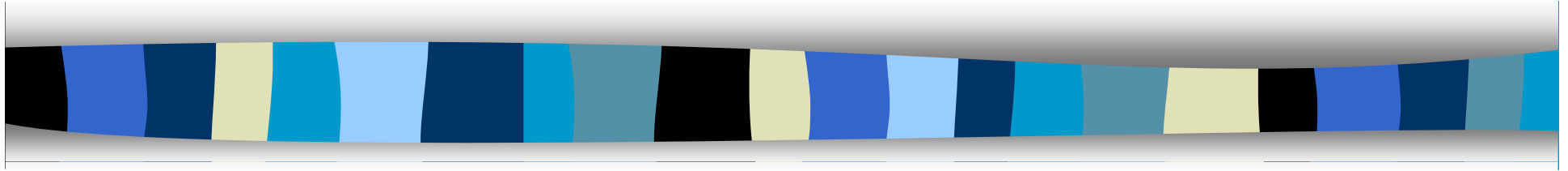
Altcoins

- Alternative cryptocurrencies
 - Often use code from Bitcoin modified in certain ways
- Some altcoins and intended features:
 - Litecoin (2011): Increased speed of transaction (4 times faster than BTC)
 - Dash (2014): Increased speed of transactions
 - Monero (2014): Enhanced privacy
 - Zcash (Zerocoin) (2016): Enhanced privacy
 - Bitcoin Cash (2017): Increased blocksize
 - Bitcoin Gold (2017): ASIC-resistance.



Ethereum — The DAO Fork Example

- The DAO
 - Decentralized Autonomous Organization
 - Self-running VC fund running over Ethereum
 - Raised \$150 million in 1 month (May 2016) from 11,000 investors
 - Intended that token holders would vote on investment proposals
- June 2016: Vulnerability exploited
 - Not a bug, but exploitation of poor code
 - \$50 million siphoned off
- Ethereum nodes voted to hard-fork to restore lost funds
 - 20 July 2016 at block 1,920,000
 - 2 branches:
 - Ethereum (prior to the exploitation)
 - Ethereum Classic (exploitation continues).



Consensus Protocols



Consensus Protocols

- Proof of Work (PoW)
- Proof of Stake (PoS)
- Proof of Authority (PoA)
- Other turn-taking protocols



Proof of Work (PoW)

- How it works
- Why was it proposed?
- Problem of power usage
- Unintended consequences
 - Power usage
 - ASICs and dedicated clusters
 - Large miners & mining pools

[BLOG »](#)[REVIEWS »](#)[BITCOIN SUSTAINABILITY »](#)[ETHEREUM ENERGY CONSUMPTION »](#)[Home](#) / [Bitcoin Energy Consumption Index](#)

Bitcoin Energy Consumption Index

NEW: [Bitcoin Electronic Waste Monitor](#)

Bitcoin Energy Consumption Index Chart

Pinch the chart to zoom in



BitcoinEnergyConsumption.com

[Download data.](#)

Source: *digiconomist.net 2020-01-26*

Key Network Statistics

| Description | Value |
|--|-----------------|
| Bitcoin's current estimated annual electricity consumption* (TWh) | 73.17 |
| Bitcoin's current minimum annual electricity consumption** (TWh) | 52.44 |
| Annualized global mining revenues | \$5,586,384,139 |
| Annualized estimated global mining costs | \$3,658,297,714 |
| Current cost percentage | 65.49% |
| Country closest to Bitcoin in terms of electricity consumption | Austria |
| Estimated electricity used over the previous day (KWh) | 200,454,669 |
| Implied Watts per GH/s | 0.075 |
| Total Network Hashrate in PH/s (1,000,000 GH/s) | 111,057 |
| Energy footprint per transaction (KWh) | 637 |
| Number of U.S. households that could be powered by Bitcoin | 6,774,625 |
| Number of U.S. households powered for 1 day by the electricity consumed for a single transaction | 21.53 |
| Bitcoin's electricity consumption as a percentage of the world's electricity consumption | 0.33% |
| Annual carbon footprint (kt of CO2) | 34,754 |
| Carbon footprint per transaction (kg of CO2) | 302.62 |

*The assumptions underlying this energy consumption estimate can be found [here](#). Criticism and potential validation of the estimate is discussed [here](#).

**The minimum is calculated from the total network hashrate, assuming the only machine used in the network is Bitmain's Antminer S9 (drawing 1,500 watts each). On February 13, 2019, the minimum benchmark was changed to Bitmain's Antminer S15 (with a rolling average of 180 days), followed by Bitmain's Antminer S17e per November 7, 2019.

Source: *digiconomist.net 2020-01-26*



Proof of Stake (PoS)

- Nodes validate blocks in proportion to their stake in the system
 - ie, how much of the internal currency the node has
 - Weighted turn-taking (weighted according to stake)
- Then other nodes have to approve
 - Could be a majority of all other nodes (eg, Tendermint)
 - Or a random selection of nodes
- If nodes behave badly, they forfeit their stake
 - No coin creation (mining), but internal currency still needed
- Examples: Casper, Tendermint
 - Ethereum will transition to PoS in Eth 2.0 under the Serenity update
 - Adoption delayed (still being tested): now estimated for 2021
 - How to manage the transition to PoS?



Proof of Stake (2)

- How do design a PoS system so that nodes are economically incentivized to behave well?
 - Cryptoeconomics
 - How to prevent Sybil attacks
- Need to pay validators only after a block is confirmed
- Vitalik Buterin's Metaphor:
 - 100 people sitting around a table signing competing documents
 - Each person only gets paid if they sign the documents that a majority of other people sign
 - So payment can only be at the end.



Proof of Stake: Possible flaws

- Initial distribution problem
 - How to incent users who have initial coins to transfer coin ownership, since their stake is valuable
- Validators signing competing blocks
 - This is rational if there is a fork
 - aka Nothing-at-stake problem
- Cartel Problem
 - How to prevent cartels
- Finality Problem
 - How to stop blocks being removed later
 - For PoW systems, the hashing power required to remove a very old block is immense. Is this true in PoS systems?
- If validators forfeit the coin, so what?



Proof of Authority

- Nodes with authority to process blocks are chosen to do so
 - Either randomly or according to some pre-determined rules
 - Nodes may take it in turns to mine blocks
 - Nodes may act as miners for other certain other nodes (eg, as Notaries)
- For example:
 - Companies in a consortium together each take turns to process blocks
- Requires trust in the nodes, and hence only suitable for permissioned ledgers
 - Permissioned ledgers normally have a consortium agreement between the companies involved, which governs investments and penalties.
- R3's Corda platform
 - Uses Notaries to witness & process transactions
 - But this is not strictly a blockchain.



Other turn-taking protocols

- Protocols could simply decide which nodes should process blocks on some basis, eg
 - Random assignment
 - Time since they last had a turn
- Proof of Elapsed Time (PoET)
 - Developed by Intel as part of their Sawtooth Lake services running on the Hyperledger platform
 - Intended to be run in a secure hardware environment for permissioned ledgers
 - So perhaps less prone to attack by malicious nodes.



Consensus Attacks

■ Sybil attacks

- A malicious node creates proxies that appear to be run by different owners
- In a system using PoW, these would only succeed if the group has sufficient combined hashing power
- They may succeed under other protocols.

■ 51% Attacks

- Where a miner or a cartel of miners are able to attack the network because of their combined hashing power
- A majority is not necessarily needed – simulations show that these attacks may succeed with only 1/3 of hashing power (depending on distribution of hash power and structure of network)
- Chinese miners currently could engineer such attacks on Bitcoin.
 - Would it be in their economic interest to do so?
 - Would it be in their strategic political interest to do so?



Forms of 51% attack

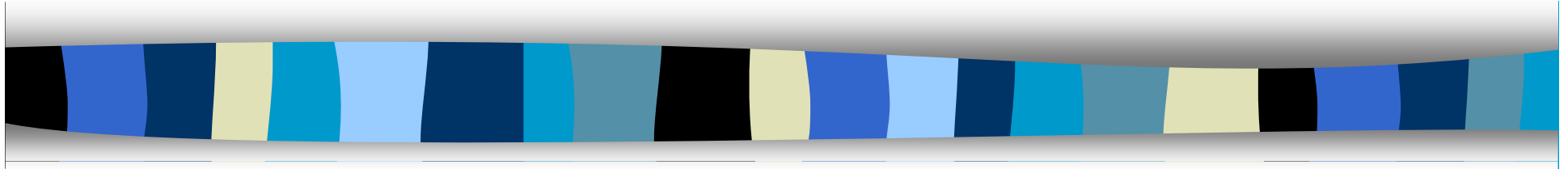
- Create deliberate forks
- Execute DoS (Denial of Service) attacks against specific Transactions or Addresses
 - Just refuse to process these transactions or transactions to/from these addresses
- Double-spend attacks (see next slide).



Double-spend attacks

- How it works:
 - A and C are attackers, B is the mark
 - A and B have an agreement to exchange bitcoin for some offline goods or fiat money
 - TX1: Attacker A sends bitcoin to B
 - B sees TX1 is confirmed & pays A the offline goods or fiat money
 - TX2: A spends same bitcoin with C (aligned with A)
 - A and colleagues put hash power behind TX2
 - So the blockchain eventually confirms TX2 and not TX1.
- Attackers can only double-spend their own transactions
 - since they have to sign transactions.
- What can B do to avoid loss?
 - Do not accept a single confirmation of TX1 (ie, wait for more blocks)
 - Eg, wait 24 hours (or 144 blocks)
 - Use an escrow (multi-sig) account for the goods/fiat currency.

Thank you!



peter.mcburney@kcl.ac.uk



Exercises

1. Create a table of the advantages and disadvantages of the major consensus protocols
 1. PoW
 2. PoS
 3. PoA
 4. Turn-taking protocols
2. List all the possible attacks that have been identified for these four protocols.
3. What was the software bug which caused a long internal fork in Bitcoin in March 2013? How was it resolved?
4. Explore the vulnerability of the Polkadot / Parity ICO. What was the problem? What are the possible solutions?