Blockchain Tech Jumpstart

Tyr

1:20 - 2:20

- Why?
- history and status-quo
- bitcoin: an revolutionary open ledger

2:30 - 3:20

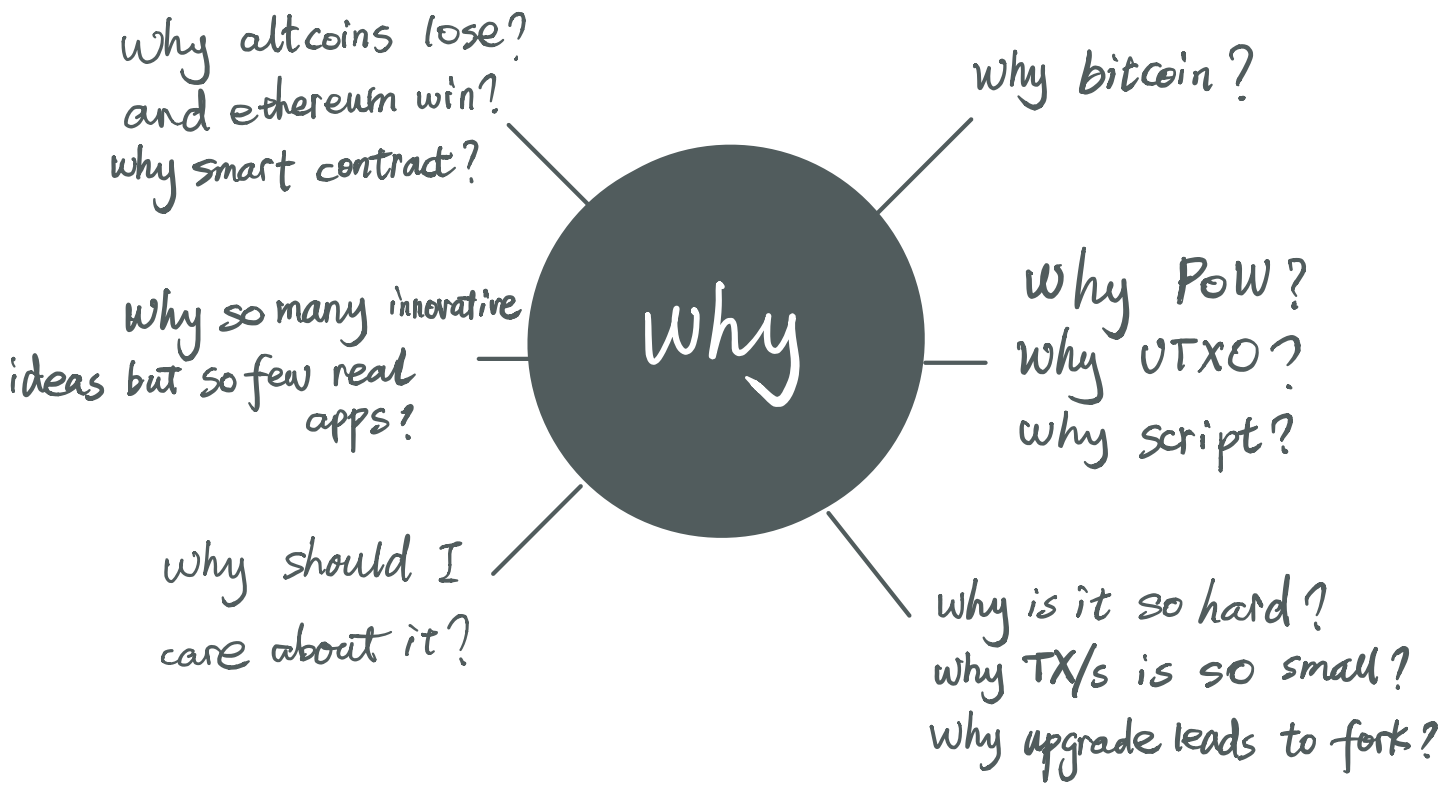
- distributed systems and consensus algo
- from bitcoin to ethereum: evolution

3:40 - 4:10

- from ethereum to ??: what's next?
- applications and innovations

4:20 - 4:40

- whitepapers in a nutshell
- how to learn blockchain tech?


why
Why altcoins lose?
and ethereum win?
why smart contract?
Why bitcoin?
Why so many innovative
ideas but so few real
apps?
Why PoW?
Why UTXO?
why script?
why should I
care about it?
Why is it so hard?
Why TX/s is so small?
Why upgrade leads to fork?

History and Status quo

Bitcoin

1997: hashcash (PoW for anti-spam)

8/18/2008: bitcoin.org registered

10/31/2008: bitcoin paper

1/3/2009: genesis block

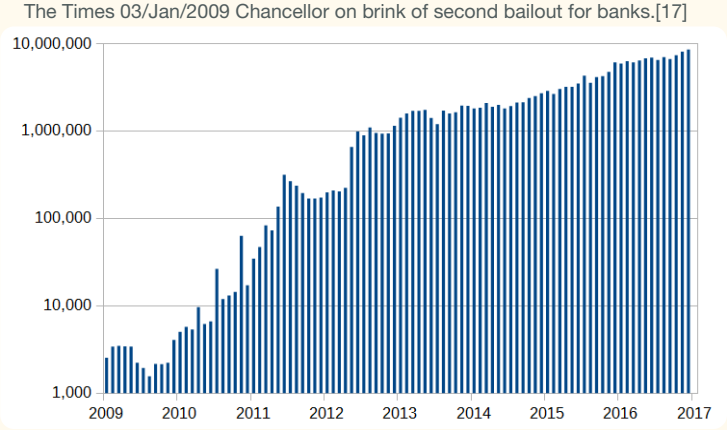
2011: wikileaks began to accept bitcoin

2/2014: Mt. Gox bankruptcy

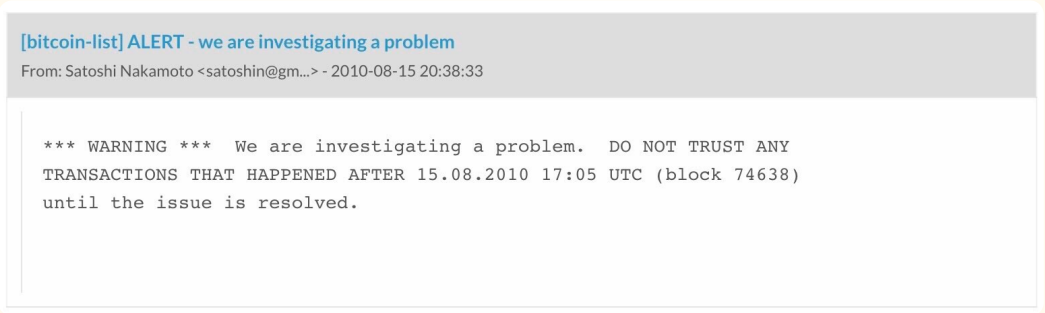
2015: Coinbase $75M raised

1/25/2016: 1 Exa·hash/s, 300,000 times of biggest supercomputer.

3/2017: github projects related to bitcoin > 10,000



On 6 August 2010, a major vulnerability in the bitcoin protocol was spotted. Transactions weren't properly verified before they were included in the transaction log or blockchain, which let users bypass bitcoin's economic restrictions and create an indefinite number of bitcoins.[25][26] On 15 August, the vulnerability was exploited; over 184 billion bitcoins were generated in a transaction, and sent to two addresses on the network. Within hours, the transaction was spotted and erased from the transaction log after the bug was fixed and the network forked to an updated version of the bitcoin protocol.[27] This was the only major security flaw found and exploited in bitcoin's history.[25][26]

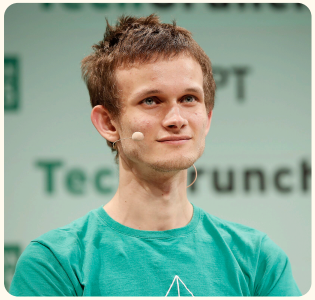


[bitcoin-list] ALERT - we are investigating a problem

From: Satoshi Nakamoto <satoshin@gm...> - 2010-08-15 20:38:33

```
*** WARNING ***  We are investigating a problem.  DO NOT TRUST ANY
TRANSACTIONS THAT HAPPENED AFTER 15.08.2010 17:05 UTC (block 74638)
until the issue is resolved.
```

Ethereum



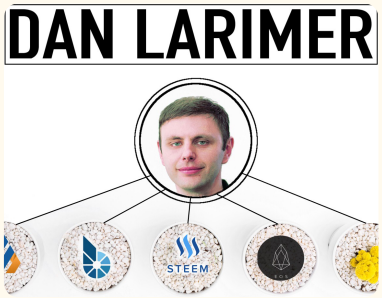
- Vitalik : bitcoin magzine
- 2013 - he wants to add a full-fledged script lang in bitcoin but was denied.
- 1/2014 - ethereum started
- 7/2015 - first PoC (olympic)
- 3/14/2016 - first stable (Homestead)
- 10/16/2017 - improve EVM (Metropolis : vByzantium)
- TBA - Casper / difficulty time bomb / EIP86 (Metropolis : vConstantinople)

hard forks

EIP86 → account abstraction

Bitshares, Steemit and EOS

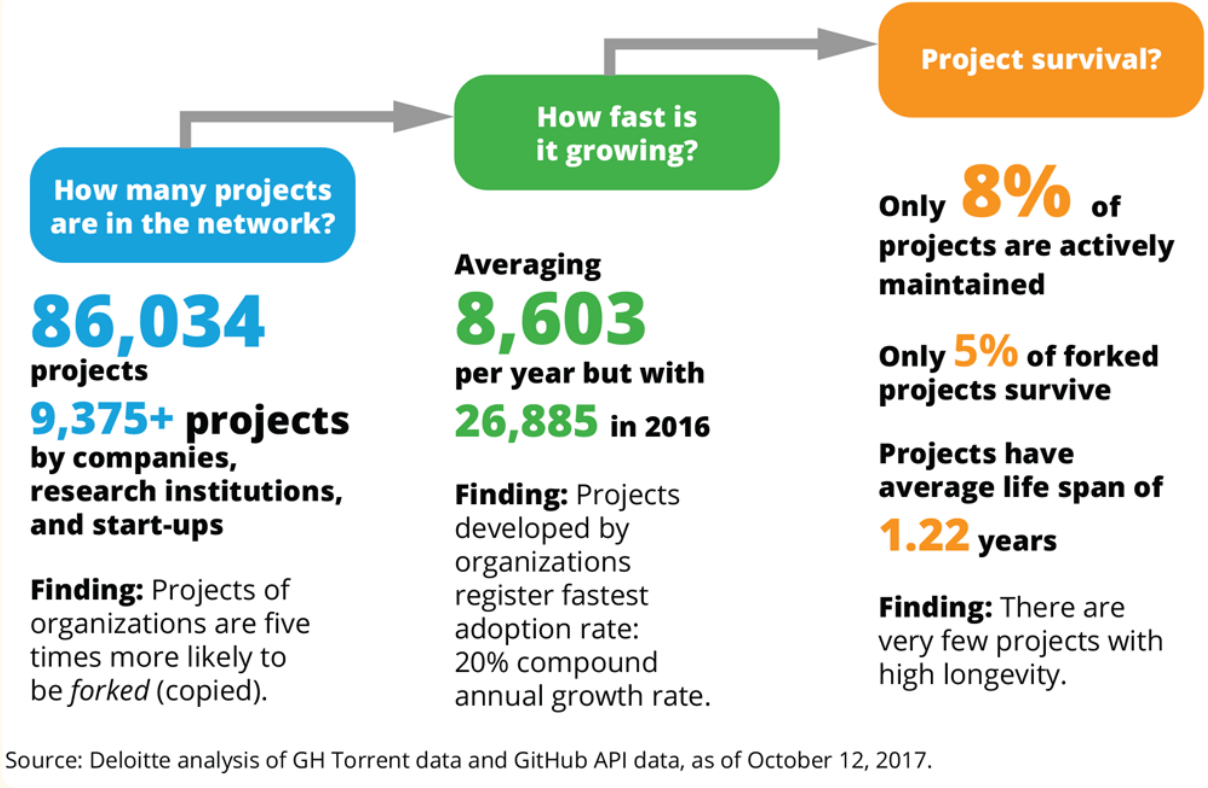
- Daniel Larimer: DPoS & graphene tech
- Bitshares: 7/19/2014 1.0
  10/13/2015 2.0 (Graphene)
- Steemit: 7/4/2016
- EOS: 2017: whitepaper



Comparison

	Timestamp/ consensus	block time	use cases	features
Bitcoin	PoW/sha256	10min	coin transfer, multisig	UTXO, limited supply
ethereum	PoW/ethash	14-15sec	ico, dApps.	smart contract
bitshares	DPoS	3 sec	DEX, referral reward	fast/scalable 100k TX/s
Steemit			social network	rewarding by activity, PoB
EOS	DPoS	500ms	OS for running app like steemit	21 super node, webAssembly

ethash: sha-3



Source: Deloitte analysis of GH Torrent data and GitHub API data, as of October 12, 2017.

Table 1. Top 20 central repositories in the blockchain ecosystem

Project name	Author	User type	Total copies	Followers	Total contributors	Description
bitcoin	Bitcoin	Organization	7,588	11,729	627	Source code behind Bitcoin
go-ethereum	Ethereum	Organization	1,717	5,603	149	Official Go implementation of the Ethereum protocol
bitcoinjs-lib	bitcoinjs	Organization	500	1,478	62	Bitcoin-related functions implemented in pure JavaScript
Electrum	spesmilo	Organization	534	1,028	187	Bitcoin thin client on Electrum (wallet)
cpp-ethereum	Ethereum	Organization	905	1,332	119	Ethereum C++ client
bips	Bitcoin	Organization	527	762	158	Bitcoin improvement proposals
bitcoinj	bitcoinj	Organization	928	1,243	106	Java implementation of Bitcoin
Rippled	Ripple	Organization	397	1,267	54	Decentralized cryptocurrency blockchain daemon implementing the Ripple Consensus Ledger in C++
mist	Ethereum	Organization	612	2,752	47	Browse to explore Decentralized Apps (DAPPs) built on Ethereum
truffle	Consen-Sys	Organization	181	768	31	Development environment, testing framework, and asset pipeline for Ethereum
pyethereum	Ethereum	Organization	348	1,135	57	Python core library of the Ethereum project
cgminer	Ckolivas	User	737	1,760	88	ASIC and FPGA miner in C for Bitcoin
ethereumj	Ethereum	Organization	350	598	58	Java implementation of the Ethereum yellowpaper
btcd	Btcsuite	Organization	380	1,272	59	An alternative full node Bitcoin implementation written in Go (golang)
testrpc	ethere-umjs	Organization	154	559	33	Fast Ethereum RPC client for testing and development
bitcoinbook	bitcoin-book	Organization	735	2,204	40	Bitcoin book
ripple-client	Ripple	Organization	500	1,239	54	A UI for the Ripple payment network built using web technologies
EIPs	Ethereum	Organization	148	650	38	The Ethereum improvement proposal
embark-framework	iurimatias	User	98	515	37	Framework for serverless Decentralized Applications using Ethereum, IPFS, and other platforms
bcoin	bcoin-org	Organization	121	649	20	JavaScript Bitcoin library for node.js and browsers

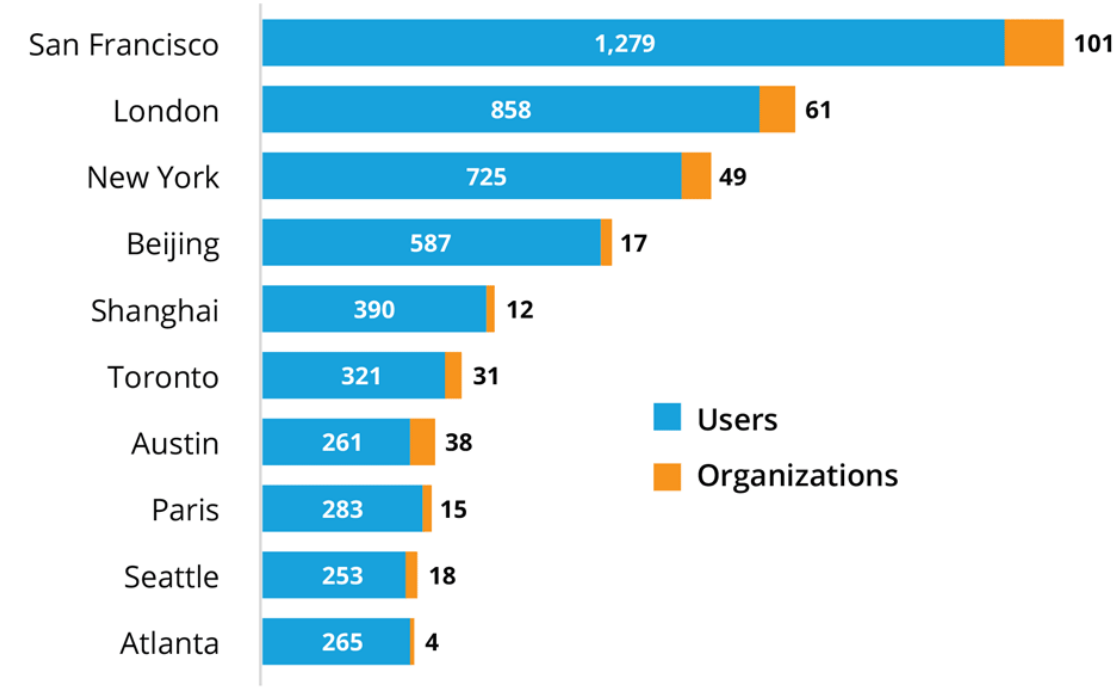
Source: Deloitte analysis of GH Torrent data and GitHub API data.

Top 20 repos are dominated by:

- bitcoin
- ethereum

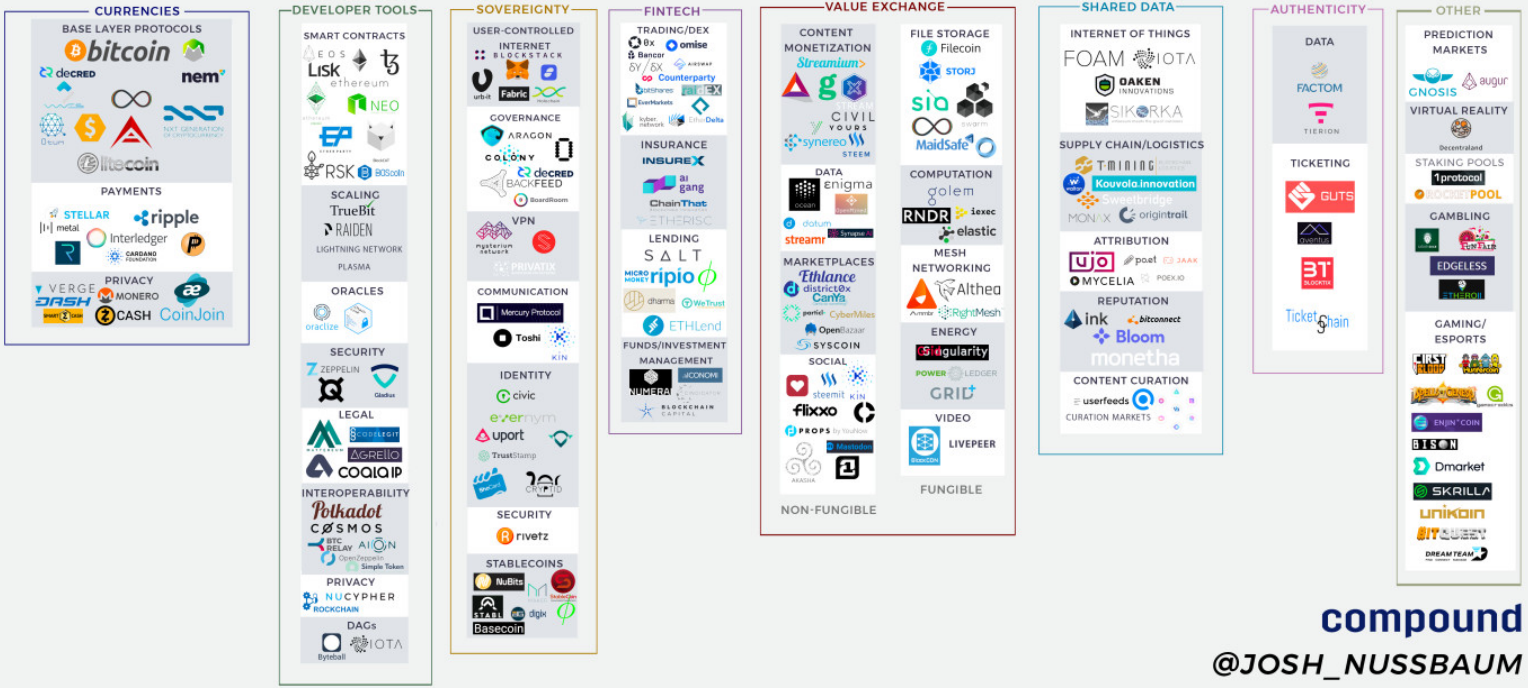
Most of projects are written in:

- C++
- golang



Source: Deloitte analysis of GH Torrent data and GitHub API data, as of October 12, 2017.

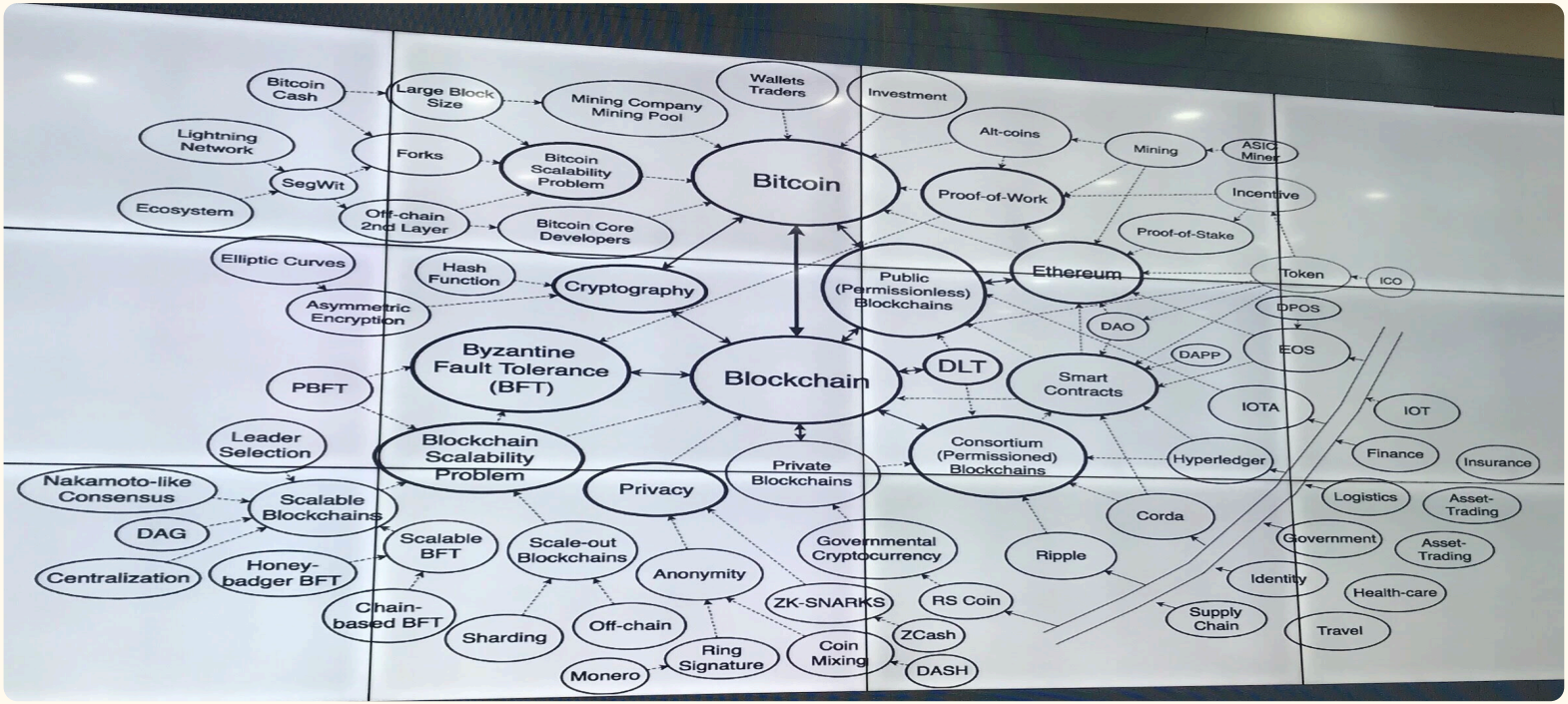
BLOCKCHAIN PROJECT ECOSYSTEM



compound

@JOSH_NUSSBAUM

from techcrunch : Mapping the blockchain ecosystem

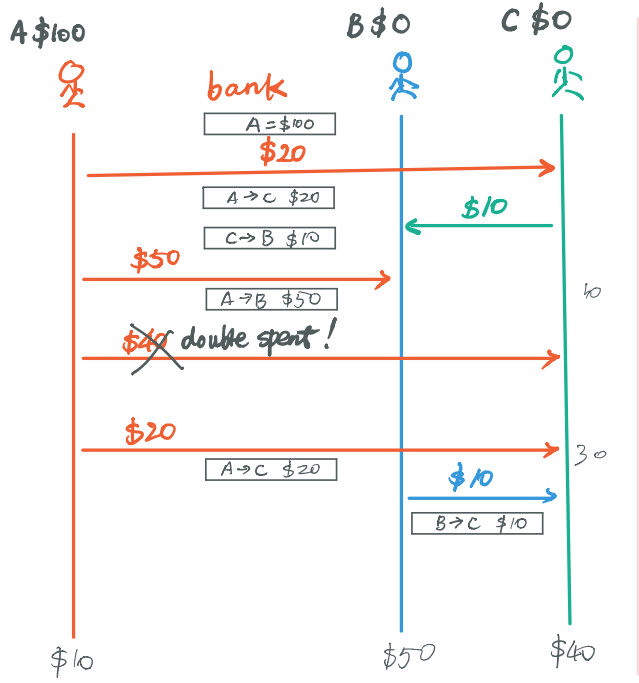

Bitcoin Cash
Large Block Size
Mining Company Mining Pool
Wallets Traders
Investment
Lightning Network
Forks
Bitcoin Scalability Problem
Bitcoin
Alt-coins
Mining
ASIC Miner
SegWit
Ecosystem
Off-chain 2nd Layer
Bitcoin Core Developers
Proof-of-Work
Incentive
Proof-of-Stake
Elliptic Curves
Hash Function
Cryptography
Public (Permissionless) Blockchains
Ethereum
Token
ICO
Asymmetric Encryption
DPOS
DAO
EOS
Byzantine Fault Tolerance (BFT)
Blockchain
DLT
DAPP
Smart Contracts
PBFT
IOTA
IOT
Leader Selection
Blockchain Scalability Problem
Consortium (Permissioned) Blockchains
Private Blockchains
Hyperledger
Finance
Insurance
Nakamoto-like Consensus
Privacy
Logistics
Asset-Trading
Scalable Blockchains
Corda
DAG
Scalable BFT
Scale-out Blockchains
Governmental Cryptocurrency
Government
Asset-Trading
Ripple
Centralization
Honey-badger BFT
Anonymity
Identity
Health-care
ZK-SNARKS
RS Coin
Chain-based BFT
Off-chain
Supply Chain
Sharding
ZCash
Travel
Ring Signature
Coin Mixing
Monero
DASH

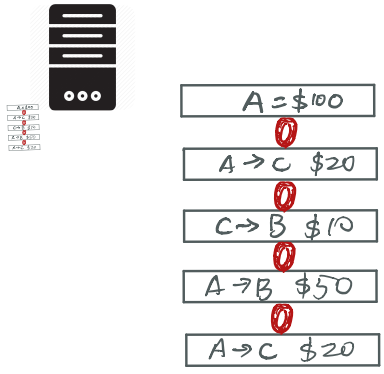
bitcoin: an revolutionary open ledger

Money transfer

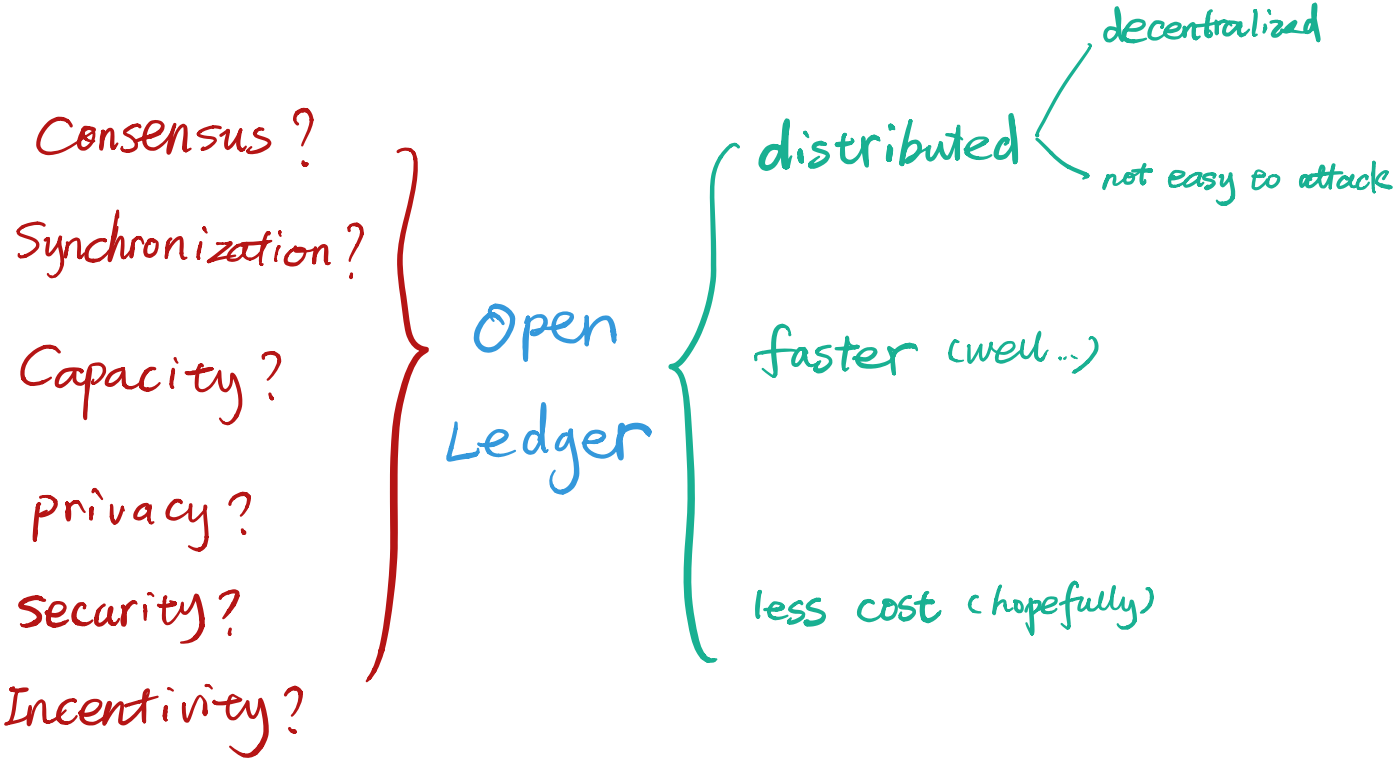
① takes too long
② fee too high

Ledger

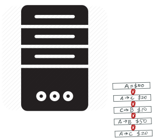


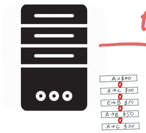


The history of the transaction is Currency.


Consensus ?
Synchronization ?
Capacity ?
privacy ?
Security ?
Incentivity ?
Open Ledger
distributed
decentralized
not easy to attack
faster (well...)
less cost (hopefully)

Why a node wants to keep your record?



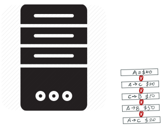


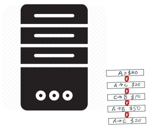
to keep a transaction →

① validate
② add to chain
③ publish

} work

compete and reward





To compete: - Gamification

1. solve a problem
2. put result, transaction and reward in a block
3. chain it.

The reward:

₿50

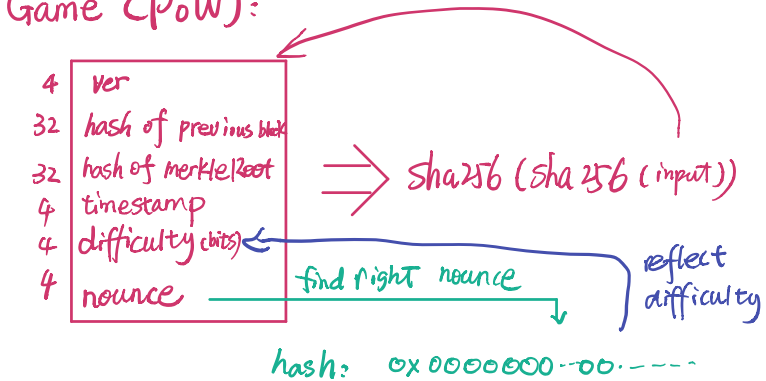
Now 12.5

- incentify system

How would you build it?

bytes	
4	magic : 0xd9b4bef9
4	length
4	ver 0x1
32	sha256 (previous block)
32	merkleRoot
4	timestamp : creation time of the block
4	difficulty : 3,007,383,866,589.732
4	nounce : the problem to solve
	transaction count
	transactions

Game (PoW):



what we learned:

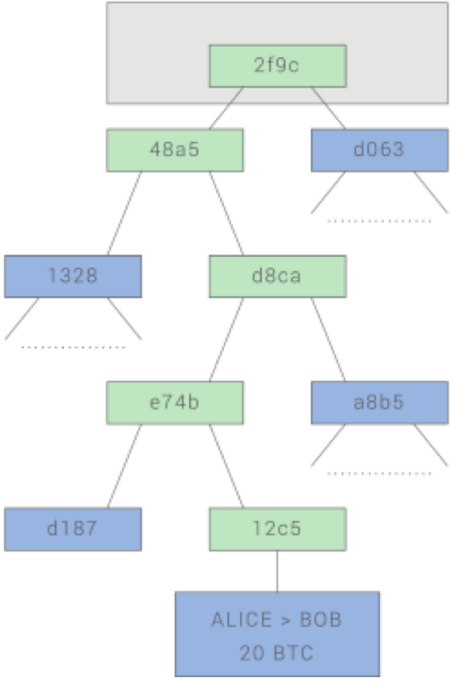
1. to find a "nounce" is hard (random)
2. by tuning difficulty, the system can limit the generation of a block to close to 10min.
3. merkle Tree + bloomfilter made wallet applicable
4. chaining + PoW greatly reduced the chance of "do evil thing".
5. "incentive" encourage nodes to be good.

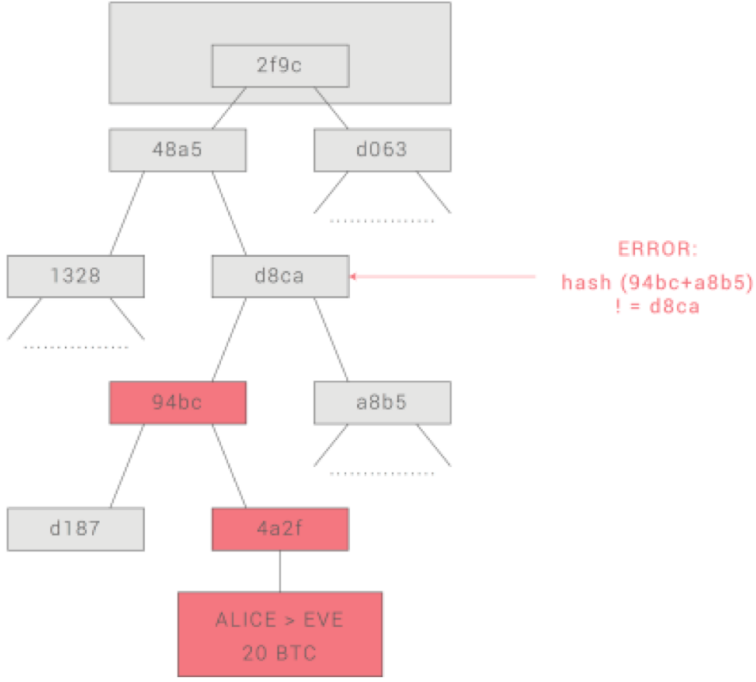
What problems have PoW solved?

- Global clock (Timestamp server)
- consensus - who and when to append to the ledger
- Security of the ledger (alter is almost impossible)
- incentification - why node do the work
- currency circulation - how new coins are published

What is Merkle Tree?

I just need to know $\log_2 N$ hashes





Now, what is a block?

- A data structure contains TXs
- TXs are organized as merkle tree
- Blocks are linked with previous work
- Blocks in longest chain excel
- Always contains a coinbase TX
- Statistically, a block is confirmed by next 6 blocks

Genesis Block

Recent Block

Block #1

BlockHash 00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048

Summary

Number Of Transactions	1	**Difficulty**	1
Height	1 (Mainchain)	**Bits**	1d00ffff
Block Reward	50 BTC	**Size (bytes)**	215
Timestamp	Jan 8, 2009 6:54:25 PM	**Version**	1
Mined by		**Nonce**	2573394689
Merkle Root	0e3e2357e806b6cdb1f70b54c3a3a...	**Next Block**	2
Previous Block	0		

Transactions

0e3e2357e806b6cdb1f70b54c3a3a17b6714ee1f0e68bebb44a74b1efd512098 — mined Jan 8, 2009 6:54:25 PM

No Inputs (Newly Generated Coins) → 12c6DSiU4Rq3P4ZxziKxzrL5LmMBrzjrJX 50 BTC (U)

Type pubkeyhash

scriptPubKey

0496b538e853519c726a2c91e61ec11600ae1390813a627c66fb8be7947be63c52da7...

510273 CONFIRMATIONS | 50 BTC

Block #510266

BlockHash 00000000000000000004f7aeba4f2ea6027343e045e1b86120093d9cdd32ebe7e

Summary

Number Of Transactions	563	**Difficulty**	3007383866429.732
Height	510266 (Mainchain)	**Bits**	175d97dc
Block Reward	12.5 BTC	**Size (bytes)**	992316
Timestamp	Feb 21, 2018 8:50:46 AM	**Version**	536870912
Mined by	BTCC Pool	**Nonce**	2966267765
Merkle Root	38967cc76b631ba5fee39e6849711...	**Next Block**	510267
Previous Block	510265		

Transactions

69e192ee04d95a74d9b16babc857255ffd9754bf2c81bbb963995d6e7f867e56 — mined Feb 21, 2018 8:50:46 AM

No Inputs (Newly Generated Coins) → 13TEThZNnKPk34HYAuo1QqYMwDdjF3qeHx 12.63484184 BTC (U)

Unparsed address [0] 0 BTC (U)

8 CONFIRMATIONS | 12.63484184 BTC

6a7eb02a56f159388cc037ba38616179c8b4b2cc5989cb39a28aebd326f9709b — mined Feb 21, 2018 8:50:46 AM

179kGoYHaHgFHmPdmG6JQD43wBFUMmFvTF 0.03619554 BTC → 12iVTz5rj1TpxXvaWrWFAxNwGKiXznYvsU 0.03619554 BTC (U)

FEE: 0 BTC

8 CONFIRMATIONS | 0.03619554 BTC

2ef404706fa4d7617d918c6260a79de69d7d57a6ed4ea63843e6616525ae9707 — mined Feb 21, 2018 8:50:46 AM

1H7ncK2i4RwdbuwCKDGwFmmthNFFB6Sb9H 29.73215027 BTC → 14piQFZmXGPVN8AHbDmGj4qqjeXSgyBywV 29.67034227 BTC (U)

17DyiI25puA5hdnk2S4nHim8HASeFLKx29 0.06 BTC (U)

FEE: 0.001808 BTC

8 CONFIRMATIONS | 29.73034227 BTC

14ef4ceda1d966e085ec1ea147821d1c65065f021428d7039a73666d93c6f2 — mined Feb 21, 2018 8:50:46 AM

A Tx

Transaction

Transaction cb671e761a9ede0550af141c01d06b4f1475a8816ae60906f5064c2972d8c86c

Summary

Size	226 (bytes)
Fee Rate	0.00058999999999999999 BTC per kB
Received Time	Feb 21, 2018 10:03:44 AM
Mined Time	N/A
Included in Block	Unconfirmed

Details

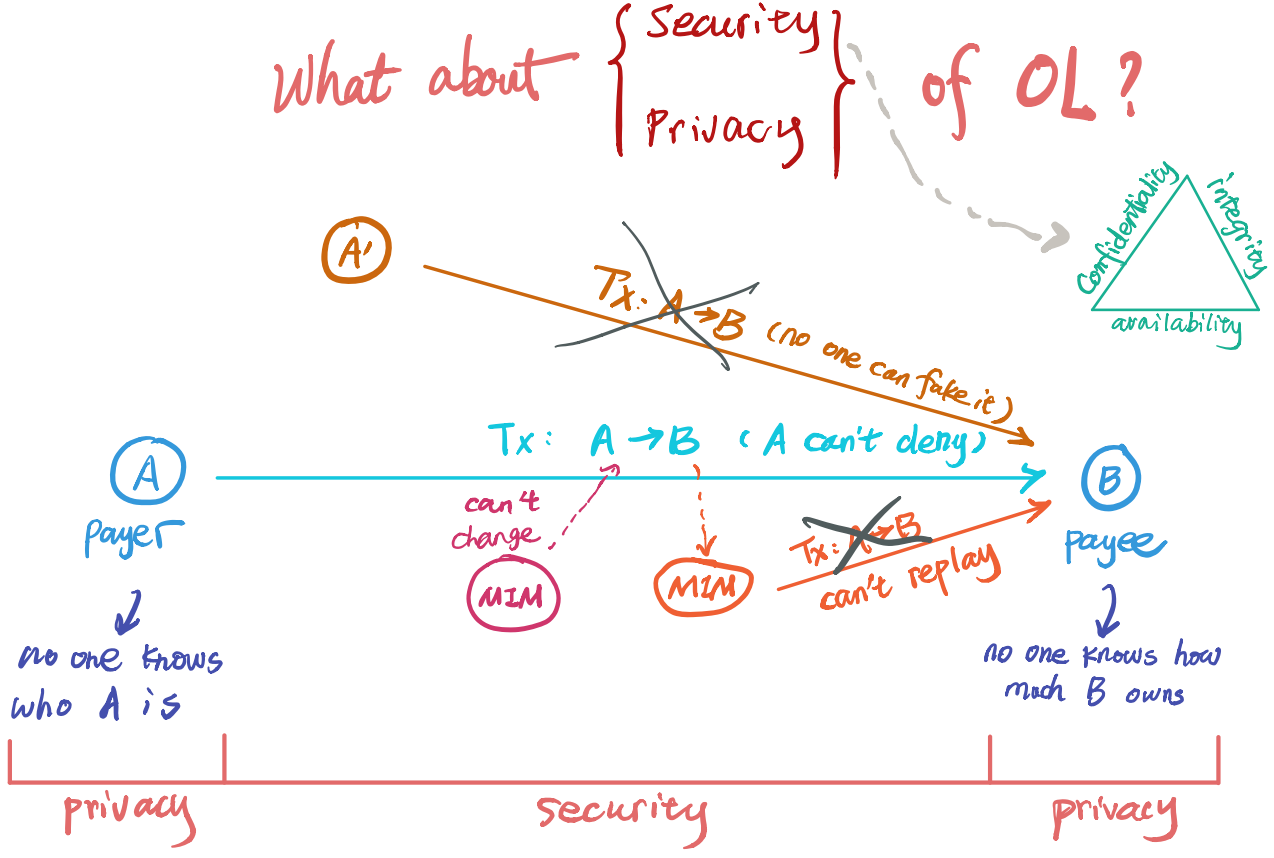
cb671e761a9ede0550af141c01d06b4f1475a8816ae60906f5064c2972d8c86c

14dKmeMV4HmpbbSbkeeYG524nNuEFP6M5Q	4.43931125 BTC	18La8SWywMxm5ibVpzgWDVYkGwSj1HPPAi	0.03584102 BTC (U)
		1KaTbQarFdLTkCUQ5beRaVyZ7UvRxKcnoE	4.40333689 BTC (U)

FEE: 0.00013334 BTC

UNCONFIRMED TRANSACTION! 4.43917791 BTC

fee = input(s) - output(s)

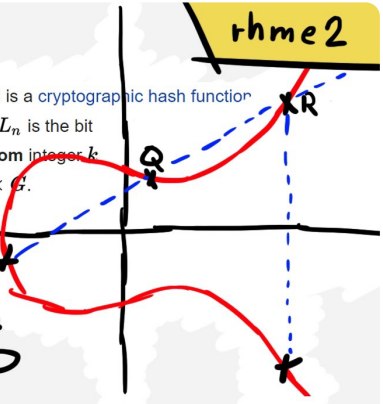

What about { Security, Privacy } of OL?
Confidentiality
integrity
availability
A'
Tx: A→B (no one can fake it)
Tx: A→B (A can't deny)
A
Payer
B
Payee
can't change
MIM
MIM
Tx: A→B
can't replay
no one knows who A is
no one knows how much B owns
privacy
security
privacy

What do we have?

ECDSA

1. Calculate $e = \text{HASH}(m)$, where HASH is a cryptographic hash function
2. Let z be the L_n leftmost bits of e, where L_n is the bit
3. Select a **cryptographically secure random** integer k
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \mod n$. If $r = 0$, go
6. Calculate $s = k^{-1}(z + rd_A) \mod$
7. The signature is the pair (r, s).

$(x_1, y_1) = k \times G$

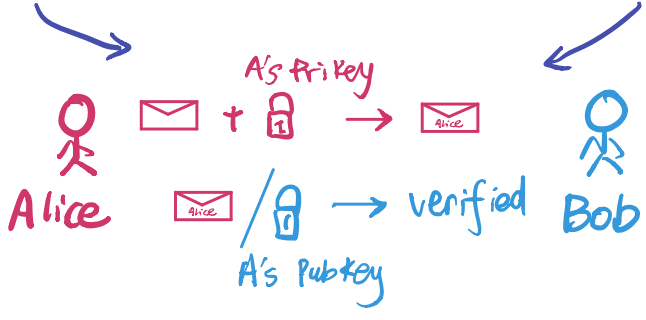


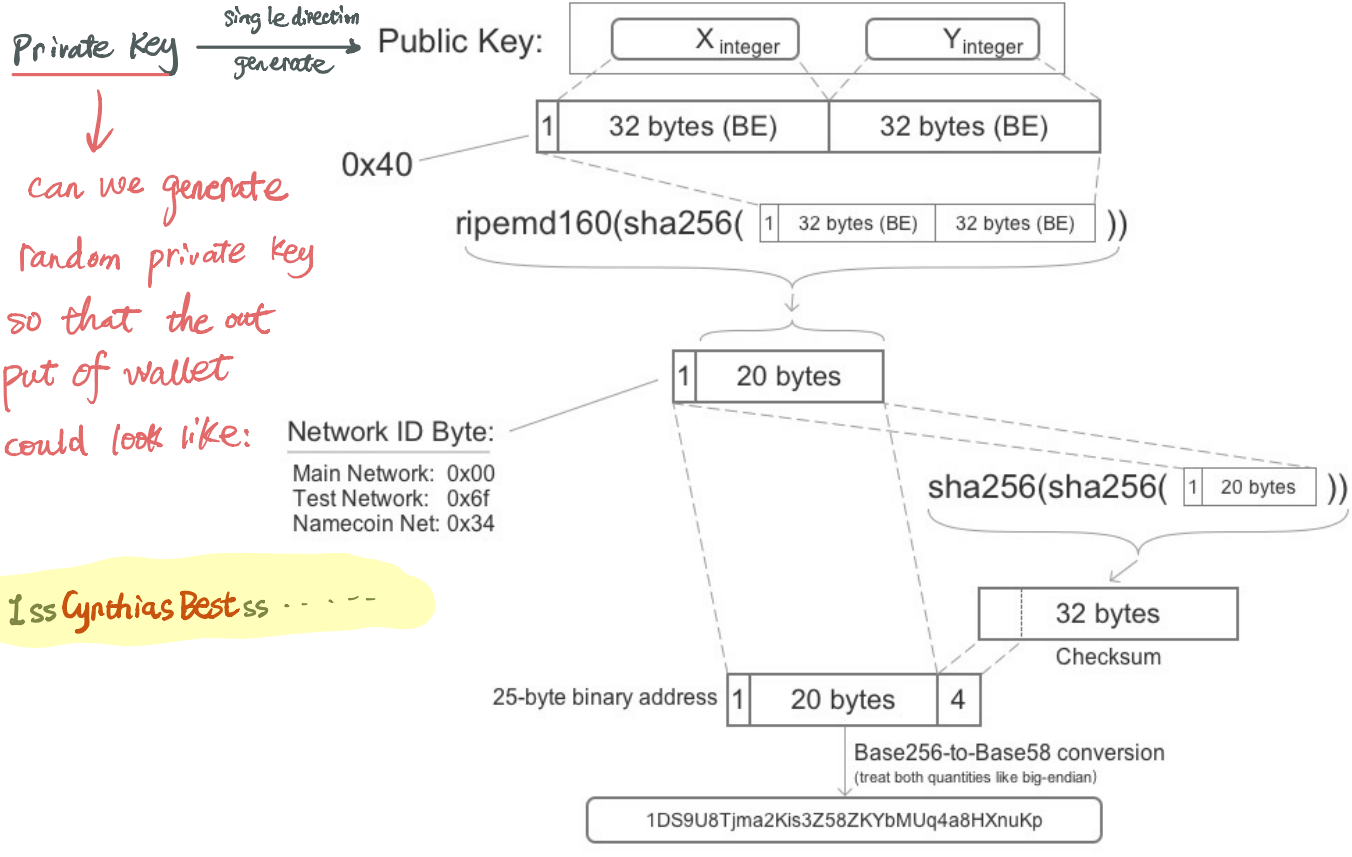
secp256k1

RSA

- Select two large prime numbers p, q
- Compute
  - n = p × q
  - v = (p-1) × (q-1)
- Select small odd integer k relatively prime to v
  - gcd(k, v) = 1
- Compute d such that
  - (d × k)%v = (k × d)%v = 1
- Public key is (k, n)
- Private key is (d, n)

- example
  - p = 11
  - q = 29
  - n = 319
  - v = 280
  - k = 3
  - d = 187
- public key
  - (3, 319)
- private key
  - (187, 319)





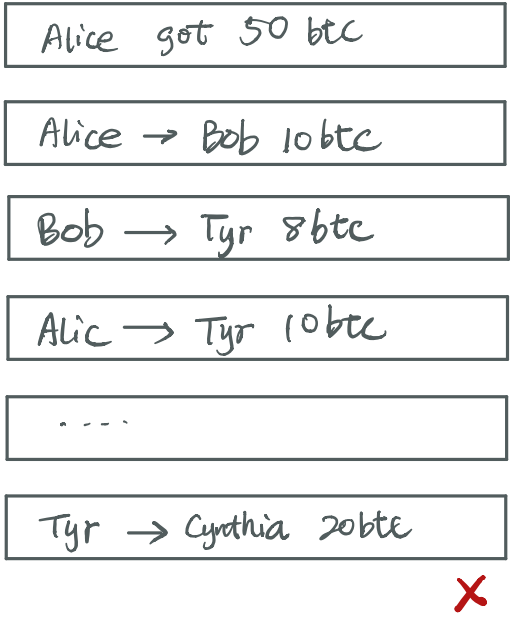
can we generate random private key so that the output of wallet could look like:

Iss Cynthias Bestss - - - - -

Who is responsible for accounts creation?

- You
- offline is possible (why?)
- What if priv key conflict?
- What if wallet address conflit?

How to verify that Tyr has enough coin?



getBalanceOf(Tyr) = 18 btc

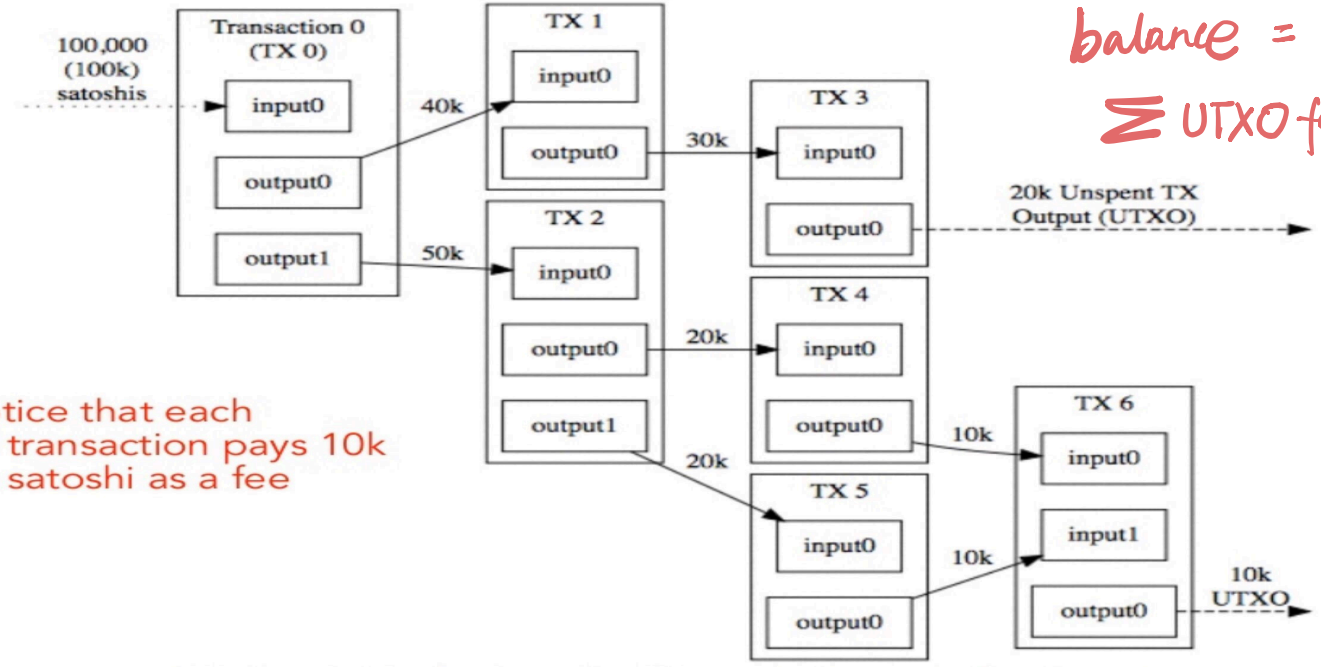
Problem: very inefficient

solution: UTXO

whe Alice sent 10 btc to Bob
she created a UTXO that only
Bob can use it create new UTXO

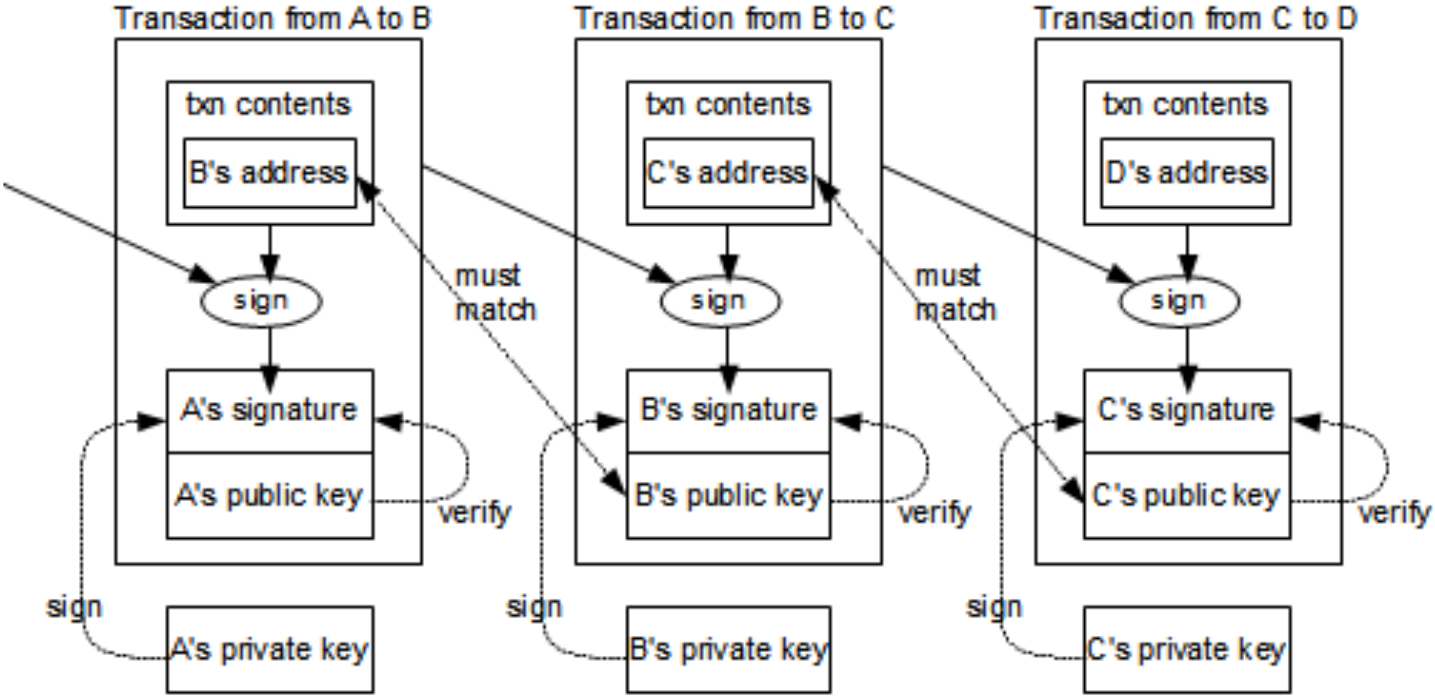
What is UTXO and why?

balance = $\sum$ UTXO for me



notice that each transaction pays 10k satoshi as a fee

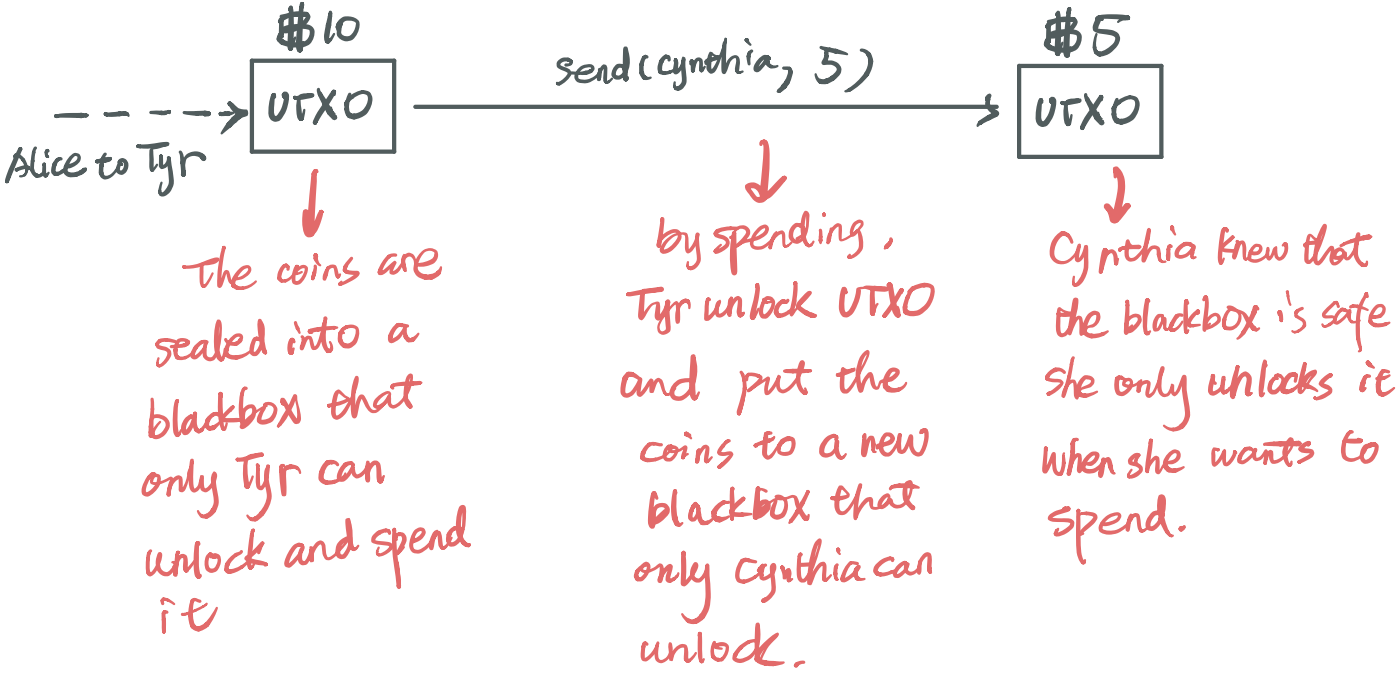
How transactions work?



How many types of Tx? and why?

- P2PKH
- P2PK
- P2SH

What is lock/unlock, and why?



Pay-to-Public-Key-Hash

Tyr $\xrightarrow{฿2}$ Cynthia

Lock:

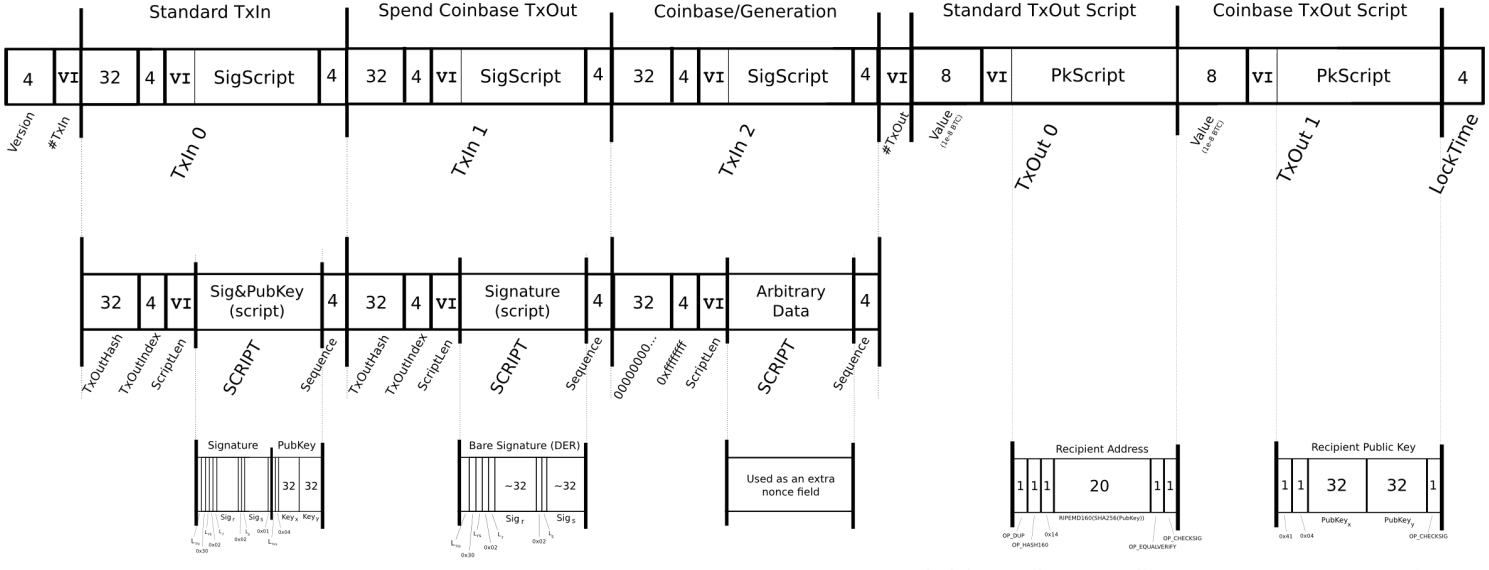
OP_DUP OP_HASH160 <Bob PK hash> OP_EQUAL OP_CHECKSIG

if you want to spend this, you need to prove you own the priv key to the address

unlock:

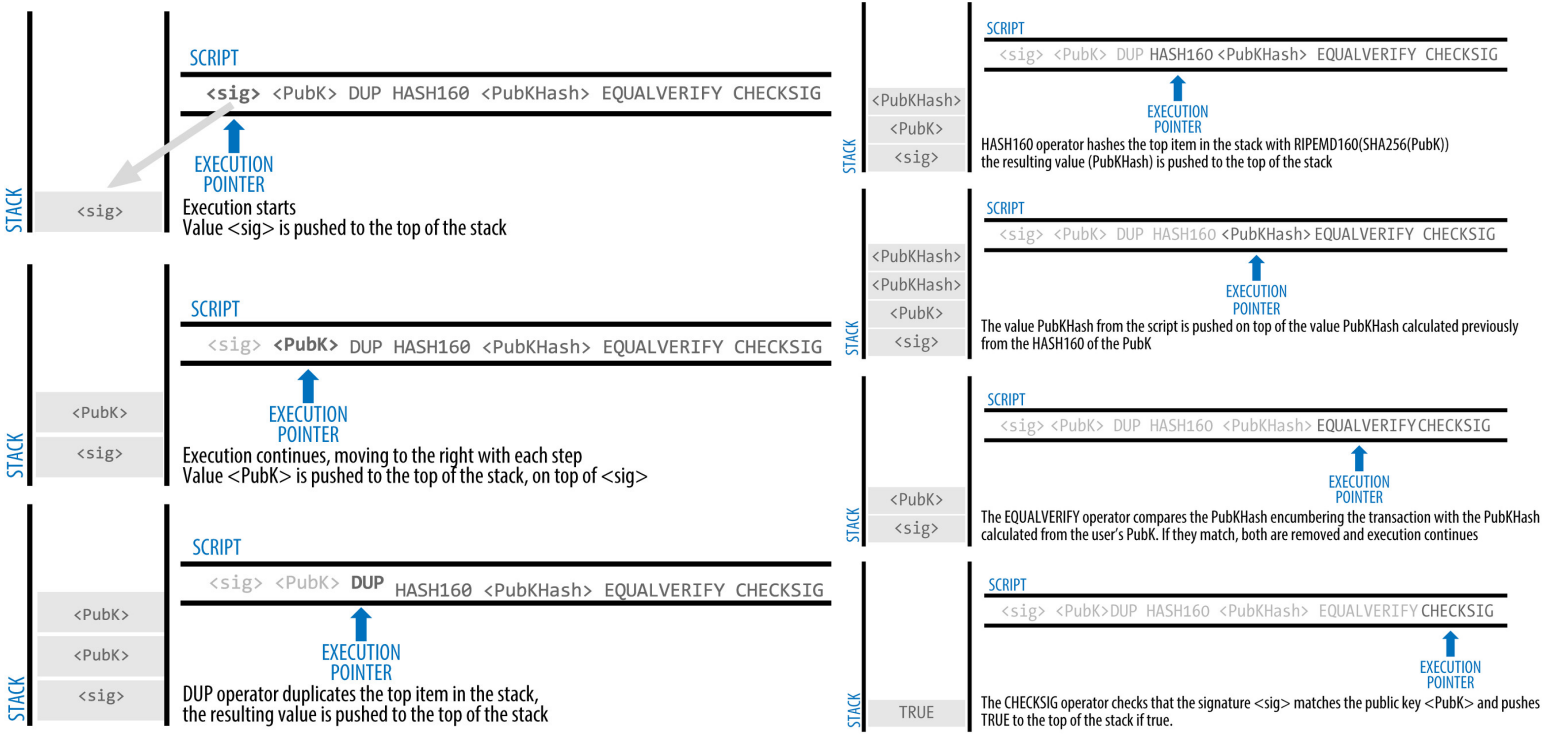
<Bob's signature> <Bob's pub key>

What does a TX look like?



Scripts and DER encoding both use big-endian values, all other serializations use little-endian

And how it gets executed?



What about security of the ecosystem?

- bad nodes
- bad nodes > 51% of computing power
- chain is too big to fit a commodity node (availability)
- Threat from Quantum computing

SPV ← Simplified Payment Verification

Shor's algo

basics of RSA

454243391453

1299721 x 349493

$O(2^{K/2}) \rightarrow O(k^3)$

could be used in ECDSA

Grover's algo

given [a, b, ----]

$O(N) \rightarrow O(\sqrt{N})$

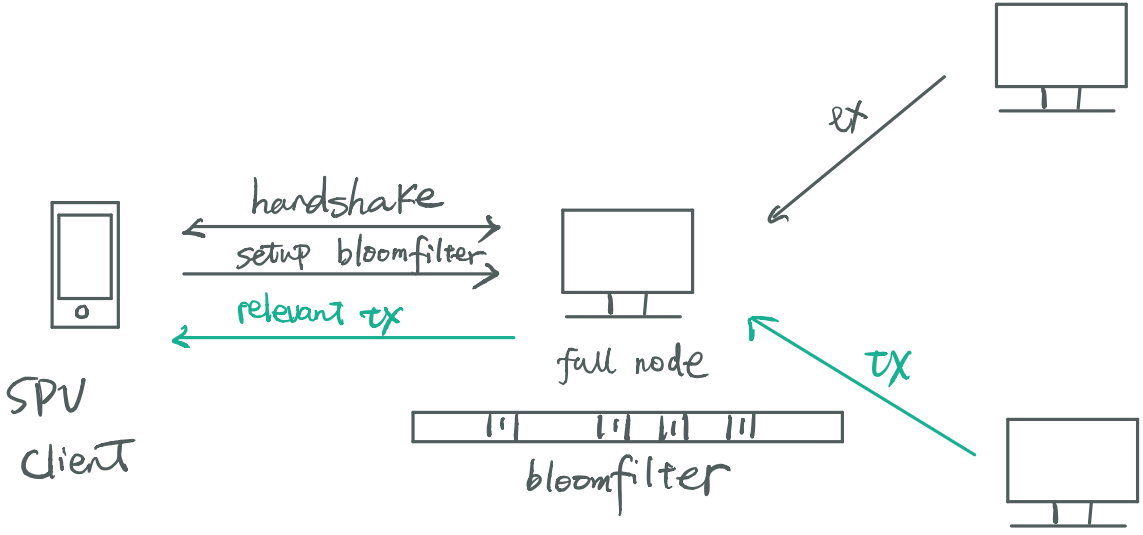
Any problem

Crack 256 bit priv key : classic 340 T T T (T = Trillion)

a few hundred Million	18 million T

hash of pub key is important!

What is SPV, why and how?

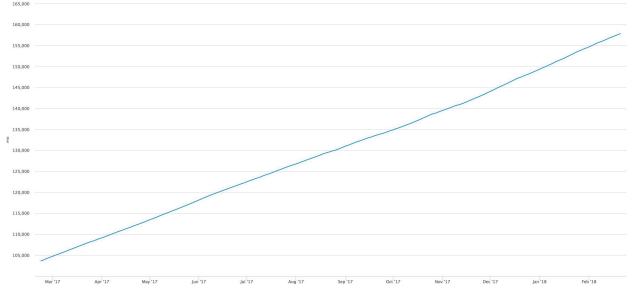


How does a client join the network?

- Seed server (resolved by DNS)
  - seed.bitcoin.sipa.be (hard coded in code)
- Download a list (up to 1000) peers
- Listen on TCP/8333
- Message are received and relayed

What about capacity?

- 1 M block added 2010
- 10 Min a block
  (6×24×365×9 = 474336) latest : 510438
- Wallet just need headers and related Merkle Tree path



why not bigger?

1MB × 8bit × 7peers/30sec = 1.86Mb/s

8MB × 8bit × 7peers/30sec = 15Mb/s

upload speed

Genius Design in Bitcoin

- Chained ledger
- Proof-of-Work
- UTXO
- scripting
- incentification

- Merkle-Tree supported SPV
- Hashed public key (hide the flaw of ECDSA)
- 21,000,000 total coins
  = 2.1 quadrillion satoshis $\approx 2^{50.9} < 2^{53}$
- base58 - why why why!
  - human readable (0, O, I, l striped)

What can we learn ?

- ask questions
- ask great questions
- ask great questions that future oriented
- Think it through and connect the dots

distributed systems and consensus algo

What ?

Definition

- A collection of computers running in a network
- communicate and coordinate towards a goal

Things to Consider

- Communication / coordination
- fault tolerance (software crash, hardware failure, network down)
- Scalability

Concepts

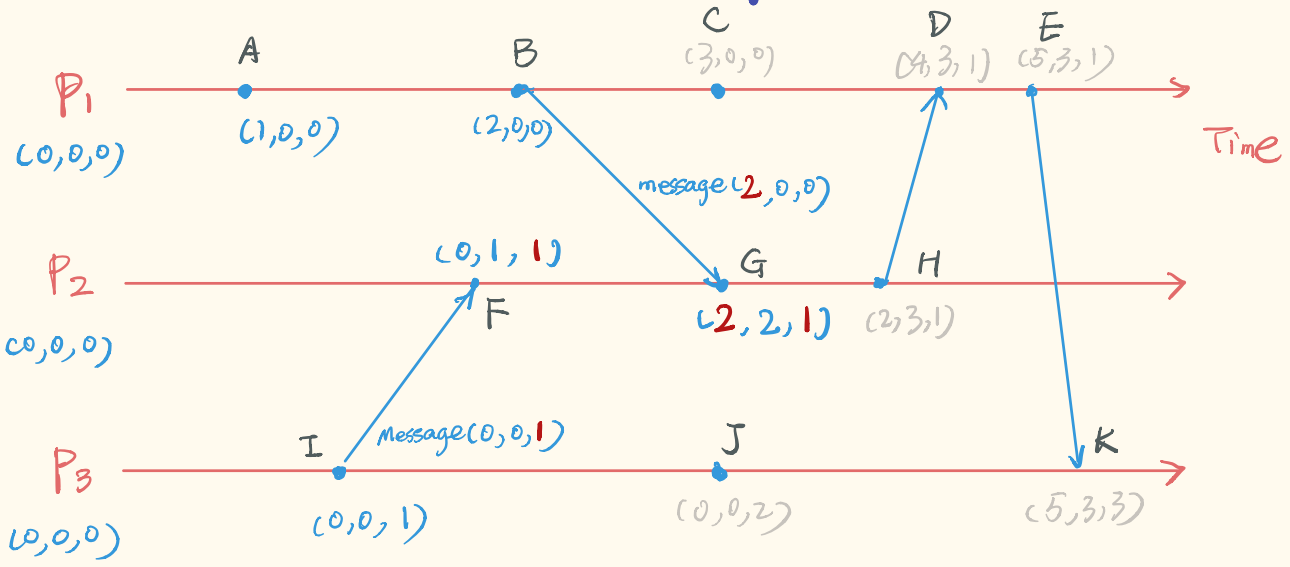
- Node : an independent computer
- Network : the channel for communication
  - could be heterogeneous
  - communication has a cost!
- Protocol : rules for transmitting information

Comparison

	Solo	Cloud computing	fully distributed
global clock	strict	yes	no
environment	simple, controlled	managed, a little moving parts	nodes come and go
admin	single	single	no
security	easy to implement	harder	hardest
fault-tolerance	bad	good	best
protocol	n/a	private/easy to change	public/hard to change
peer discovery	n/a	static/dynamic (consul)	gossip/
upgrade	easy	rolling / blue-green	hard (consensus)
consensus	n/a	static/paxos/zab/raft	PBFT/PoX

Why global clock matters?

Vector Timestamp (trust env)



Causally-related

$A \rightarrow B \quad (1,0,0) < (2,0,0)$

$B \rightarrow G \quad (2,0,0) < (2,2,1)$

Concurrent

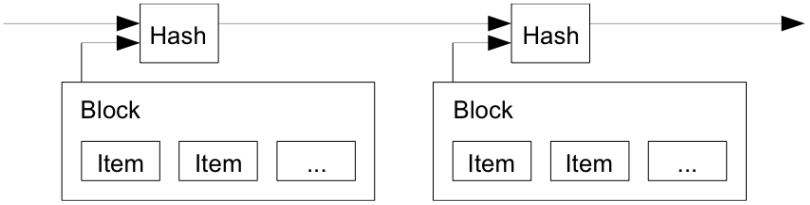
$C \,\&\, G \quad (3,0,0) \parallel (2,2,1)$

$I \,\&\, C \quad (0,0,1) \parallel (3,0,0)$

What about an untrust env?

3. Timestamp Server

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



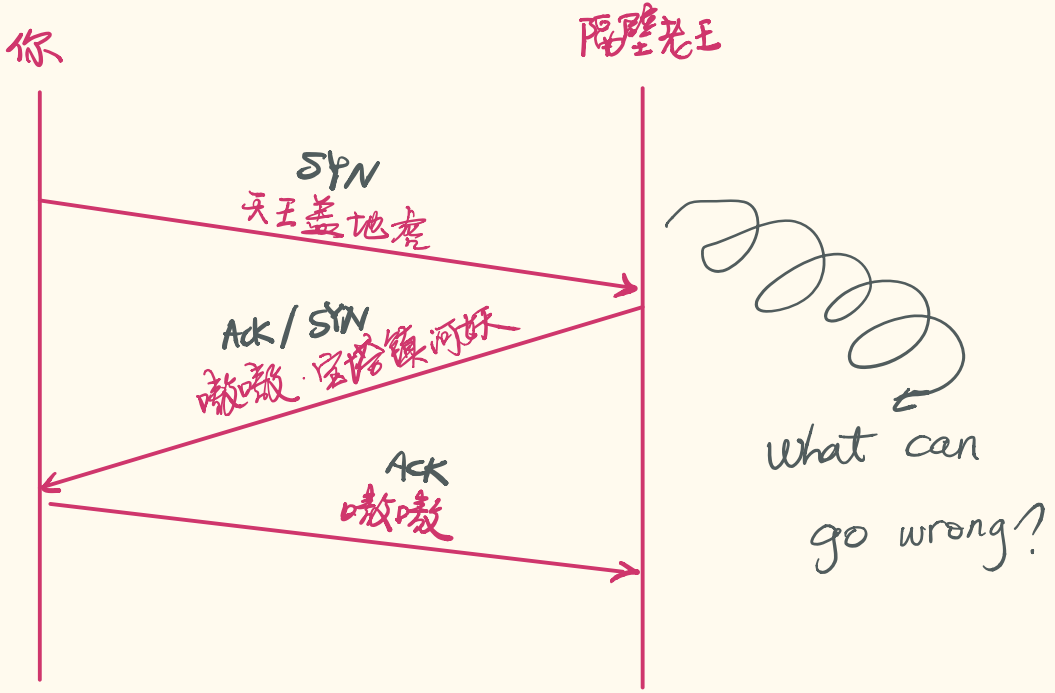
p2, bitcoin paper

Why have I never {heard / cared} about it?

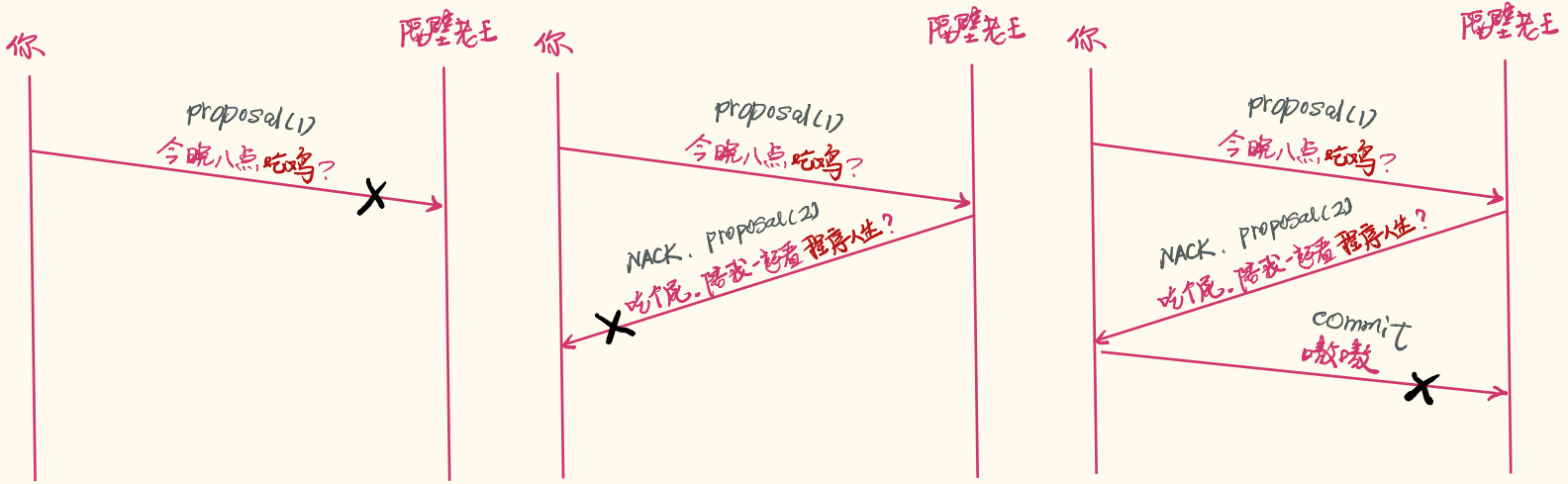
- DB (e.g. postgres)
- MQ (e.g. kafka)
- service discovery tool (e.g. consul)
- ...

hided those details for you

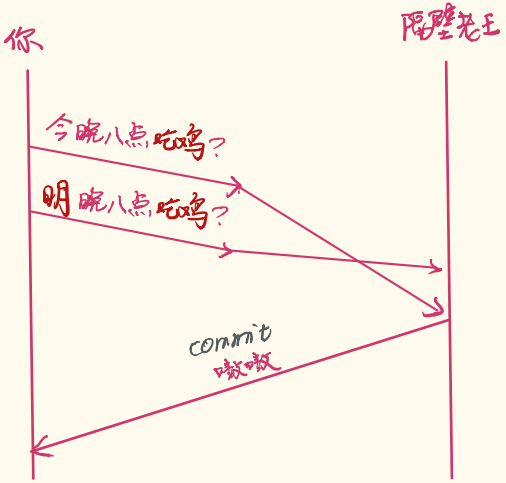
Coodination & communication

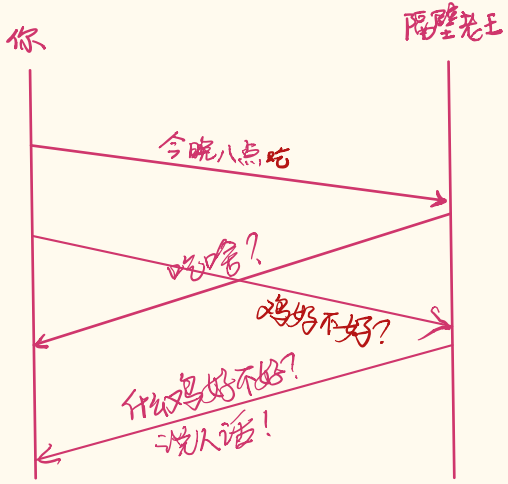


Message Lost

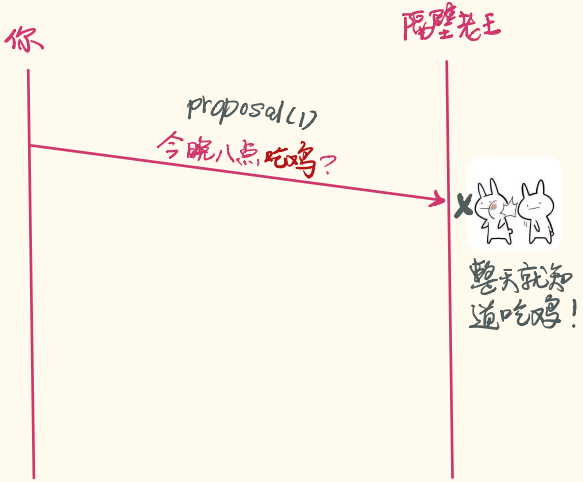


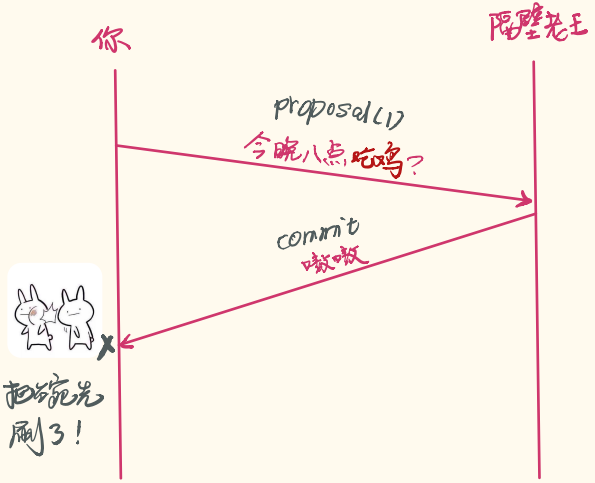
Out of order


你
隔壁老王
今晚八点吃鸡?
明晚八点吃鸡?
commit
啦啦啦


你
隔壁老王
今晚八点吃
吃啥?
鸡好不好?
什么鸡好不好?
说人话!

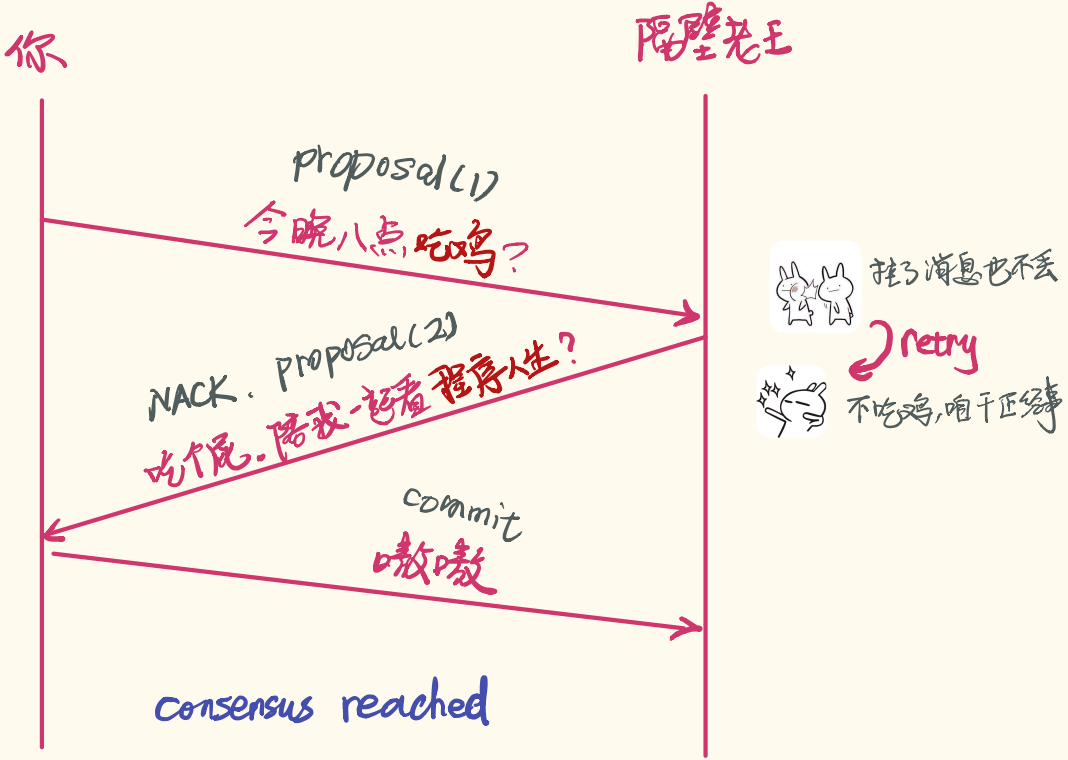
Application Crash





Message Passing

- at least once (1-n)
- at most once (0-1)
- essentially once { visibility timeout / delete manually


你
隔壁老王
proposal(1)
今晚八点吃鸡?
挂了消息也不丢
retry
不吃鸡,咱干正经事
NACK, proposal(2)
吃俩,陪我一起看程序人生?
commit
嗷嗷
consensus reached

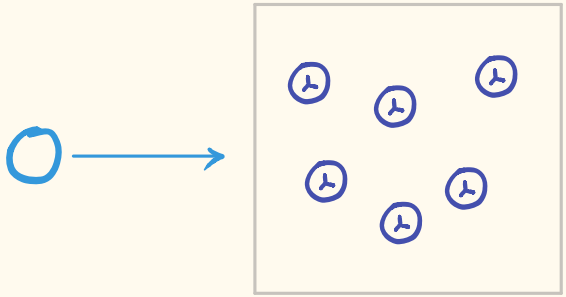
Consensus

Traditional distributed systems (trust)

- Paxos
- raft

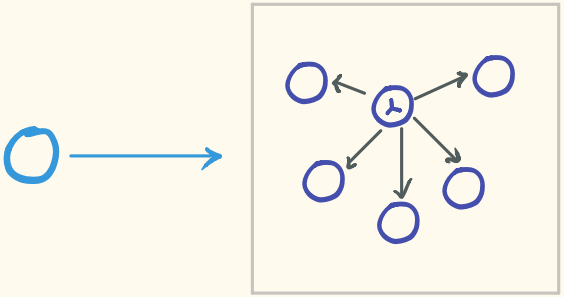
Blockchain Tech (untrust)

- PoW (Proof of Work)
- PoS (Proof of Stake)
- DPoS (Delegated Proof of Stake)
- PoI (Proof of Importance)
- PoD (Proof of Devotion)
- PBFT (Practical Byzantine Fault Tolerance)
- FPA (Federated Byzantine Agreement)
- Hybrid PoW/PoS
- Proof-of-DDOS
- PoH, PoB, ...



Symmetric consensus

Any server can respond



Asymmetric consensus

A leader is elected to issue command

Paxos made Simple

Phase 1. (a) A proposer selects a proposal number n and sends a *prepare* request with number n to a majority of acceptors.

(b) If an acceptor receives a *prepare* request with number n greater than that of any *prepare* request to which it has already responded, then it responds to the request with a promise not to accept any more proposals numbered less than n and with the highest-numbered proposal (if any) that it has accepted.

Phase 2. (a) If the proposer receives a response to its *prepare* requests (numbered n) from a majority of acceptors, then it sends an *accept* request to each of those acceptors for a proposal numbered n with a value v, where v is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

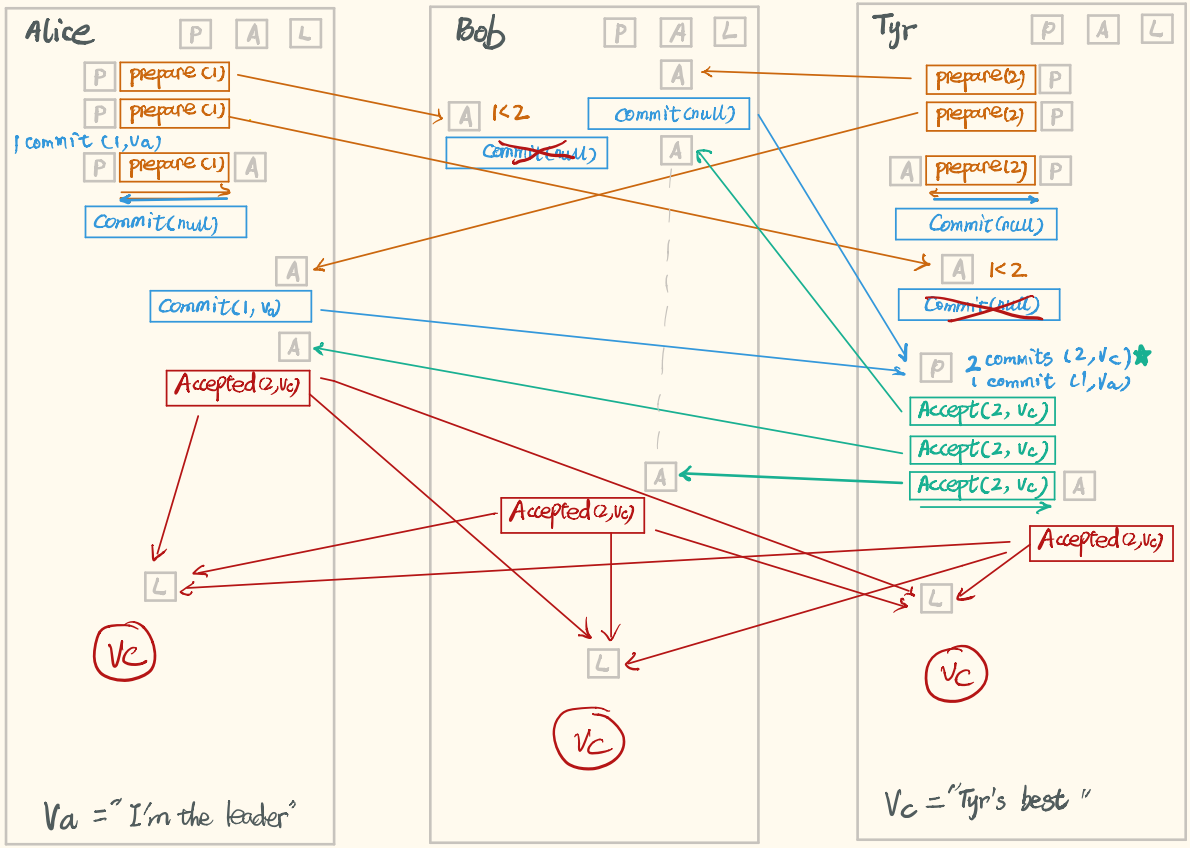
(b) If an acceptor receives an *accept* request for a proposal numbered n, it accepts the proposal unless it has already responded to a *prepare* request having a number greater than n.

To learn that a value has been chosen, a learner must find out that a proposal has been accepted by a majority of acceptors. The obvious algorithm is to have each acceptor, whenever it accepts a proposal, respond to all learners, sending them the proposal. This allows learners to find out about a chosen value as soon as possible, but it requires each acceptor to respond to each learner—a number of responses equal to the product of the number of acceptors and the number of learners.

Proposer

Acceptor

Learner


Alice
P
A
L
prepare (1)
prepare (1)
1 commit (1, Va)
prepare (1)
Commit (null)
Commit (1, Va)
Accepted (2, Vc)
Vc
Va = "I'm the leader"
Bob
1<2
commit (null)
Accepted (2, Vc)
Vc
Tyr
prepare (2)
prepare (2)
prepare (2)
Commit (null)
1<2
2 commits (2, Vc)
1 commit (1, Va)
Accept (2, Vc)
Accept (2, Vc)
Accept (2, Vc)
Accepted (2, Vc)
Vc
Vc = "Tyr's best"

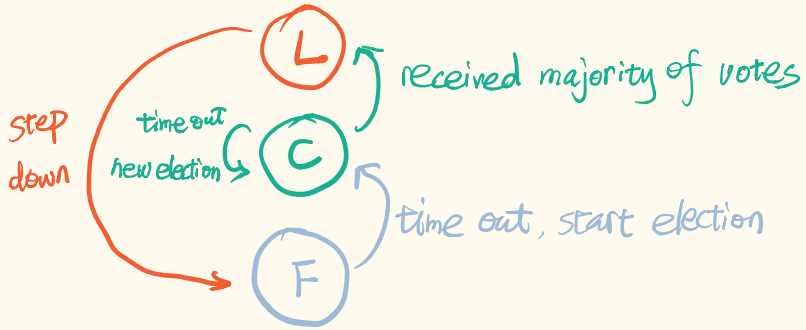
Raft

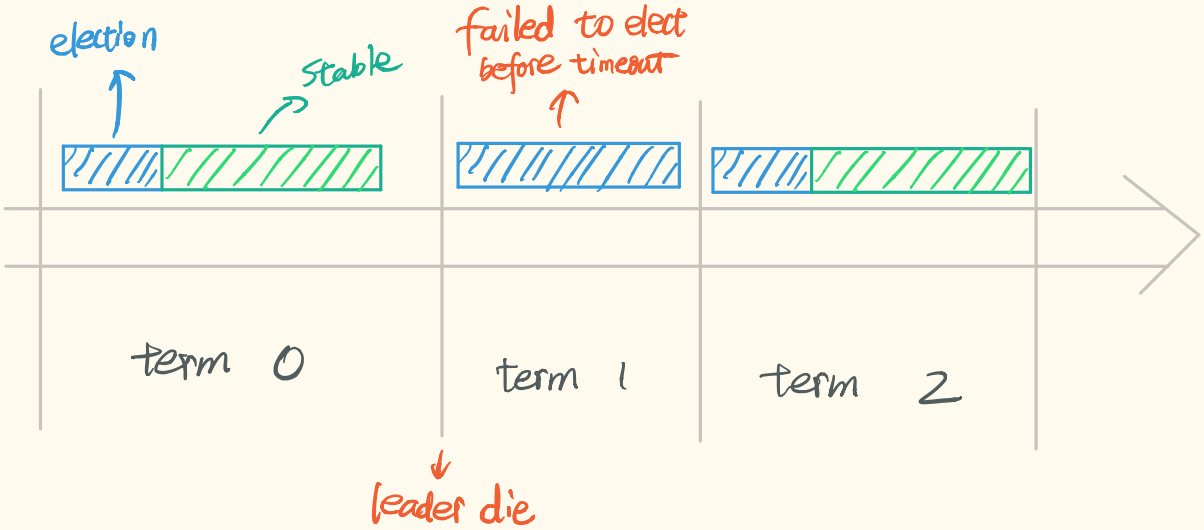
Goal: Have an available replicated state machine.

- available: provide meaningful respone for a log
- replicated: makes other nodes aware of its log
- state machine: follow the log to arrive at the same state

Assymetric

- leader:
- Candidate:
- Follower





stable >> election

Election

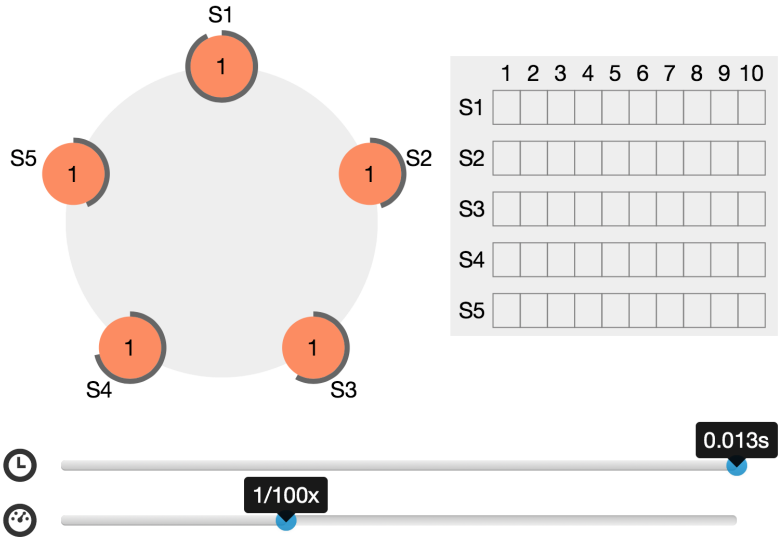
1. Increment current term
2. vote for self
3. request votes

properties
- Safety: at most one winner per term.
- Liveness: Someone eventually wins.

outcome
- I'm leader: receive majority votes
- I'm not: receive valid heartbeat
- no leader: election timeout

Raft Visualization

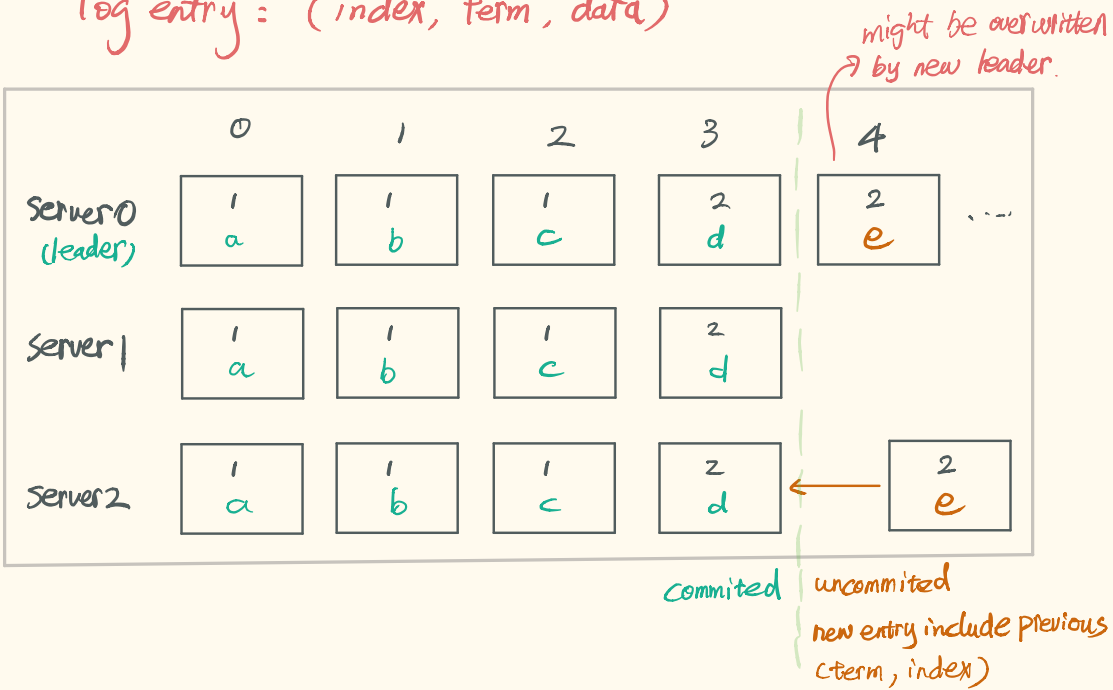
Here's a Raft cluster running in your browser. You can interact with it to see Raft in action. Five servers are shown on the left, and their logs are shown on the right. We hope to create a screencast soon to explain what's going on. This visualization (RaftScope) is still pretty rough around the edges; pull requests would be very welcome.



The Secret Lives of Data is a different visualization of Raft. It's more guided and less interactive, so it may be a gentler starting point.

Log Replication

log entry : (index, term, data)



log replication
- log safety : committed entries in the logs of all future leaders.
- log coherency : deny votes if candidate has "less complete" log
- log integrity : an entry is committed if 1) it's majority stored;
  2) at least one entry from leader's term is also majority-stored

who's using : Cloud Foudary, Kubernetes, etcd, docker swam, Consul

not vote
- if your term is larger
- if your index is better

$(t_v > t_c)$ or $((t_v = t_c)$ and $(i_v > i_c))$

PoW revisit

- what is work { W = Pt = Hash Power × 10 minutes
  something you pay people to do
- what is it soved

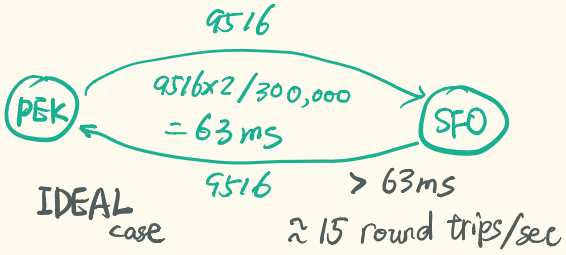
inefficient to get consensus {
- when next block should be produced (global clock)
- who should produce the next block
- make it expensive to produce alt chain
- deciding on best blockchain
- currency distribution } get user onboard without buying it
- incentification ← self reinforce lock in
- side benefit

How does PoS work?

- Is stake (holding a token for a period) a work?
  - It's a power
  - people expect interest

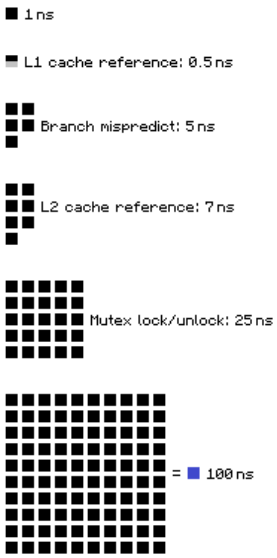
Wrong assumption (in distributed systems)

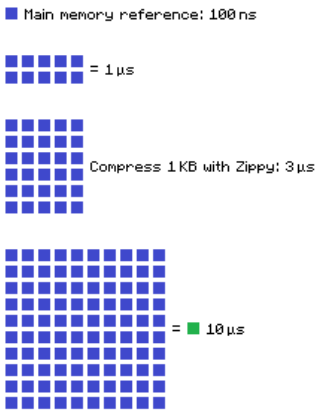
- Network is reliable
  - Can we rely on network to deliver the message?
- Network is homogeneous
  - wired, wireless, cellular, satellite
  - speed varies a lot
- Latency is neglectable
  - L = distance / C

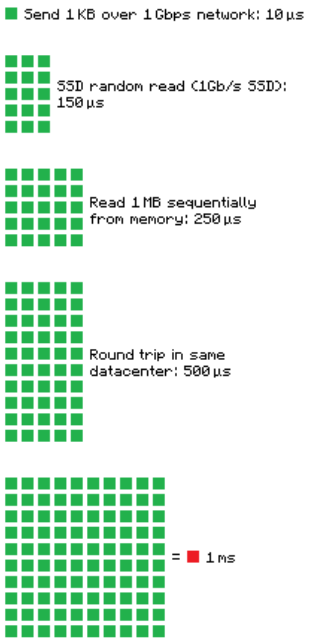


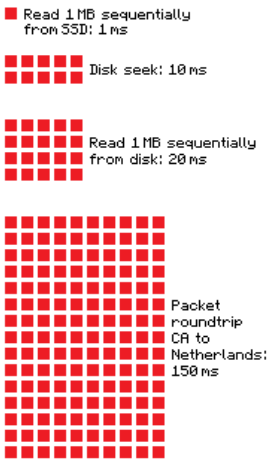
Latency you should know

Latency Numbers Every Programmer Should Know





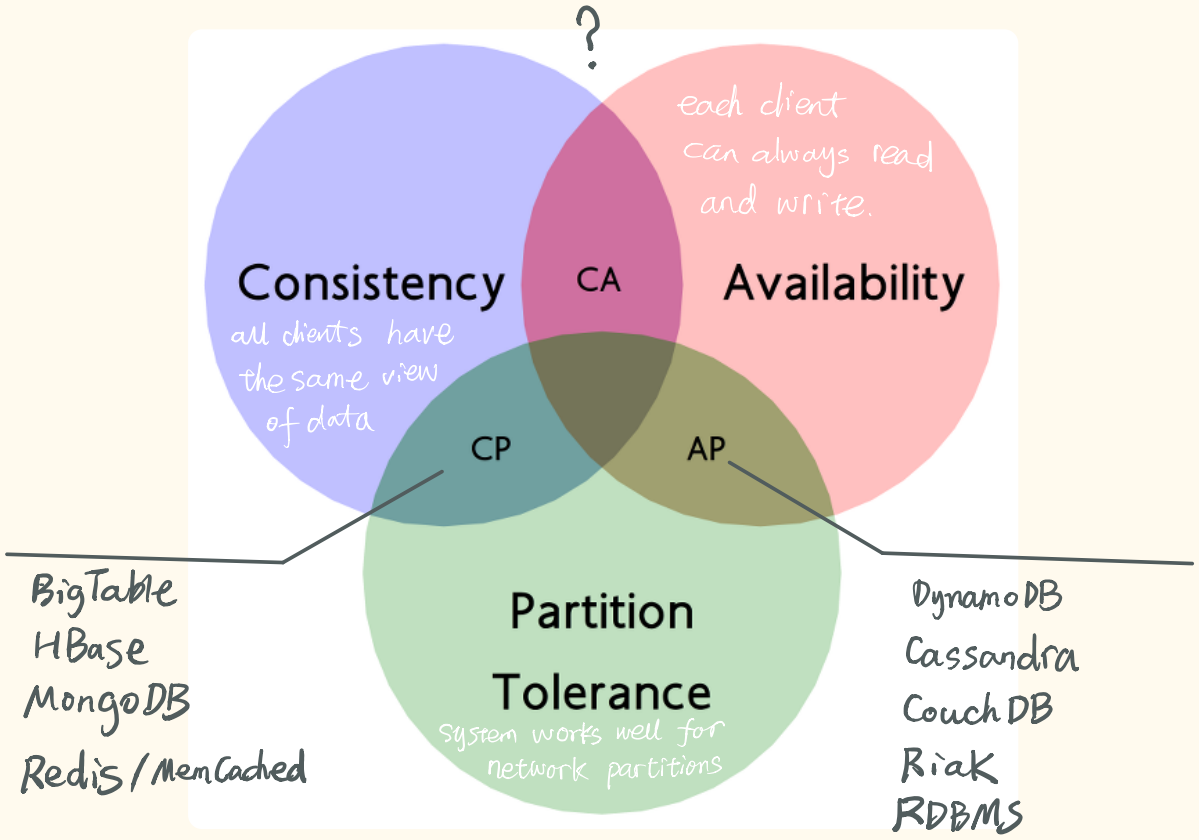




Source: https://gist.github.com/2841832

Wrong assumption (in distributed systems)

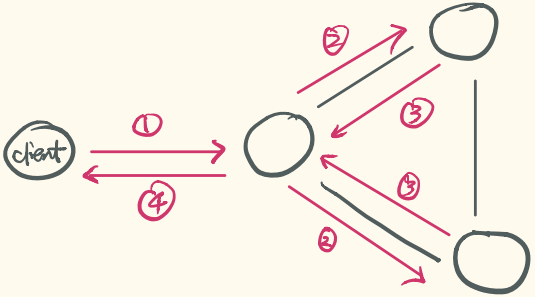
- Bandwidth is not a problem
  - upload speed is pretty limited for home users
- Network is secure
- Topology won't change
  - Node could come and go

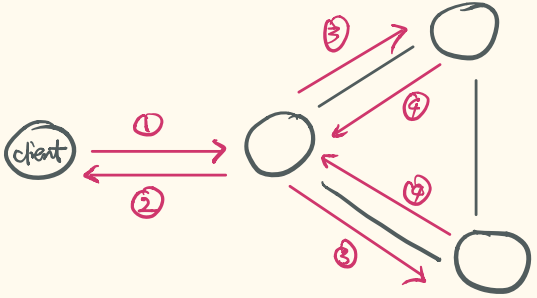

?
each client can always read and write.
Consistency
all clients have the same view of data
CA
Availability
CP
AP
Partition Tolerance
system works well for network partitions
BigTable
HBase
MongoDB
Redis / MemCached
DynamoDB
Cassandra
CouchDB
Riak
RDBMS

Partition and Replication

- Split data to multiple nodes
- Copy data over to ensure consistency
  - sync : consisteney over performance
  - async : performance over consistency

Algorithms
- Gossip
- Consistent hashing





from bitcoin to ethereum: evolution

Alt coins

- Namecoin :

  DNS: tubitv.com → 35.160.85.88

  Namecoin: tyrchen → 1LW79cm5ZB....

  } first-to-file paradigm

  No double spending

- Colored coin:

  Digital token in bitcoin blockchain

- Metacoin: using bitcoin TX to store metacoin TX

  bitcoin: APPLY(S, TX) → S' or ERROR

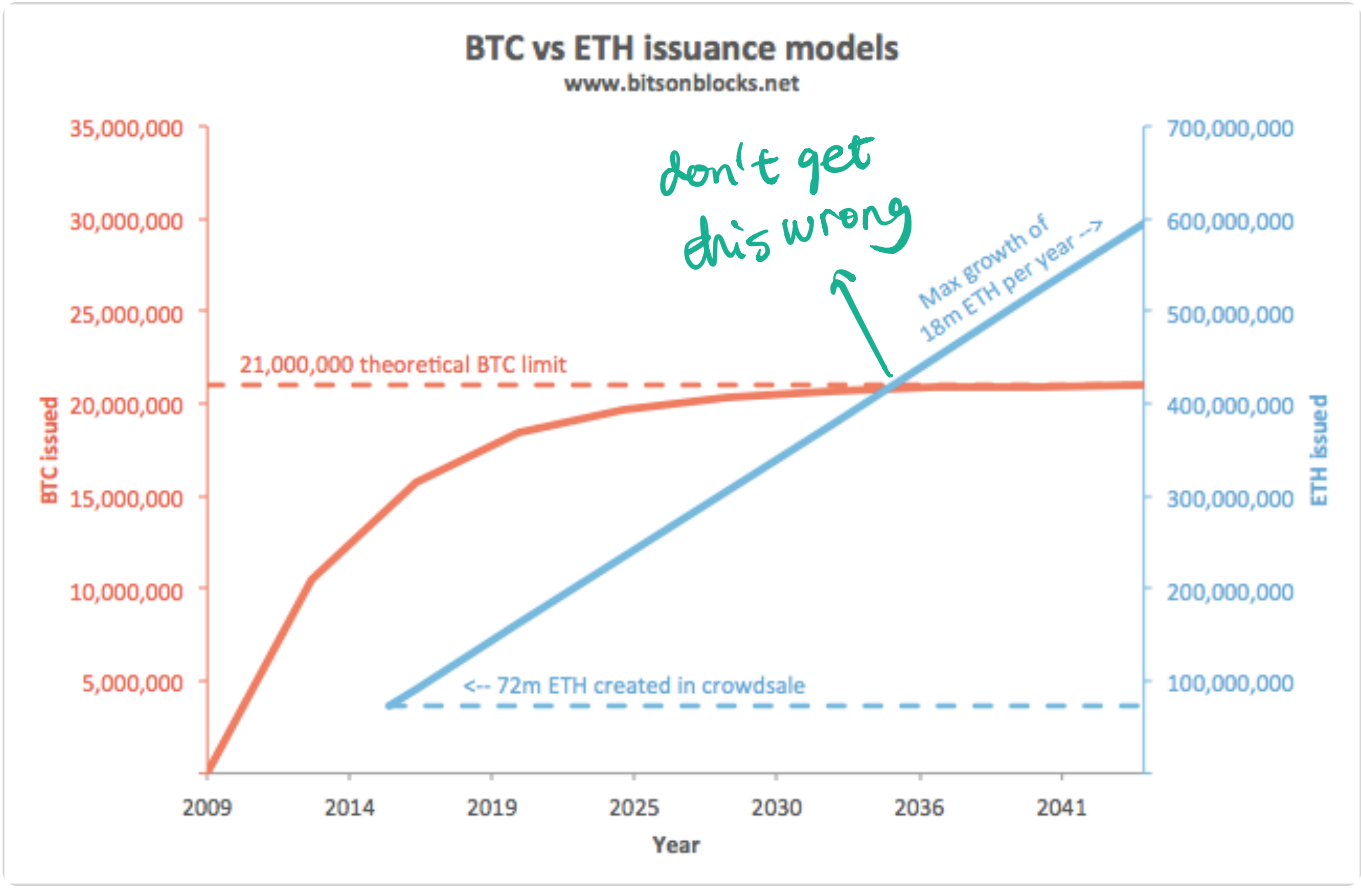
  metacoin: APPLY'(S, TX) { S' / S on ERROR

Problems of bitcoin

- slow
- scripting language is limited
  - Lack of turing completeness (e.g. no loop)
  - Lack of fine-grained control over UTXO
  - Lack of state (e.g. how to do multi-stage contract)
  - UTXO are blind to blockchain data

Comparison

	bitcoin	ethereum
avg block time	10min	12 sec (GHOST protocol)
consensus	PoW	PoW → PoS (casper)
incentive	only the block creator	block creator & uncle
VM	simple & turing incomplete	turing complete (use GAS to solve halt problem)
Scripting	op code	solidity
smart contract	simple	full fledged
total coins	21,000,000	Infinity
account	wallet	externally owned / contract account


BTC vs ETH issuance models
www.bitsonblocks.net
don't get this wrong
Max growth of 18m ETH per year -->
21,000,000 theoretical BTC limit
<-- 72m ETH created in crowdsale
BTC issued
ETH issued
Year
35,000,000
30,000,000
25,000,000
20,000,000
15,000,000
10,000,000
5,000,000
700,000,000
600,000,000
500,000,000
400,000,000
300,000,000
200,000,000
100,000,000
2009
2014
2019
2025
2030
2036
2041

Ethereum Account

- nounce
- account's current ether balance
- contract code, if present
- account storage (empty by default)

UTXO vs balance
||
FP vs Imperative programming

account
- externally owned: controlled by private key
- contract: controlled by code

when a message arrived, code is activated, read/write storage, and send msg/creat new contract

Transaction

- the recipient of the message
- a signature identifying the sender
- amount of ether to transfer
- an optional data field
- STARTGAS / GASPRICE
  - to solve the halting problem

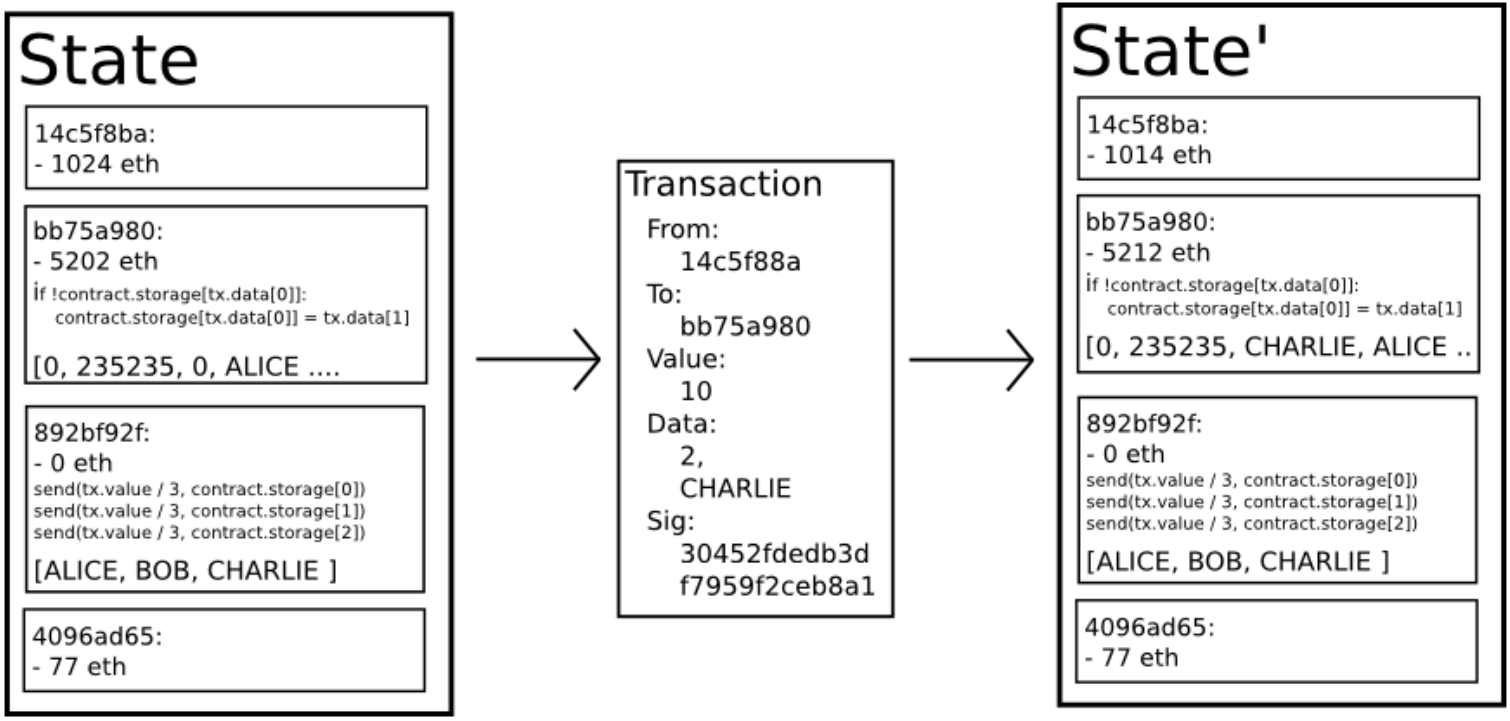
For externally owned account

Message

- sender of the message
- recipient
- amount of ether to transfer alongside the message
- optional data field
- STARTGAS

contract account

Ethereum State Transition

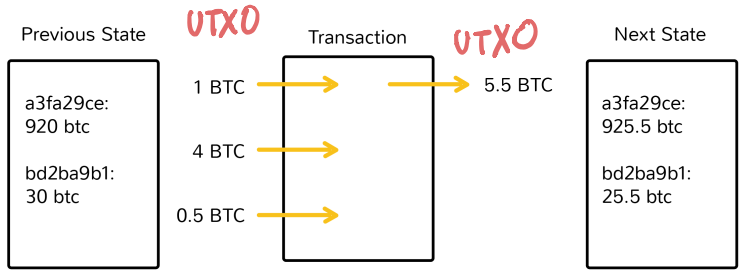


In detail

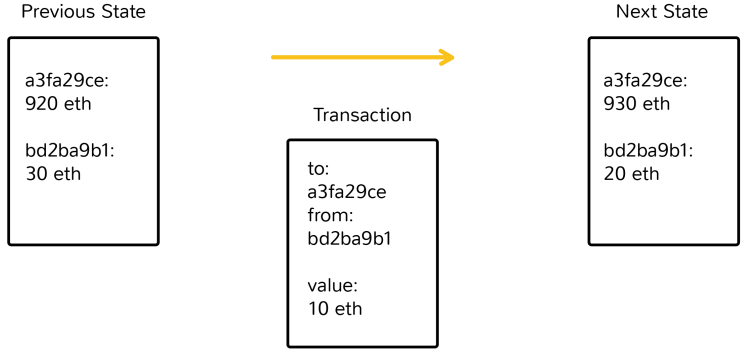
- Sanity check
- Cost: STARTGAS x GASPRICE
- GAS = STARTGAS. reduce GAS based on TX bytes
- transfer ether. If recipient not in state, create it
  if recipient is a contract account, run contract code
- rollback if GAS is used up.
- TX finished: return unused GAS back

State Transition Comparison

Bitcoin



Ethereum

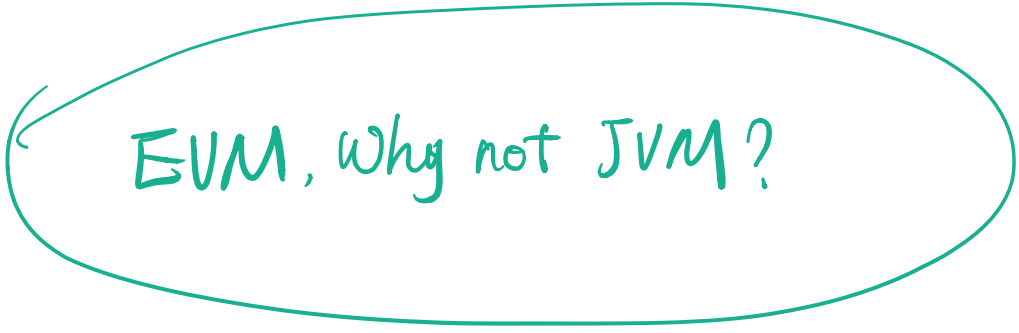


EVM

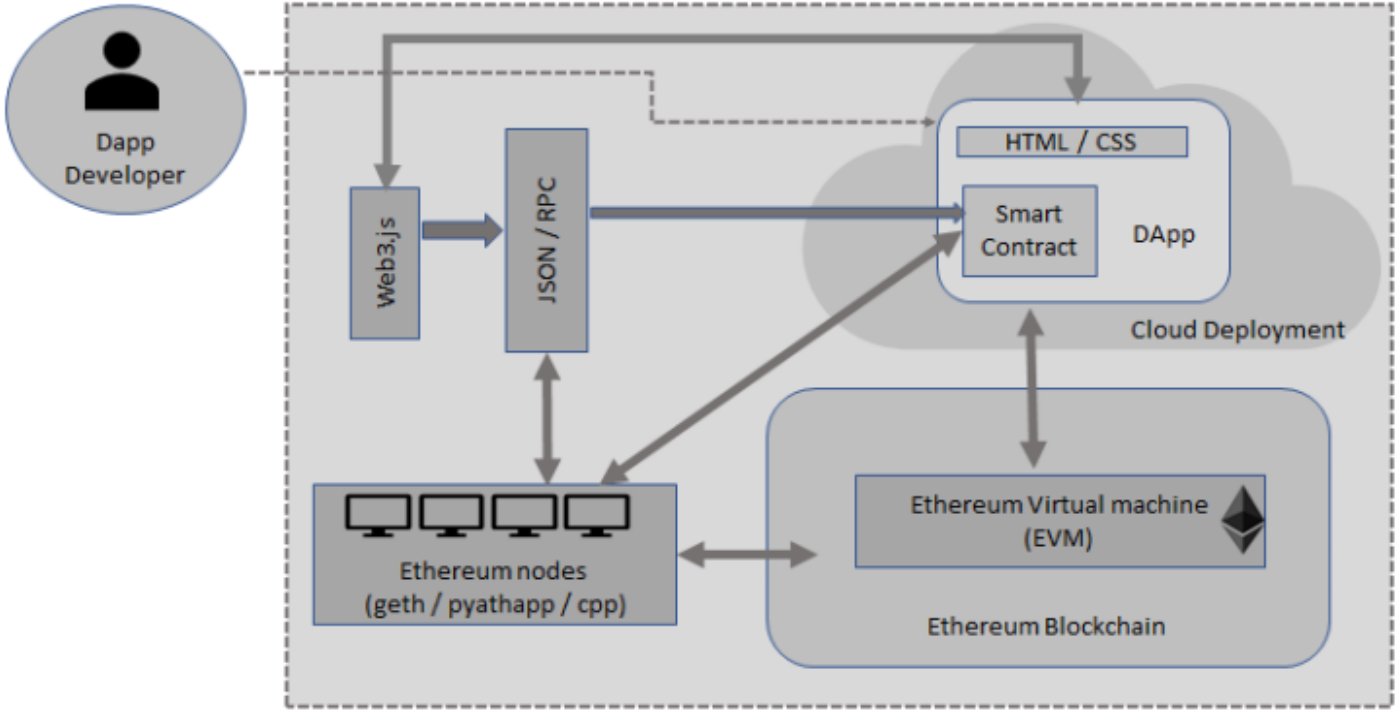
- stack based
- Turing complete (e.g. introduced loop)
- consumes GAS
- To store data:
  - stack
  - Memory (byte array) } volatile
  - Long-term storage (K/V store) – persistant
- full computation state:

  (block_state, tx, message, code, memory, stack, PC, gas)

deterministic is imporeant!


EVM, Why not JVM?

Ethereum Ecosystem



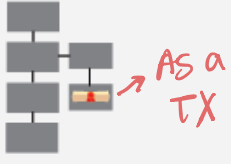
Smart Contract

Vending Machines

A Primitive Smart Contract

Smart Contract

Option contract written as code into a blockchain.



Contract is part of the public blockchain.

Parties involved in the contract are anonymous.

Contract executes itself when the conditions are met.

Regulators use blockchain to keep an eye on contracts.

Solidity

- Object oriented language
- Compiled to EVM
- Can send and receive ether
- Can be run on production, testnet & private net
- Every node can run the byte code

Solidity

```
contract Mortal {
    address owner;

    function Mortal() { owner = msg.sender; }

    function kill() {
      if (msg.sender == owner) selfdestruct(owner);
    }
}

contract Greeter is Mortal {
    string greeting;

    function Greeter(string _greeting) public {
        greeting = _greeting;
    }

    function greet() constant returns (string) {
        return greeting;
    }
}
```

```
greeter_abi:
[{"constant":false,"inputs":[],"name":"kill","outputs":[],"payable":false,"type":"function"},{"constant":true,"in
puts
":[],"name":"greet","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"inputs":[{"name"
:"_g
reeting","type":"string"}],"payable":false,"type":"constructor"}]

mortal_abi:
[{"constant":false,"inputs":[],"name":"kill","outputs":[],"payable":false,"type":"function"},{"inputs":[],"payabl
e":f
alse,"type":"constructor"}]


greeter_code:
60606040523461000057604051602c13803806102c1833981016040528051015b5b60008054600160a060020a0319163360160a060020a0
3161790555b806001900051906020019082805460018160011615610100020316600290049060005260206002090601f016020900481019
82601f10610009057805160ff1916838001178555610b6565b8280016001018555821561b06579182015b8281111561b06578251825591
02001919060010190610090b565b5b506100d79291505b8082111561b06a03576000815560010161b06b565b5090565b50505b505b6101d68061
00eb6000396000f300606060405263ffffffff60e060020a60003504166341c0e1b5811461002f578063cfae32171461003e575b610000565
b346100005761003c6100cb565b005b346100005761004b61010d565b60408051602080825283518183015283519192839290830191850190
8083838215610091575b8051825602083111561009157601f1990920191602091820191016100715656b50505090509810190601f1680156
100bd5780820380516001836020036101000a031916815260200191505b509250505060405180910390f35b60005433733ffffffffffffffff
ffffffffffffffffffffffffffffffff90811691161415610010a5760005473ffffffffffffffffffffffffffffffffffffffff16ff5b5b565b6040805
160208082018352600082526001805484516002828416156101000260001901909216919010404601f8101849004840282018401909552848152
92939091830182828015610191f5780601f1061017457610100808354040283529160200191610191f565b820191906000526020600020905
b81548152906001019060200180831161018257829003601f168201915b5050505050905b905600a165627a7a72305820293955f201e154
5746c248227c00553ddded3cab3195c1f640197fc52fb562600029

mortal_code:
60606040523461000057604051602c13803806102c1833981016040528051015b5b60008054600160a060020a0319163360160a060020a0
3161790555b5b609b806100356000396000f3006060604052363fffffff60e060020a60003504166341c0e1b5811460225575b6000565b346000
0057602c602e565b005b6000543373ffffffffffffffffffffffffffffffffffffffff16ff5b5b5600a165627a7a72305820457c5dba1b6a3f4cf5a0382433ee7f2229eafc63890b4aad9b91cc002303f4920029
```

ERC 20 - ICO

a set of rules that an ethereum token has to implement

Functions

ERC20 token has the following method-related functions:

The specific wording of the function is followed by a clarification of what it does, in [brackets]

1. `totalSupply` [Get the total token supply]
2. `balanceOf(address _owner) constant returns (uint256 balance)` [Get the account balance of another account with address *_owner*]
3. `transfer(address _to, uint256 _value) returns (bool success)` [Send *_value* amount of tokens to address *_to*]
4. `transferFrom(address _from, address _to, uint256 _value) returns (bool success)` [Send *_value* amount of tokens from address *_from* to address *_to*]
5. `approve(address _spender, uint256 _value) returns (bool success)` [Allow *_spender* to withdraw from your account, multiple times, up to the *_value* amount. If this function is called again it overwrites the current allowance with *_value*]
6. `allowance(address *_owner*, address *_spender*) constant returns (uint256 remaining)` [Returns the amount which *_spender* is still allowed to withdraw from *_owner*]

Events format:

1. `Transfer(address indexed _from, address indexed _to, uint256 _value)`. [Triggered when tokens are transferred.]
2. `Approval(address indexed _owner, address indexed _spender, uint256 _value)` [Triggered whenever *approve(address _spender, uint256 _value)* is called.]

Problems of ERC20

↳ ERC223 will fix it

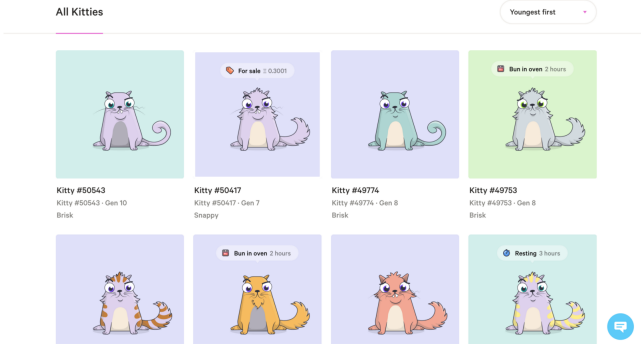
How much ERC20 tokens are currently lost (27 Dec, 2017):

1. QTUM, **$1,204,273** lost. watch on Etherscan
2. EOS, **$1,015,131** lost. watch on Etherscan
3. GNT, **$249,627** lost. watch on Etherscan
4. STORJ, **$217,477** lost. watch on Etherscan
5. Tronix , **$201,232** lost. watch on Etherscan
6. DGD, **$151,826** lost. watch on Etherscan
7. OMG, **$149,941** lost. watch on Etherscan

tokens might get stucked in the balance of contract

ERC 721 - Non-fungible

- unique, indivisible set of property
- cannot be divided
- scarcity: value might be different

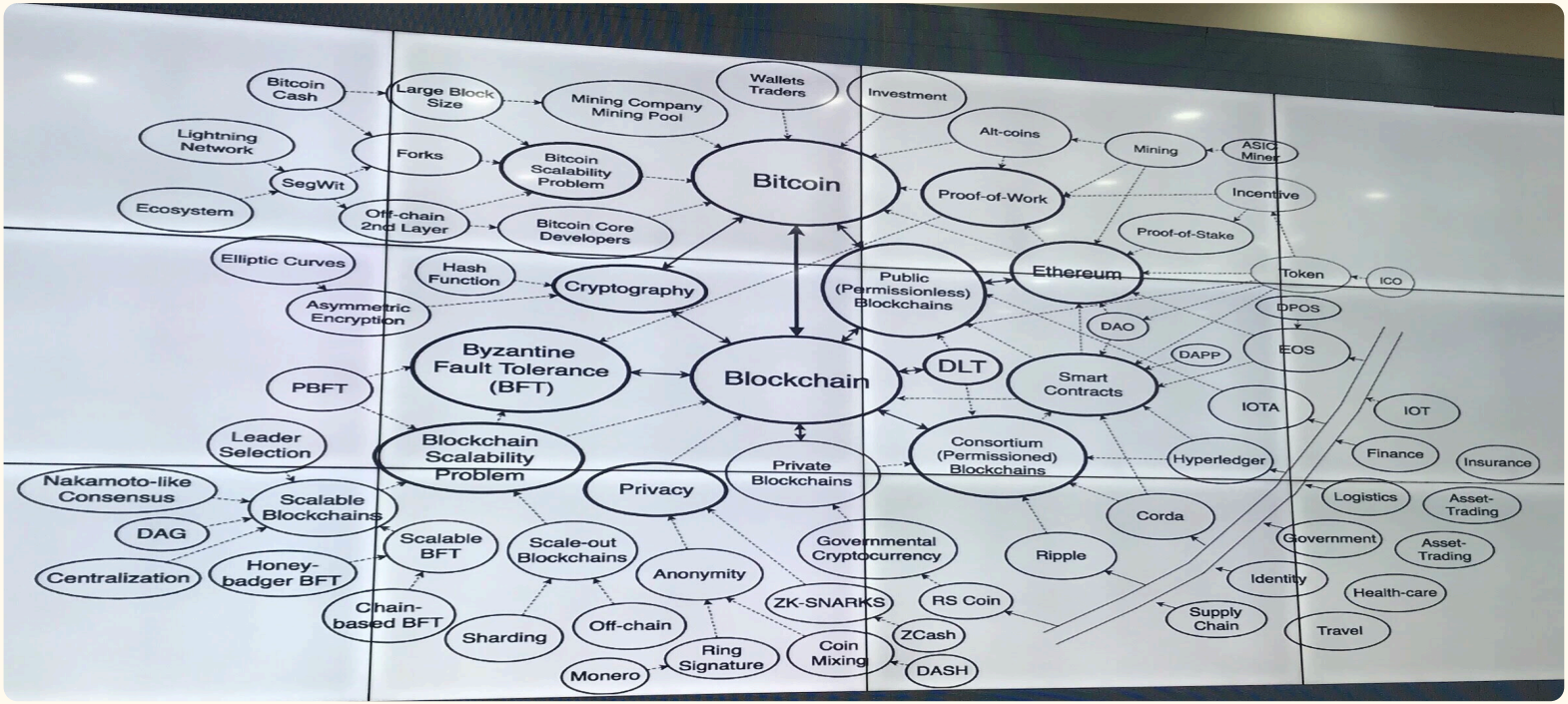


from ethereum to ??: what's next?

Comparison

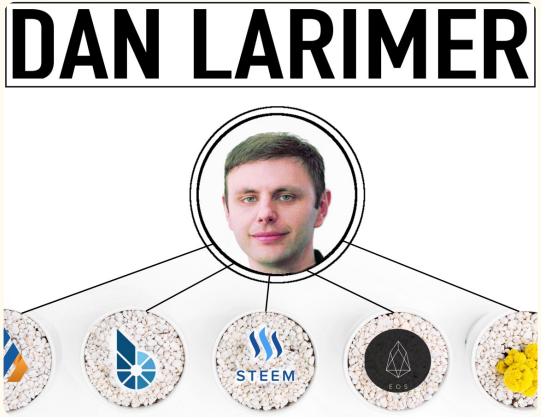
	Timestamp/ consensus	block time	use cases	features
Bitcoin	PoW/sha256	10min	coin transfer, multisig	UTXO, limited supply
ethereum	PoW/ethash	14-15sec	ico, dApps.	smart contract
bitshares	DPoS	3 sec	DEX, referral reward	fast/scalable 100k TX/s
Steemit			social network	rewording by activity, PoB
EOS	DPoS	500ms	OS for running app like steemit	21 super node, webAssembly

ethash: sha-3


Bitcoin Cash
Large Block Size
Mining Company Mining Pool
Wallets Traders
Investment
Lightning Network
Forks
Bitcoin Scalability Problem
Bitcoin
Alt-coins
Mining
ASIC Miner
SegWit
Ecosystem
Off-chain 2nd Layer
Bitcoin Core Developers
Proof-of-Work
Incentive
Proof-of-Stake
Elliptic Curves
Hash Function
Cryptography
Public (Permissionless) Blockchains
Ethereum
Token
ICO
Asymmetric Encryption
DPOS
DAO
Byzantine Fault Tolerance (BFT)
Blockchain
DLT
DAPP
EOS
PBFT
Smart Contracts
IOTA
IOT
Leader Selection
Blockchain Scalability Problem
Consortium (Permissioned) Blockchains
Hyperledger
Finance
Insurance
Nakamoto-like Consensus
Privacy
Private Blockchains
Logistics
Asset-Trading
Scalable Blockchains
Corda
DAG
Scalable BFT
Scale-out Blockchains
Governmental Cryptocurrency
Government
Asset-Trading
Ripple
Centralization
Honey-badger BFT
Anonymity
Identity
Health-care
ZK-SNARKS
RS Coin
Chain-based BFT
Sharding
Off-chain
Supply Chain
Travel
ZCash
Ring Signature
Coin Mixing
Monero
DASH

EOS

- Daniel Larimer : DPoS & graphene tech
- Bitshares : 7/19/2014 1.0
  10/13/2015 2.0 (Graphene)
- Steemit : 7/4/2016
- EOS : 2017 : whitepaper



EOS in a Nut Shell

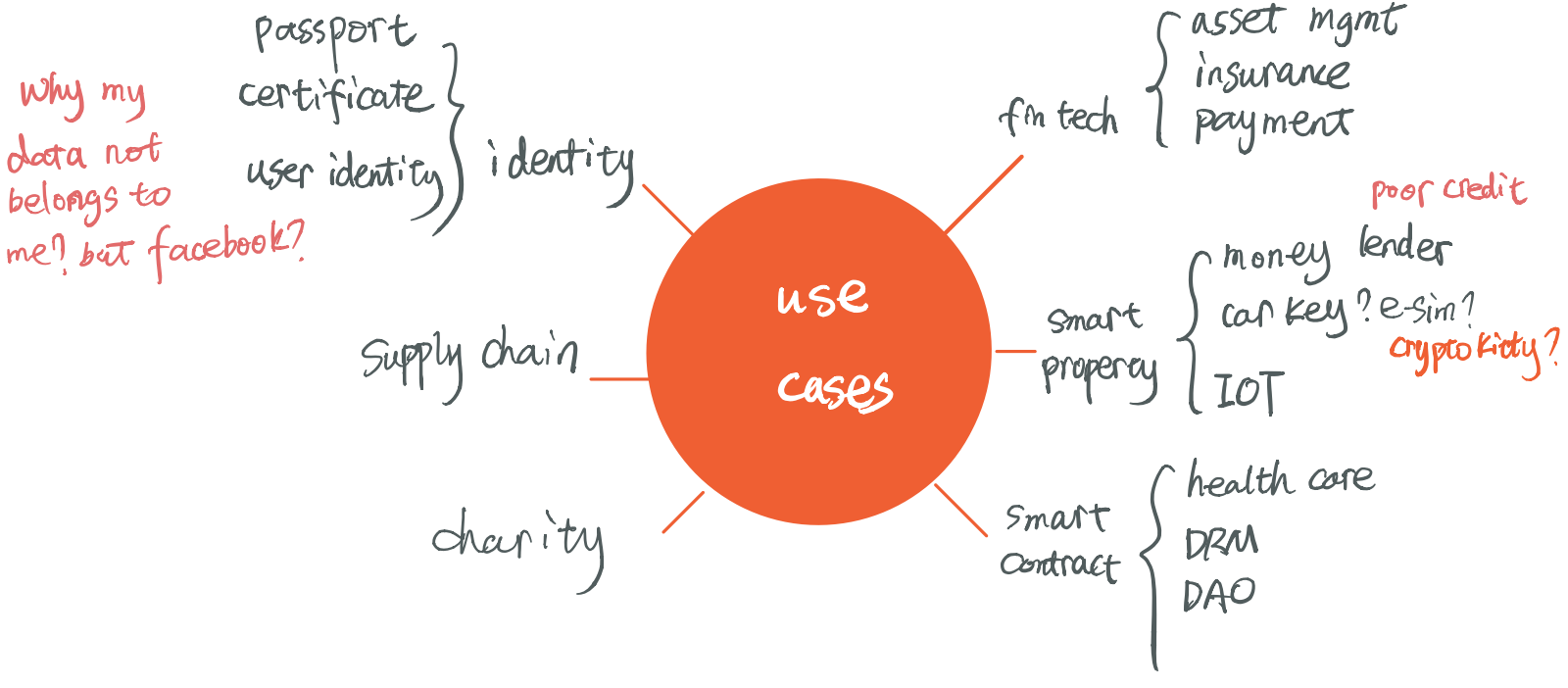
- A blockchain platform that is dev-friendly
- Support millions of users
  - High TX/S, low latency
- Free usage
- Easy to upgrade and bug recovery
- fast sequential performance / parallel performance

Account names	I don't want my users to remember random character strings. Know what a PK is. What a hash or 20 bytes means!	Nice human readable name that my user can unlock with a password
Account recovery	My user just forgot his password. Does he still have his private key saved? He does not seem to know what a private key is... Locked out forever.	You forgot your password? Have a buddy help you get back in.
Account security	My user just posted his private key on twitter (he just wanted to know what he can do it it). All his stuff is gone. Sorry:(	You did what? Good thing you can call your buddy to let you back into your account and cancel that trasaction!
Transaction fees	My user needs to know what gas is and have gas and pay the gas. Facebook was free and now they're gone	I can pay for the network not the user. Just holding stake gives me the resoures I need.
Speed	3 transactions per seccond? That can't be right. Perhaps they'll make it go faster.	Incredibly fast. 20k transactions per second without a problem.Can scale up to more!
Data Storage	Key value store. I guess I could work with that.	Relational database with indexes. Enough said!
Upgradable code	Shoot I have this bug in my code! But code is immutable. I need to transfer everything to a new contract. This is going to be expensive!	Just upgrade my code!

Consensus

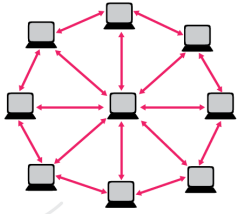
- BFT-DPOS
  - 21 super nodes
  - 500ms per block
  - 250ms for comfirmation after broadcast

applications and innovations


use cases
identity
passport
certificate
user identity
Why my data not belongs to me? but facebook?
Supply chain
charity
fin tech
asset mgmt
insurance
payment
smart property
money
poor credit
lender
car key? e-sim?
crypto kitty?
IOT
smart contract
health care
DRM
DAO

Payment

Someone requests a transaction.



The requested transaction is broadcast to a **P2P network consisting of computers, known as nodes.**

Validation

The network of nodes **validates the transaction and the user's status using known algorithms.**

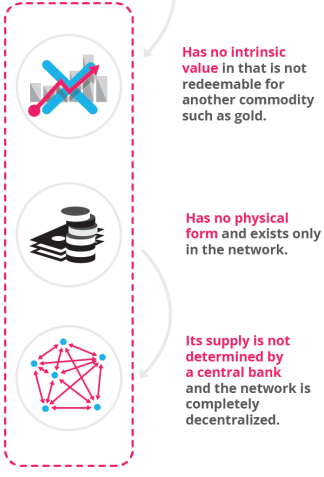
A verified transaction can involve **cryptocurrency,** contracts, records, or other information.

cryptocurrency

Once verified, the transaction is combined with other transactions **to create a new block of data for the ledger,**

The new block is then added to the **existing blockchain,** in a way that is permanent and unalterable.

The transaction is complete.

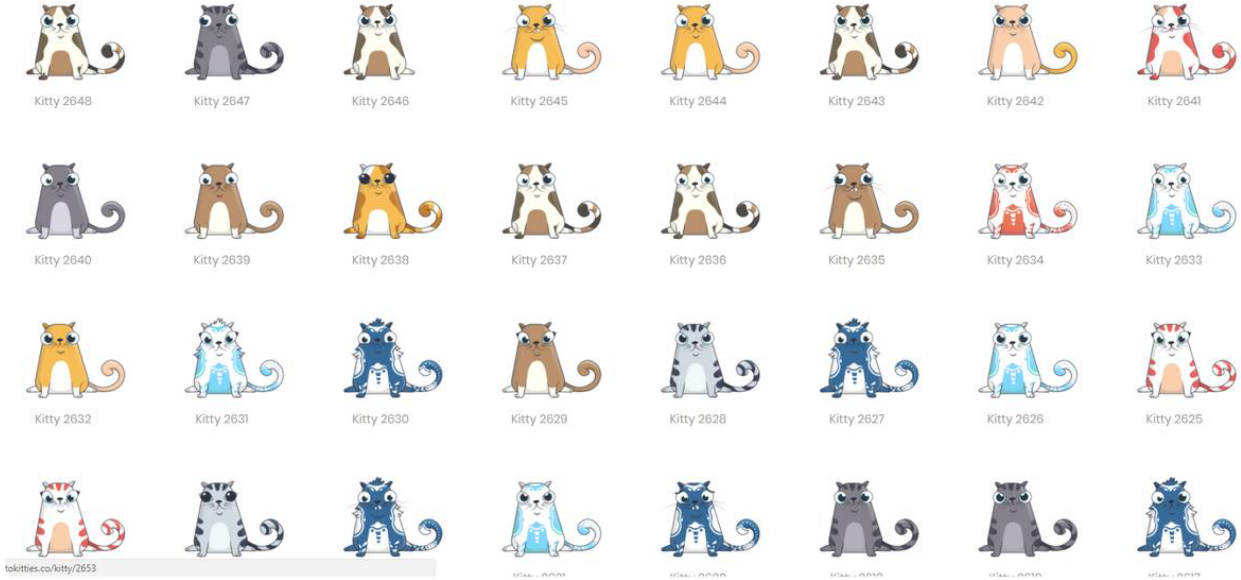


Has no intrinsic value in that is not redeemable for another commodity such as gold.

Has no physical form and exists only in the network.

Its supply is not determined by a central bank and the network is completely decentralized.

Crypto Kitties



How?

- How the team make money?
  - auction (Gen 0, 50,000)
  - TX tax (3.75%)
- How use make money?
  - buy - hold - sell
  - multiply

How to multiply?

- each kitty has her own generation
  - Team own · Gen 0
- Gen X + Gen Y ⇒ max(x, y) + 1
- no gender for simplicity
- cooldown time
  - 0-1, 1min, 2-3, 2min, 4-5, 5min
  - the more you multiply, the snappy you will be

Genetic

Kitty #1

Owner: 0x79bD592415fF6c91CFe69A7f9cD091354fc65a18
Born: November 23, 2017 at 6:19 AM UTC
Generation: 0
Block number: 0x4645a2
Block hash: 0x62b5de48e43c2ff66623d272f9dd1db879870f9d78c840b450501b9e4fbe93ab
Official profile: CryptoKitty #1
Cattributes: Genesis

Genes

Use the explorer below to browse this kitty's genes in different formats. The most useful format is named after @kaigani, the CryptoKitty scientist who mapped the genome.

kai hex binary

ccac 7787 fa7f afaa 1646 7755 f9ee 4444 6766 7366 cccc eede

Sale history

Sold for 246.926 ETH	Dec. 2, 2017, 8:32 PM UTC

Parents

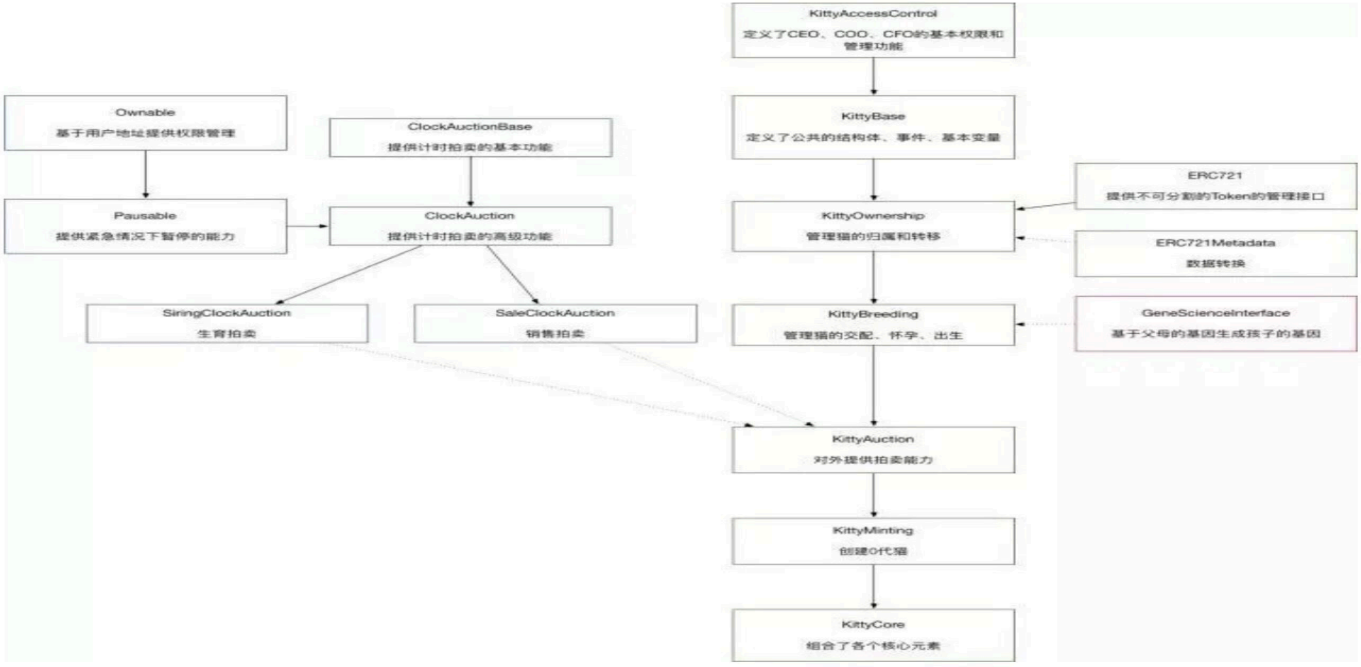
This kitty has no parents!

Bebehs

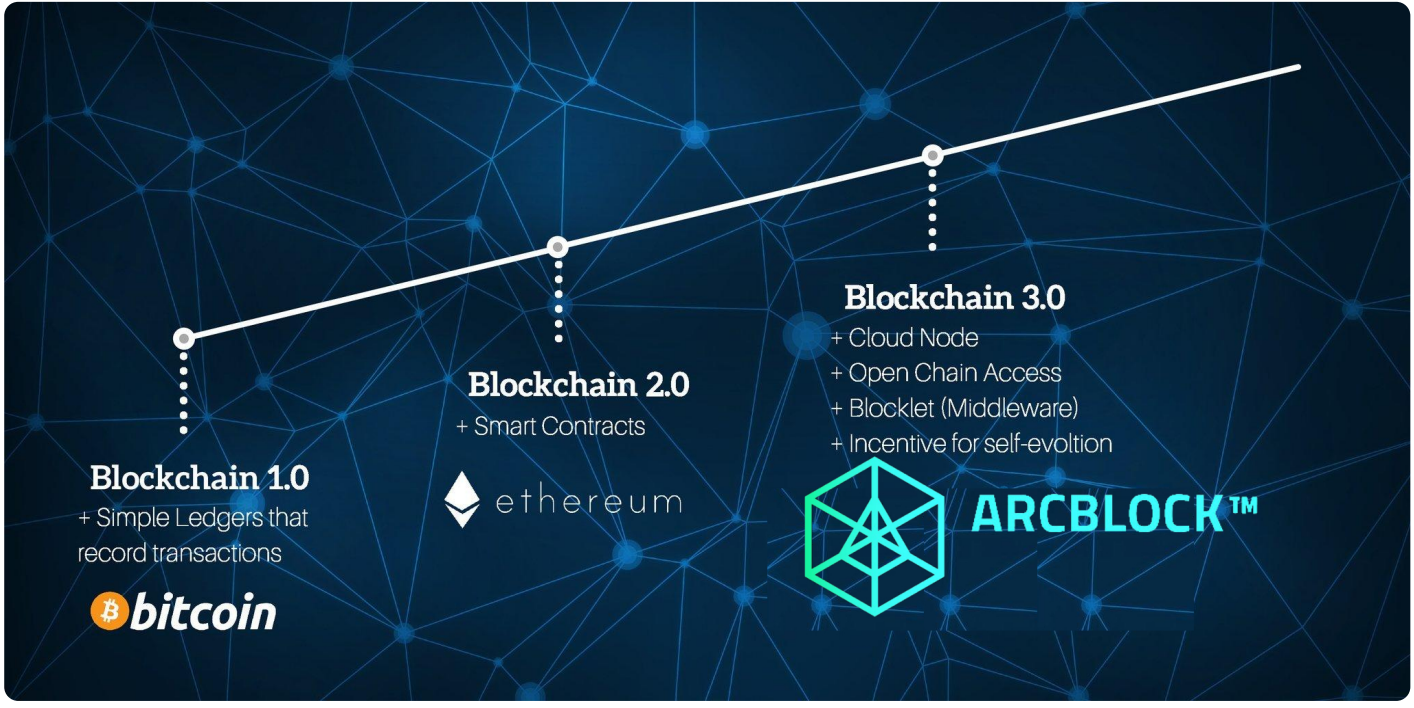
This kitty has 0 bebehs

Gene Code

- 256 bit - 16 = 240 bit
  - every 5 bit Kai encoding to a group
  - every 4 group to a block
  - 12 blocks
- 12 blocks
  - mouth, wild, color, pattern color
  - body color, eye type, eye color, pattern, body
  - 3 reserved


KittyAccessControl
定义了CEO、COO、CFO的基本权限和管理功能
Ownable
基于用户地址提供权限管理
ClockAuctionBase
提供计时拍卖的基本功能
KittyBase
定义了公共的结构体、事件、基本变量
ERC721
提供不可分割的Token的管理接口
Pausable
提供紧急情况下暂停的能力
ClockAuction
提供计时拍卖的高级功能
KittyOwnership
管理猫的归属和转移
ERC721Metadata
数据转换
SiringClockAuction
生育拍卖
SaleClockAuction
销售拍卖
KittyBreeding
管理猫的交配、怀孕、出生
GeneScienceInterface
基于父母的基因生成孩子的基因
KittyAuction
对外提供拍卖能力
KittyMinting
创建0代猫
KittyCore
组合了各个核心元素

Arc block


Blockchain 1.0
+ Simple Ledgers that record transactions
bitcoin
Blockchain 2.0
+ Smart Contracts
ethereum
Blockchain 3.0
+ Cloud Node
+ Open Chain Access
+ Blocklet (Middleware)
+ Incentive for self-evoltion
ARCBLOCK™

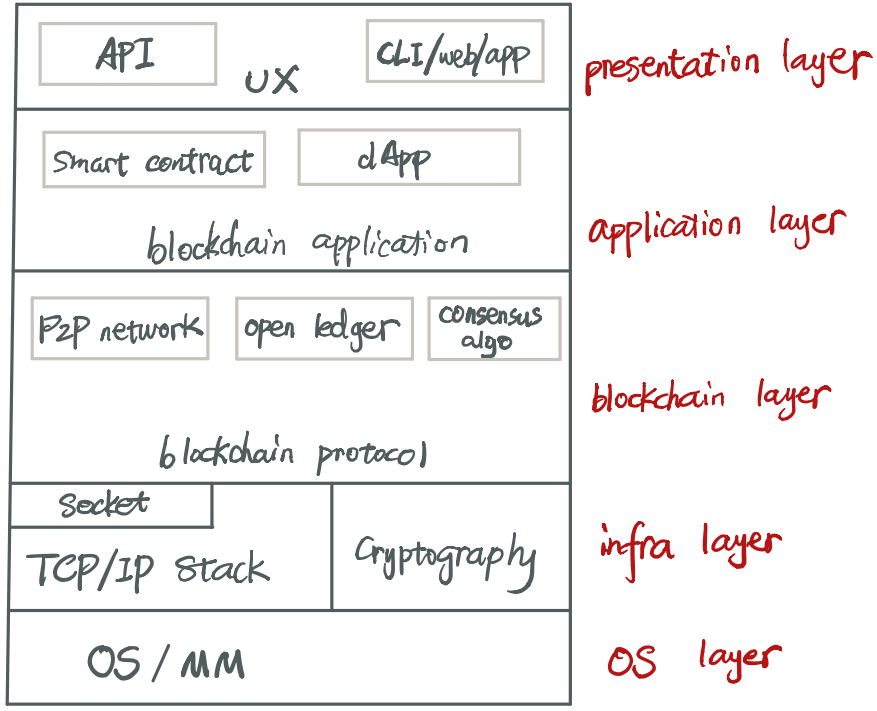
Problem to solve

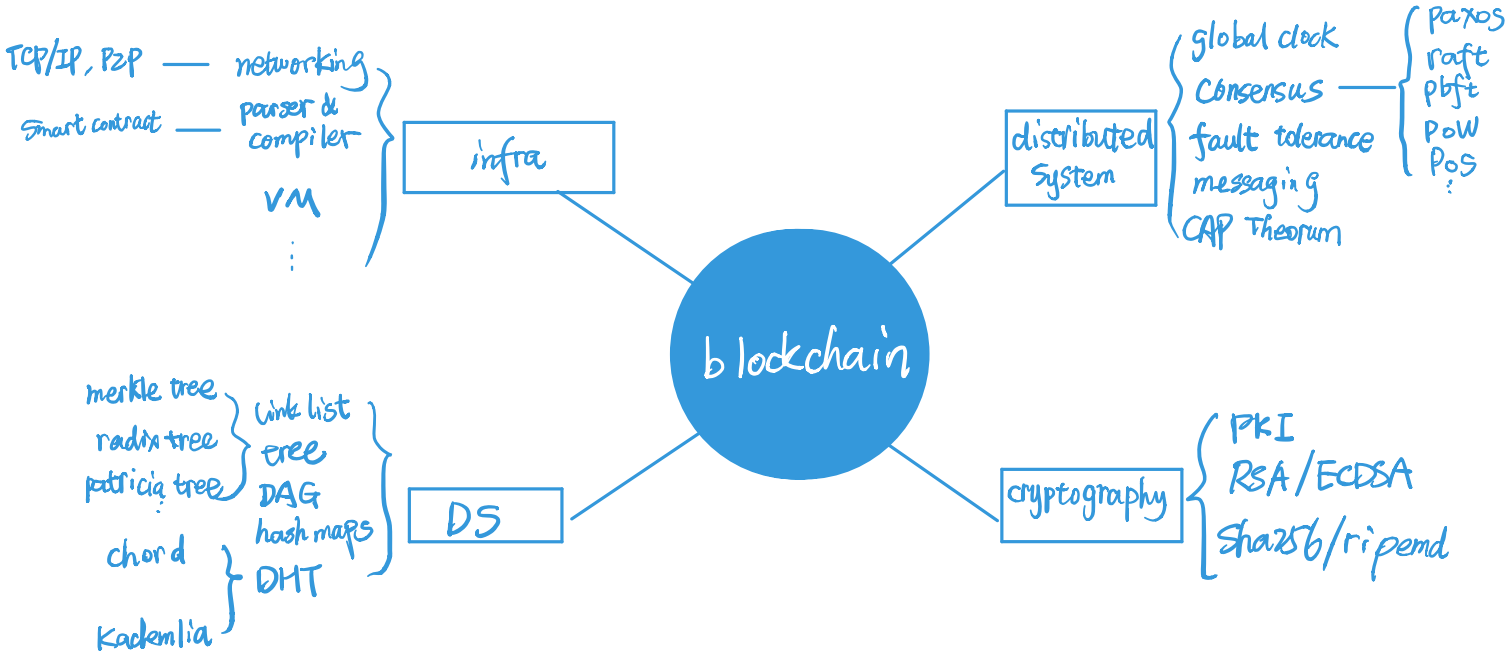
- blockchain tech is not easy for dev to learn
  - How do you learn a new thing?
- existing companies has incentive to embrace blockchain
  - but how?
  - how to make {user credit / badget} to a token?
  - how to add steem like incentive program?
- Too many chains, too many low level detail
- AWS for blockchain?

whitepapers in a nutshell

How to read white paper?

how to learn blockchain tech?


API
UX
CLI/web/app
presentation layer
Smart contract
dApp
blockchain application
application layer
P2P network
open ledger
consensus algo
blockchain layer
blockchain protocol
Socket
TCP/IP stack
Cryptography
infra layer
OS / MM
OS layer


blockchain
infra
networking
TCP/IP, P2P
parser & compiler
Smart contract
VM
distributed system
global clock
consensus
paxos
raft
pbft
PoW
PoS
fault tolerance
messaging
CAP Theorum
DS
link list
merkle tree
radix tree
patricia tree
tree
DAG
hash maps
DHT
chord
Kademlia
cryptography
PKI
RSA/ECDSA
Sha256/ripemd

Find the right resources

- Most of good readings are in English
- follow the famous developers
    - read their blog, medium, etc.
- Book : master bitcoin (I've not read, but everyone recommend it)
- source code
- get your hands dirty

Final words: believe is self-fulfilling prophecy


谢
谢！