TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Programming Distributed Systems

13 Blockchains

Christian Weilbach & Annette Bieniusa

AG Softech
FB Informatik
TU Kaiserslautern

Summer Term 2019

Introduction

Blockchain?


YOUR DATABASE MAY
BE FASTER AND CHEAPER
BUT MINE'S A BLOCKCHAIN
imgflip.com
memegenerator.net

What is a blockchain?

- It is a chain of blocks.
- Actually just the (replicated) transaction log
- What is the point actually???

The Bitcoin blockchain: the world's worst database[1]

Would you use a database with these features?

- Uses approximately the same amount of electricity as could power an average American household for a day **per transaction**
- Supports 3 transactions / second across a global network with millions of CPUs/purpose-built ASICs
- Takes over 10 minutes to "commit" a transaction
- Doesn't acknowledge accepted writes [..]
- Can only be used as a transaction ledger denominated in a single currency, or to store/timestamp a maximum of 80 bytes per transaction

But it's decentralized! (is it?)

[1] Source: https://tonyarcieri.com/on-the-dangers-of-a-blockchain-monoculture

Political motivation

Satoshi Nakamoto

- Mysterious inventor of Bitcoin
- This is not Satoshi Nakamoto

On 31 October 2008 on some crypto mailing list

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing

Anarchocapitalism

- Strong form of free market ideology
- Directed against (central) banks and states
- Market and money are sacrosanct (following Friedrich Hayek, Ayn Rand)
- Affiliated to libertarian ideology prominent in Silicon Valley
- *But*: can also be read as reaction to monopolisation and privatisation

Platform economy

Examples: Facebook, Uber, Google, Amazon, AirBnB, ...

Strategy:

1) Get users on your platform and grow as fast as possible with vencture capital (VC) money
2) Encourage network effects through open strategy and free products
3) Privatize platform and own data $\Rightarrow$ **profit**

post-68 Internet vision

- Platform economy focuses on **individualism of consumer**
- Turned into vague, "Orwellian" startup terminology: disruption, democratization, participation, openness, progress, community
- **But**: today it is threatening surveillance capitalism
  - Amazon Teams Up With Law Enforcement to Deploy Dangerous New Face Recognition Technology
  - Google Is Quietly Providing AI Technology for Drone Strike Targeting Project
  - We work for Google. Our employer shouldn't be in the business of war

What now?

- P2P systems & free/open source movement
- Cypherpunks: cryptography, e.g. PGP
- Political ideologies against centralization:
  - left anti-state, right anti-state
- Examples: BitTorrent, Bitcoin, Wikis, git
- Idea: *Software emancipates from hardware*
- *Problem*: no economic system
- Answer: ICO-mania as response to VC funding??

ICO (initial coin offering) = a quantity of cryptocurrency is sold in the form of "tokens/coins" to speculators and investors, in exchange for legal tender or other cryptocurrencies

Bitcoin

- Political argument as code
- *Game theory* as programmable economics
- Technical design *not from angle* of DB architect
- Distributed system as answer to **centralization of power**

What is a blockchain technically?

Blockchain as DB

- $\approx$ *Strongly-consistent* database:

  $\Rightarrow$ **total order** of events (like atomic broadcast)

  $\Rightarrow$ scalability $\leq$ any strongly consistent DB

- Problem is **permissionless environment**: Adversarial

- Needs to be decentral/neutral w.r.t. to peers running the network

- Cannot be privatized

Byzantine Fault Tolerance

- Paxos, Raft, etc. are supposed to run in trusted environment
- Adversarial environment: fake messages, drop messages, delay messages
- Threshold of honest peers (generals), e.g. $> 2/3$

Bitcoin

Design objectives

- **Economics**: game theoretic equilibrium
- **State**: no censorship or seizing of money
- **Money**: no inflation through central banks
- **Politics**: decentralized network

Nakamoto consensus[1]

- Byzantine fault-tolerance (fake message, dropped messages, delayed messages)
- Technology existed 10-15 years before Bitcoin
- *Recombination* is novel
- Interesting usage of cryptography

HashCash (1997)

- Problem: spam flooding protection
- Idea: To post on message board you have to do a **tiny** amount of crypto work, *but* spammers have to pay proportional price
- Use property of cryptographic hash functions like SHA-256

On cryptographic hash functions

Hash function H takes arbitrary string as input and produces fixed-size output (here: 256 bit)

Properties:

1) Efficient to compute
2) **Practically** collision-free
3) Given $H(x)$, it is infeasible to find x
4) Puzzle-friendly: For every possible output value y, it is infeasible to find x such that $H(k \cdot x) = y$ if k is chosen from a distribution where every value is chosen with negligible probability

($\rightarrow$ No strategy is much better than trying random values of x)

How can cryptographic hashing be useful

- If we know $H(x) == H(y)$, then it is safe to assume that $x == y$
- Use hash as a message digest (much smaller than message)
- Can commit to a message, but only reveal it later
- Set up "search puzzle": Given k and a target set Y, find a solution x such that $H(k \cdot x) \in Y$

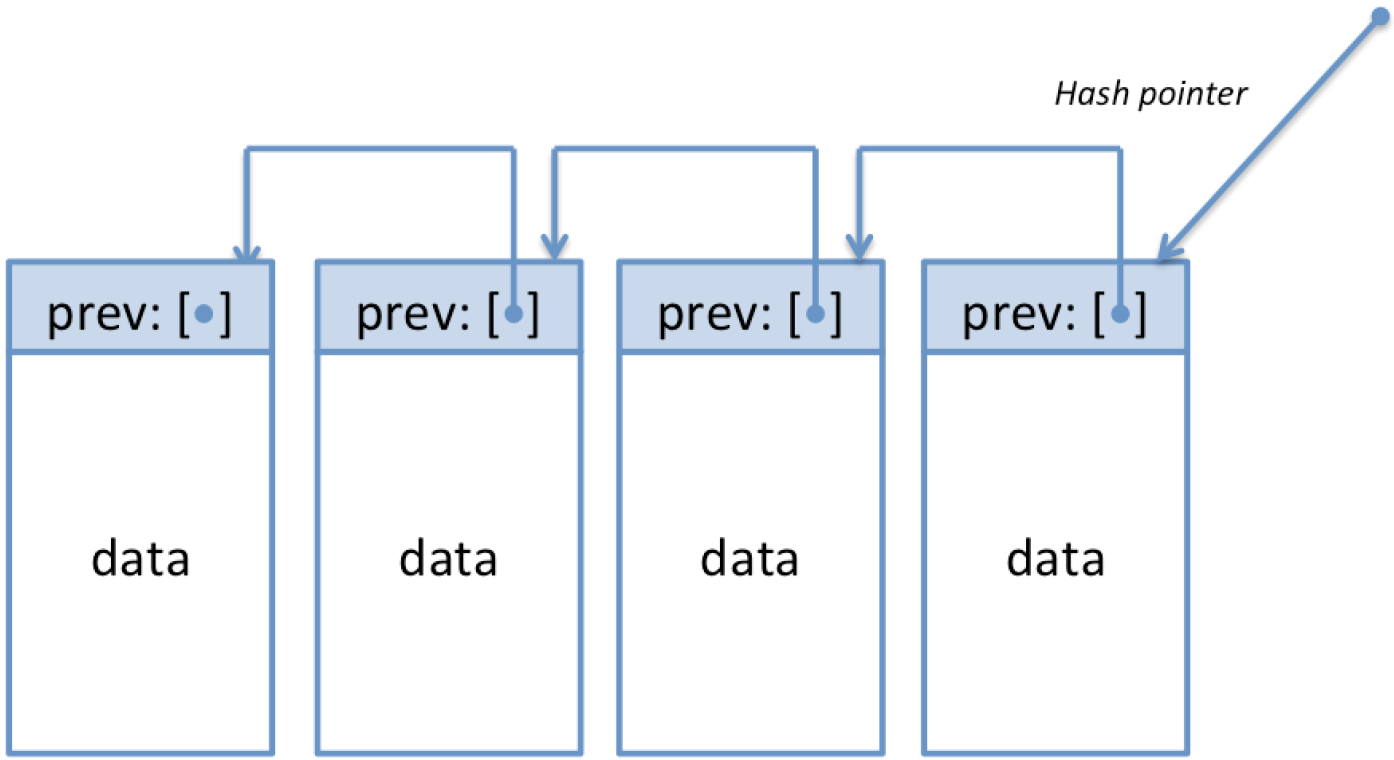
On hash pointers

A hash pointer is a pointer to some information plus the cryptographic hash of the information.

Purpose:

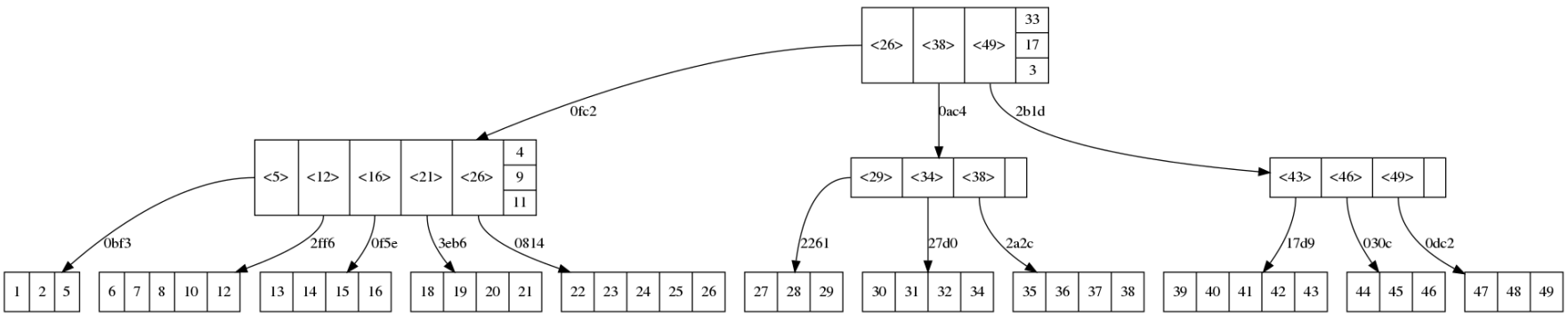
- Access to the information
- Verification that information hasn't changed
- Build temper-evident data structures!

Blockchain: A temper-evident log



What happens if somebody tries to modify the data in one block?

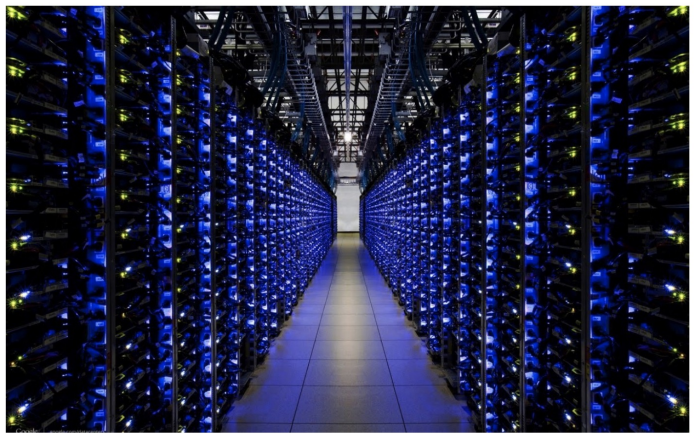
Merkle Trees



Mining a block: Proof of Work

- Difficulty target: Hash must be smaller than this value (leading zero bits, defines Y)
- $H(b \cdot x) \in Y$, b block bits, x chosen **nonce**
- Quadrillions of hash operations per second
- Today: mining pools with ASIC hardware

The Dragonmint 16T miner.



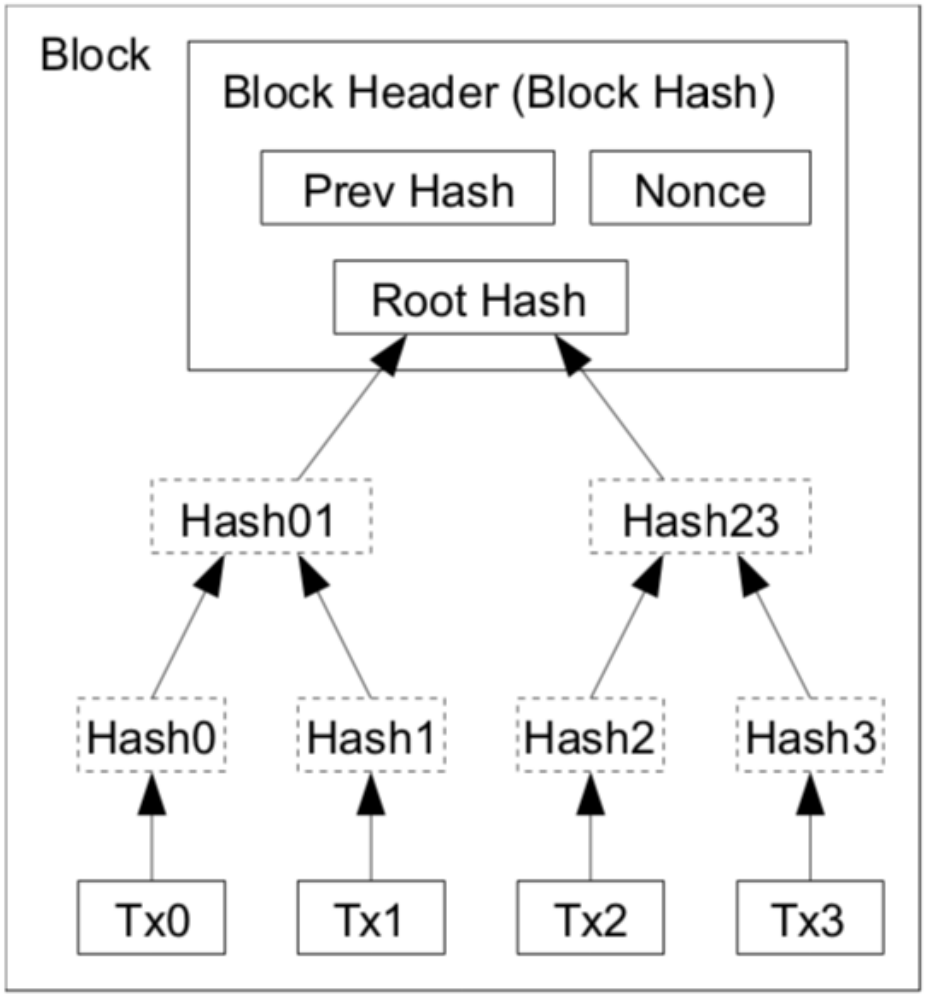
A bitcoin mining rig in China

Source: https://www.buybitcoinworldwide.com/mining/hardware/

Bitcoin's block chain

- Started with "genesis" block by Satoshi Nakamoto on Jan 3, 2009
- Blocks can join and leave: $\Rightarrow$ replay operations to obtain actual state
- Distributed ledger of 235 gigabytes (Jan 2019)
- Most difficult ($\approx$ longest) chain wins
- **Race between miners**
- **Gossiping P2P** network of Bitcoin nodes (aka Bitcoin Core)
- Milliseconds matter!

Block structure



Transactions Hashed in a Merkle Tree

Consensus specification

- Choice between “Immutability” or “Code as law” . . .
- Rules: Implementation is specification (including bugs)
- C++ codebase + dependencies (Ughh)

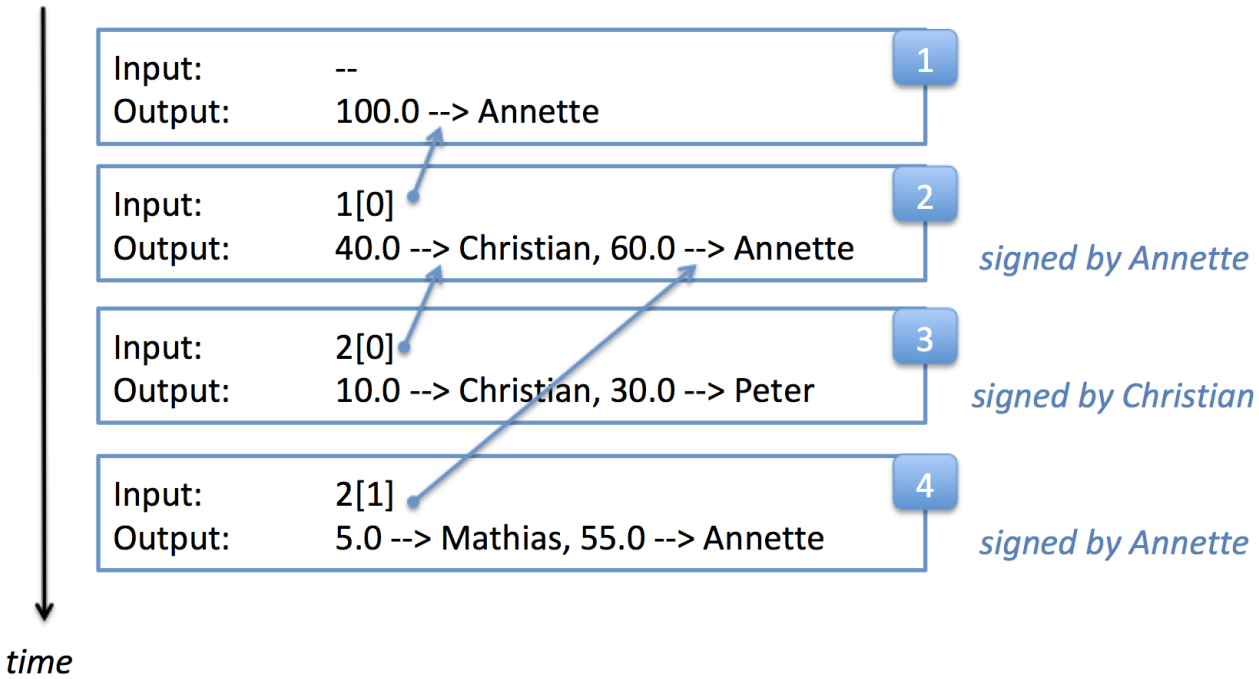
Trust model

- Checked *before* a block is accepted
- 30-40 rules for transaction
- Importantly: 0 **sum changes, positive balance**
- 30-40 rules for each block
- Rules are specified in C++

Pseudo-algorithm

1) Take chain with most work behind it
2) Take received transactions and build a block
3) Try to brute-force a $H(b \cdot x) \in Y$ with current difficulty level
4) Either find a block first and propagate it as quickly as possible or receive a new block: Repeat with 1.

Transaction-based ledger



- Authorize txn by signing with owner's key
- Simplification here: only one txn/block
- Validation check with hash pointers

Miners against users?

- Idea: incur cost vs. expected reward
- Fixed amount of block reward
- *Assumption*: at least 50% of nodes are honest
- Corresponds to voting/betting on winning chain
- **Cheating**: create **invalid blocks** or **delay** network
- *But*: does not pay to cheat

How high is the probability of a fork of length N?

p^N, where p is the probability that both partitions mine a new block in each step at approximately the same time.

$\Rightarrow$ astronomically small for larger N.

(Imaginary) Example of fork

- Example: Germany blue $\leftrightarrow$ Japan red
- Partition in network happens
- Next block either is created in blue or red or in blue
- Orphan the block
- Red wins: Take transactions from orphaned block, replay blue txs
- Other chain never happened

Convergence

- **Probabilistic convergence**
- A fork of size 1 happens daily
- A fork of size 2 weekly
- ...
- A fork of size 6 practically never happens...

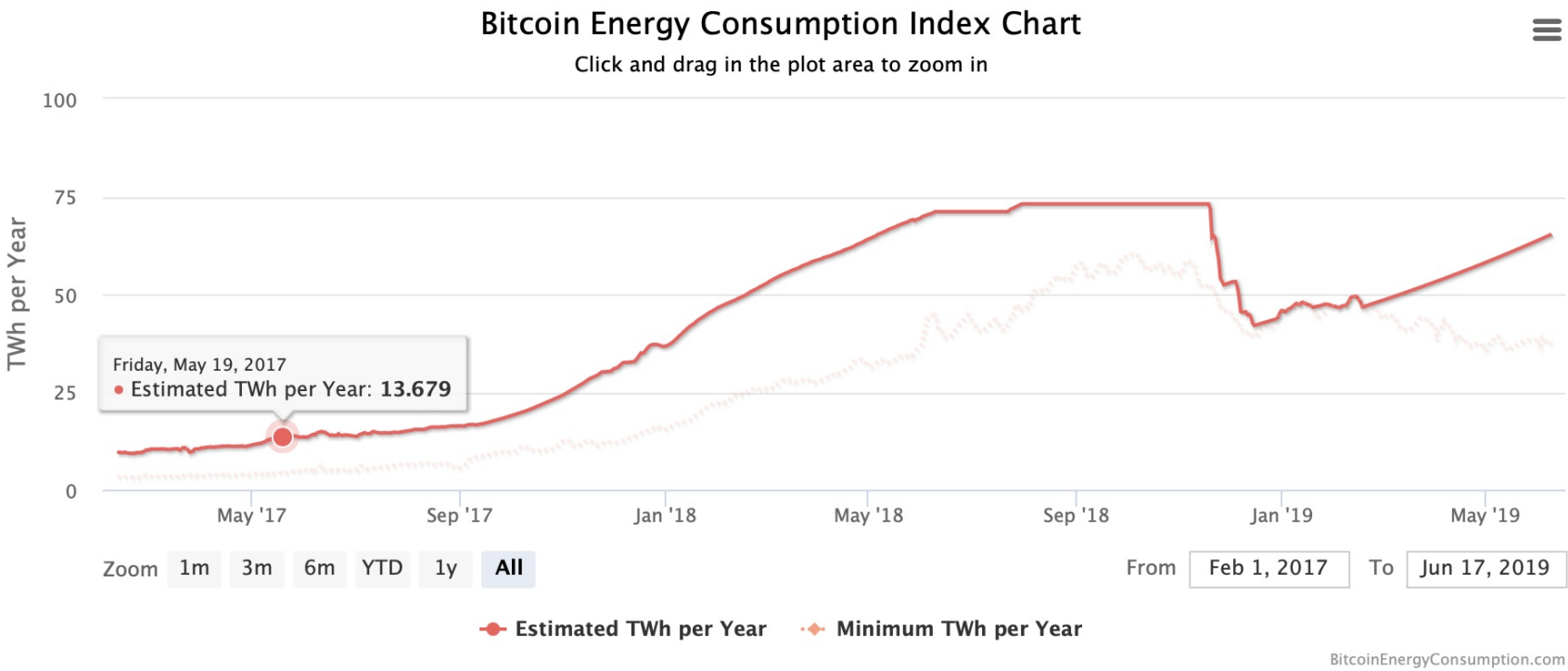
Bitcoin bugs

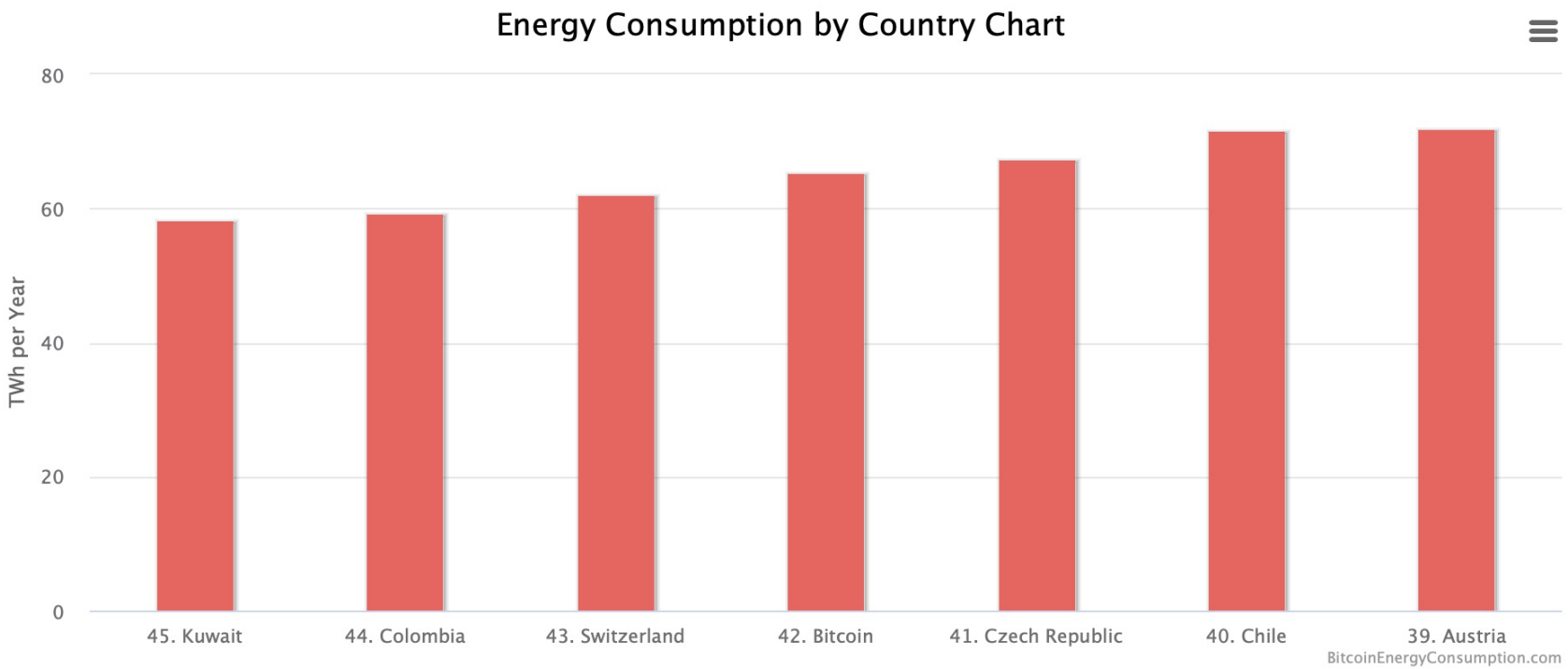
- April 2013: 7 blocks fork
- Cause: switch to LevelDB in implementation
- Block with 1200 transactions
- $\Rightarrow$ crashed BerkelyDB (max. 1024 txs) (bug)

```
A block that had a larger number of total transaction inputs than
    previously seen was mined and broadcasted. Bitcoin 0.8 nodes
    were able to handle this, but some pre-0.8 Bitcoin nodes
    rejected it, causing an unexpected fork of the blockchain.
```

Source: https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki

Problems


Bitcoin Energy Consumption Index Chart
Click and drag in the plot area to zoom in
100
75
50
25
0
TWh per Year
Friday, May 19, 2017
Estimated TWh per Year: 13.679
May '17
Sep '17
Jan '18
May '18
Sep '18
Jan '19
May '19
Zoom
1m
3m
6m
YTD
1y
All
From
Feb 1, 2017
To
Jun 17, 2019
Estimated TWh per Year
Minimum TWh per Year
BitcoinEnergyConsumption.com



- Currently consumes electricity, powered by coal mines (!!!)
- High latency: $10 - 60$ minutes (6 blocks confirmation)
- Low throughput (< 10 tx/sec)
- Actually eventually consistent (always reversible)

Ethereum

Ethereum

- Generalization of ledger
- Currency: Ether
- Attempt to make blockchain **programmable**: “world computer”

I want you to write a program that has to run in a concurrent environment under Byzantine circumstances where any adversary can invoke your program with any arguments of their choosing. The environment in which your program executes (and hence any direct or indirect environmental dependencies) is also under adversary control. If you make a single exploitable mistake or oversight in the implementation, or even in the logical design of the program, then either you personally or perhaps the users of your program could lose a substantial amount of money. Where your program will run, there is no legal recourse if things go wrong. Oh, and once you release the first version of your program, you can never change it. It has be right first time.

I don't think there are many experienced programmers that would fancy taking on this challenge. But call it 'writing a smart contract' and programmers are lining up around the block to have a go! Most of them it seems, get it wrong.[2]

Source: The morning paper, Zeus: Analyzing safety of smart contracts MARCH 8, 2018

[2] Kalra et al. ZEUS: Analyzing Safety of Smart Contracts

Ledger → Runtime

- Transactions are interpreter state transitions
- *Turing-complete*, general purpose imperative environment
- Replicate a deterministic state machine
- Programs: **Smart Contracts**
- Deployed as immutable code
  - Low-Level Lisp (LLL)
  - Solidity

Example: Solidity

```
pragma solidity ^0.4.0;
contract C {
    function isSix(uint8 num)
    returns (bool) {
        return num == 6;
    }
}
```

Ethereum Virtual Machine (EVM)

- Stack machine
- **no IO**!
- *Ephemeral* on-chain memory
- 256 bit words
- 65 logically distinct instructions [2][3]

[3] Implementation in Clojure

Gas model

- Important innovation
- **Every instruction** has a **gas price** (in Ether)
- Proportional to **memory access cost**
- Invoker of smart contract has to provide ether
- Smart contracts can call each other

What happens if gas runs out?

Problems

- still PoW (high energy cost)
- still high latency: 15 secs block time[4]
- still low throughput: (~ 100 txs/sec)

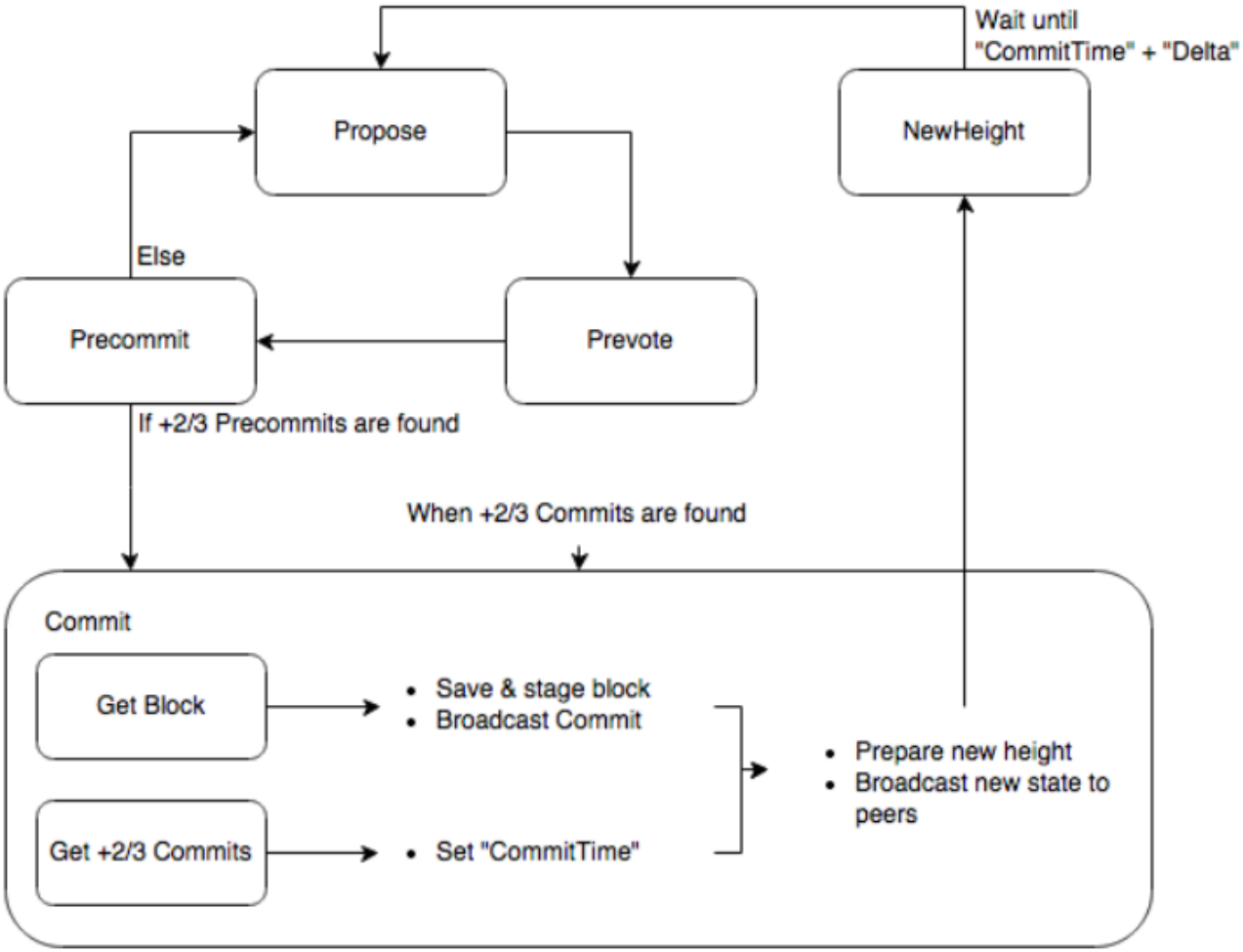
[4] https://ethstats.net/

Tendermint

- Adapted from a traditional BFT style approach[5]
- **Immediate finality**
- Low latency (~ 2 secs)
- **Fork Accountability**
- *No mining*

[5] https://github.com/tendermint/tendermint/wiki/Byzantine-Consensus-Algorithm

Tendermint state machine



Proof of Stake (PoS)

- Desire: **Get rid of wasteful mining**
- *Idea*: Replace PoW leader election by stake based voting.
- Votes are **weighted by their stake** or the money you have in your account.
- **Hard Problem**: What are economic incentives for convergence?

Delegated Proof of Stake

- *Idea*: Elect **validator nodes** who run traditional BFT consensus
- $\Rightarrow$ Small and known subnetwork
- Advantage: Higher **quality of service** (QoS) is possible with known network topology
- Problem: **Easier to attack** or less decentralized

Anonymity

- Is Bitcoin **anonymous**? Nope, rather the opposite[6]
- **Zero-knowledge proofs** (zksnarks): ZCash
- Idea: Anonymity by design using alt coins

[6] https://media.ccc.de/v/FWTYS3

Summary: Comparison

System	Consensus	Finality	Network	Fork-Acc.	Program.
Bitcoin	Nakamoto	eventual	open	no	no*
Ethereum	Nakamoto*	eventual*	open	no	yes
Tendermint	PoS-based	immediate	closed	yes	optional
Avalanche	PoS-based	immediate	open	no	optional

Outlook

- Similar to Dotcom bubble
- Majority of systems today will not survive / have not survived
- But: “Blockchains” will not go away!
- Possibility for decentralized funding (ICO, . . .)
- Possibility to build new forms of society with distributed database technology!

Applications

- Crypto currencies
- Smart property
- Smart contracts
- Identity management
- etc.

Further reading I

[1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system"*. 2009. URL: http://bitcoin.org/bitcoin.pdf.

[2] Dr. Gavin Wood. "Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain". In: (2014). URL: https://github.com/ethereum/yellowpaper.