

# 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
- from ethereum to ??: what's next?

3:40 - 4:10

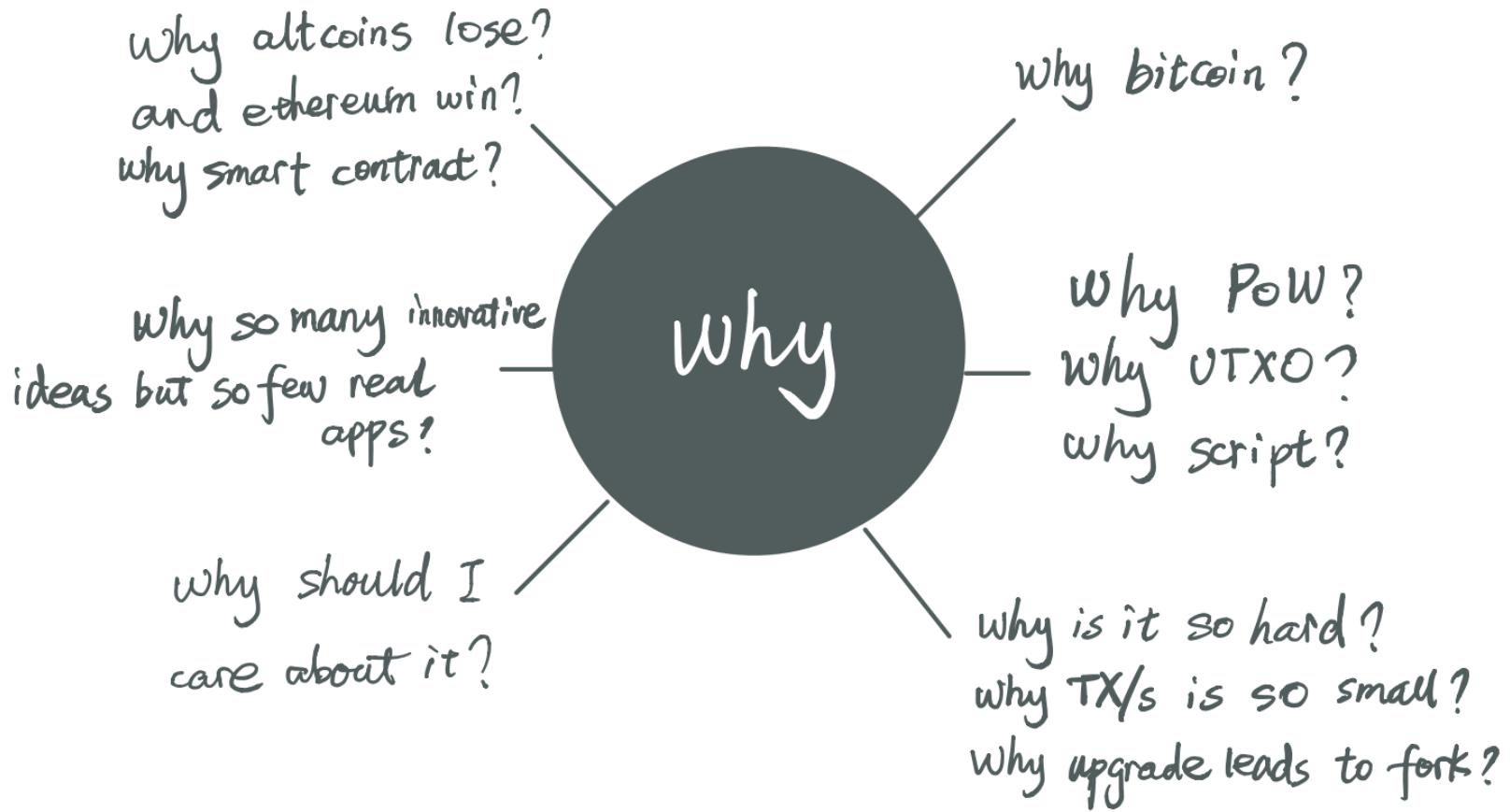


- applications and innovations

4:20 - 4:40



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



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

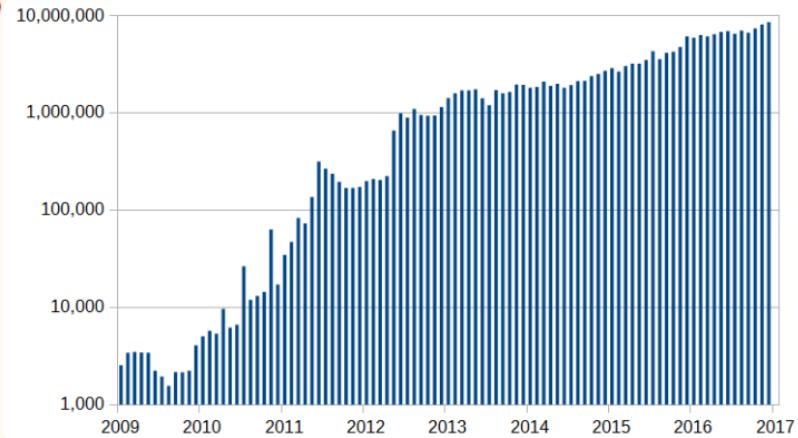
2/2014: Mt. Gox bankruptcy

2015: Coinbase \$75M raised

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

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

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.[17]



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 <satoshi@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 magazine
- 2013 - he wants to add a full-fledged script lang in bitcoin but was denied.
- 1/2014 - ethereum started
- 7/2015 - first PoC (olympic)

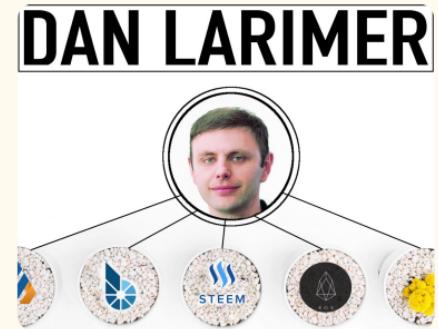
- hard forks {
- 3/14/2016 - first stable (Homestead)
  - 10/16/2017 - improve EVM (Metropolis : vByzantium)
  - TBA - Casper / difficulty time bomb / EIP86 (Metropolis : vConstantinople)

↓  
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/ <u>ethash</u>	14-15 sec	ICO, dApps.	smart contract
bitshares	DPoS	3 sec	PEX, referral reward	fast/scalable 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

How many projects  
are in the network?

**86,034**

projects

**9,375+ projects**  
by companies,  
research institutions,  
and start-ups

**Finding:** Projects of  
organizations are five  
times more likely to  
be *forked* (copied).

How fast is  
it growing?

Averaging  
**8,603**  
per year but with  
**26,885** in 2016

**Finding:** Projects  
developed by  
organizations  
register fastest  
adoption rate:  
20% compound  
annual growth rate.

Project survival?

Only **8%** of  
projects are actively  
maintained

Only **5%** of forked  
projects survive

Projects have  
average life span of  
**1.22 years**

**Finding:** There are  
very few projects with  
high longevity.

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
cminer	Ckoilivas	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
btcld	Btcsuite	Organization	380	1,272	59	An alternative full node Bitcoin implementation written in Go (golang)
testrpc	ethereumjs	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

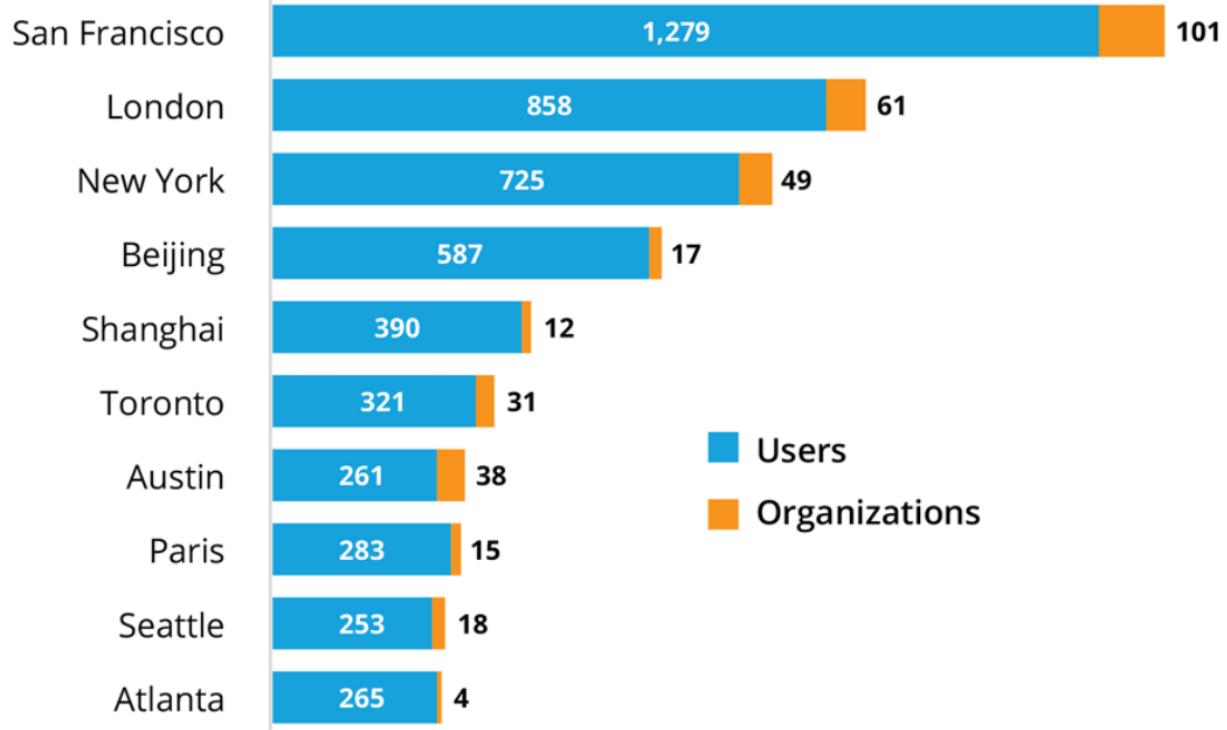
Source: Deloitte analysis of GH Torrent data and GitHub API data.

Top 20 repos are dominated by:

- bitcoin
- ethereum

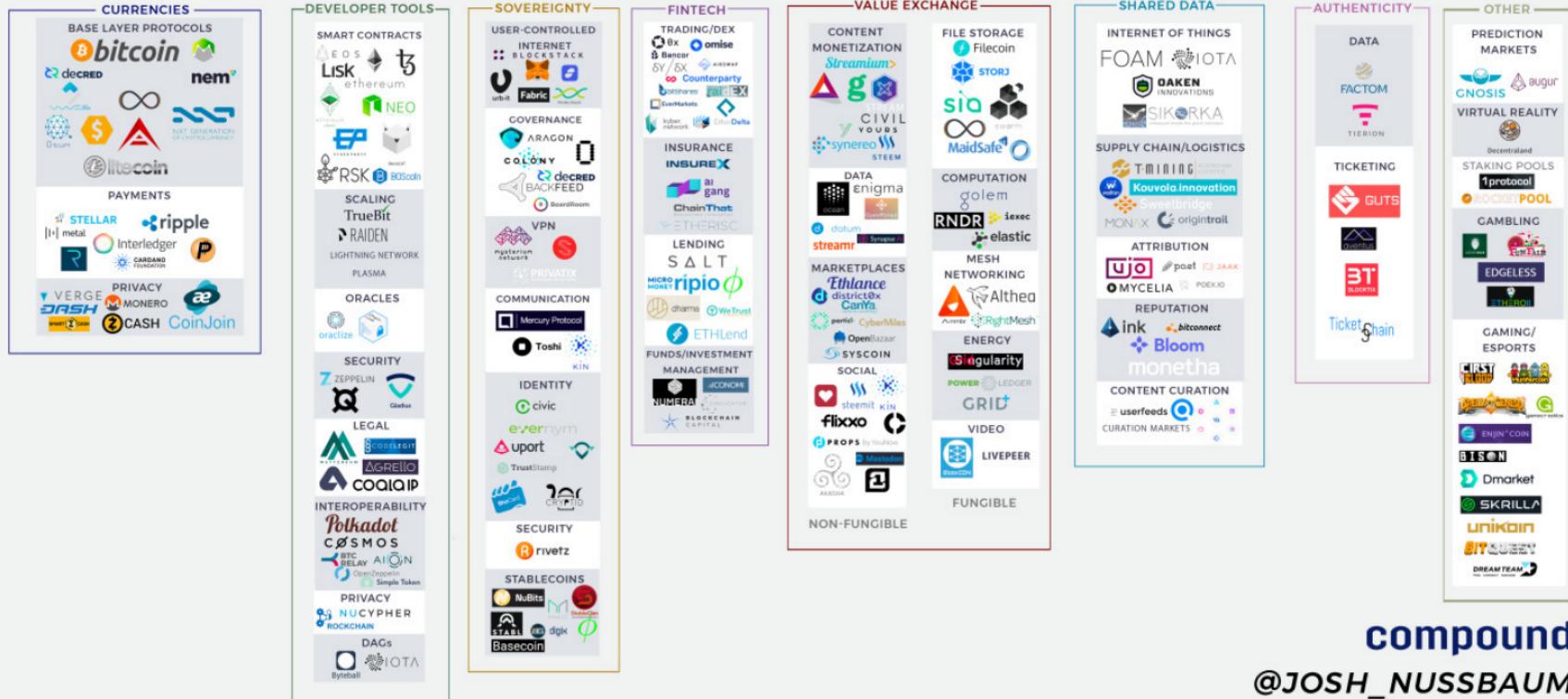
Most of projects are written in:

- C++
- golang



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

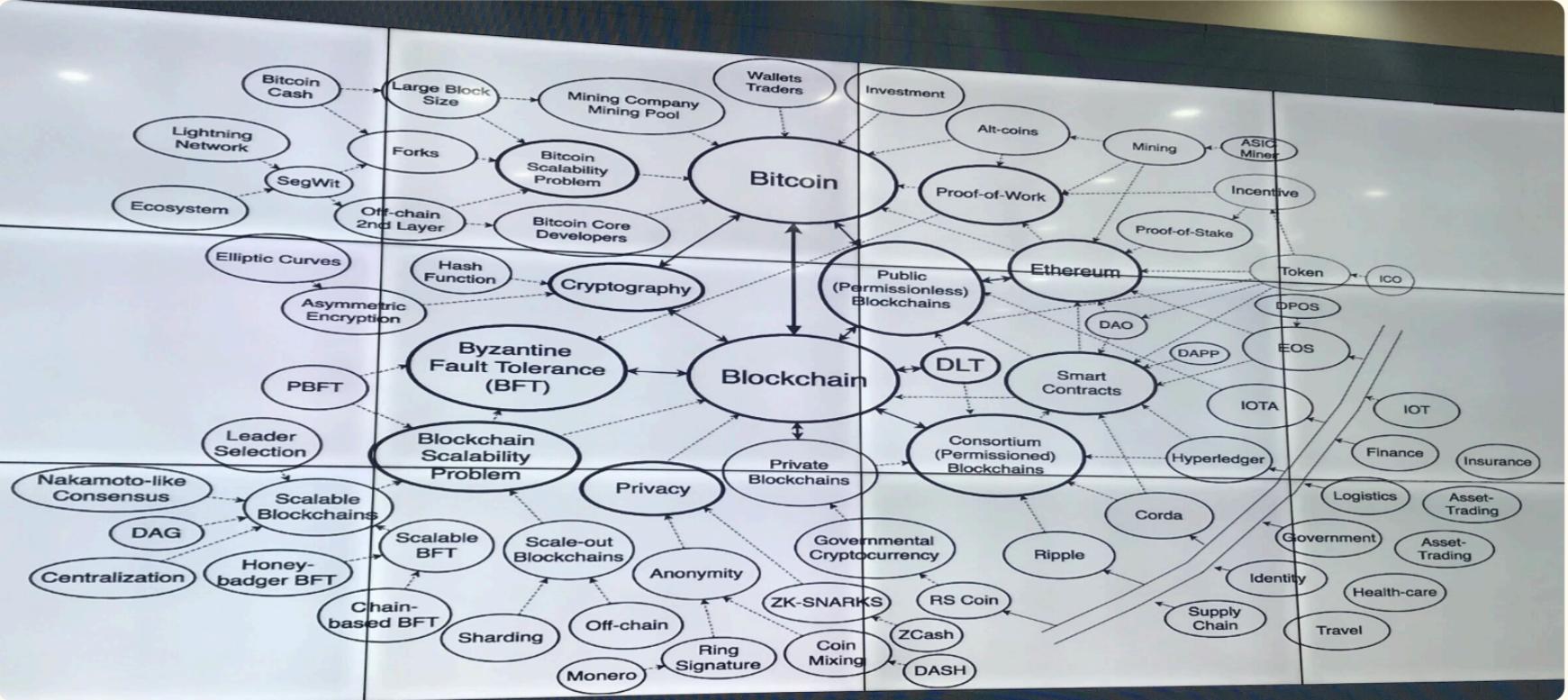
# BLOCKCHAIN PROJECT ECOSYSTEM



compound

@JOSH\_NUSSBAUM

from techcrunch : Mapping the blockchain ecosystem



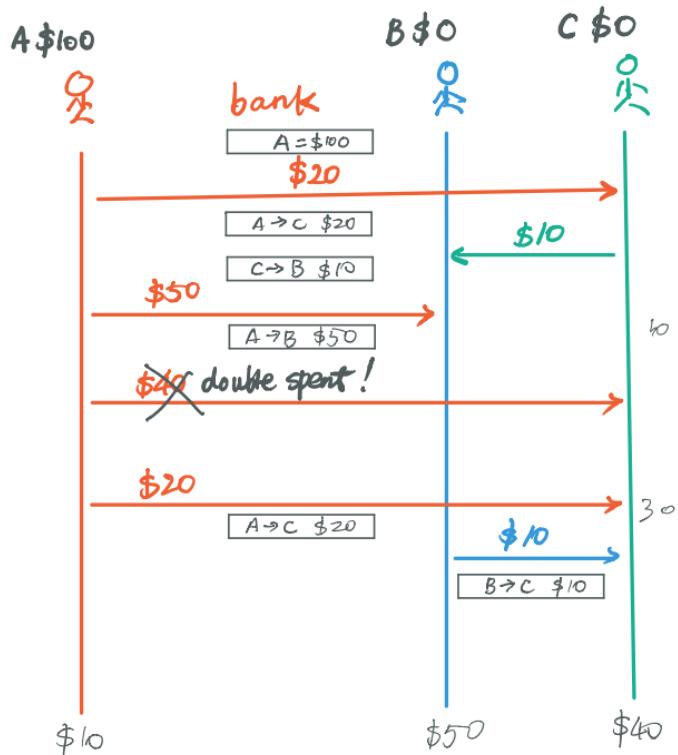
bitcoin: an revolutionary open ledger

# Money transfer



- ① takes too long
- ② fee too high

# Ledger



$A = \$100$
$0$
$A \rightarrow C \$20$
$0$
$C \rightarrow B \$10$
$0$
$A \rightarrow B \$50$
$0$
$A \rightarrow C \$20$
$0$



The history of the transaction is Currency.

Consensus?

Synchronization?

Capacity?

privacy?

Security?

Incentivity?

} Open Ledger

distributed

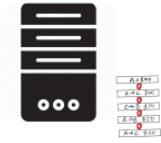
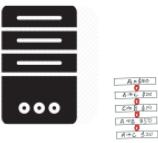
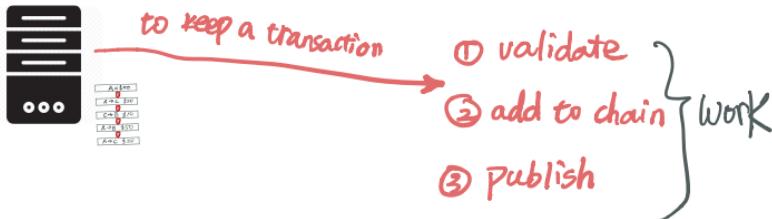
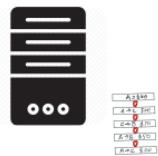
decentralized

not easy to attack

faster (well...)

less cost (hopefully)

# Why a node wants to keep your record?



To compete : - Gamification

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

The reward :

\$50

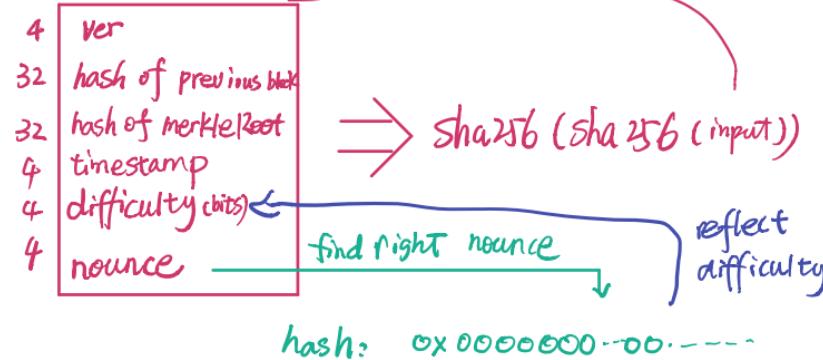
Now 12.5



How would  
you build  
it?

4	bytes
4	magic: 0xd9b4bef9
4	length
4	ver 0x1
32	sha256 (previous block)
32	merkleRoot
4	timestamp : creation time of the block
4	difficulty : 3,007,383,866,427.732
4	nounce: the problem to solve
	transaction count
	transactions

## Game (PoW):



what we learned:

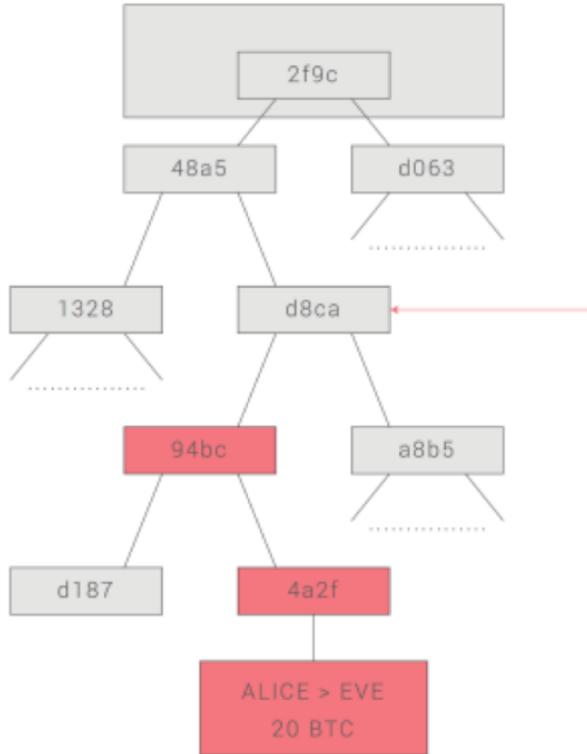
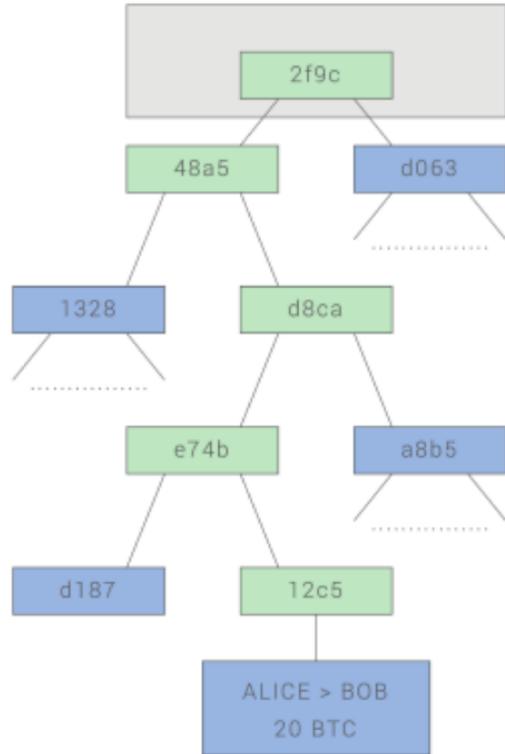
1. to find a "nounce" is hard (random)
2. by tuning difficulty , the system can limit the generation of a block to close to 10min.
3. merkle tree + bloom-filter made wallet applicable
4. chaining + PoW greatly reduced the chance of "do evil thing".
5. "incentive" encourage nodes to be good.

## 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)
- incentivization - 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

## Block #1

BlockHash 00000000839a8e686ab5951d76f411475428afc90947ee320161bbf18eb6048 ⓘ

### 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	0e3e2357e806b6cdbf1f70b54c3a...	Next Block	2
Previous Block	0		

### Transactions

0e3e2357e806b6cdbf1f70b54c3a17b6714ee1f0e68bebb44a74b1ef512098 ⓘ	mined Jan 8, 2009 6:54:25 PM
No Inputs (Newly Generated Coins)	
126050U4Rq3P4ZxziKzriLSLmMBirjrJX	50 BTC (U)
Type pubkeyhash scriptPubKey 0490d538e853519c726a2c9fe61ec11600se1390813a627cd6fb8be7947be63c52de7... ⓘ	
51073 CONFIRMATIONS	50 BTC

# Recent Block

## Block #510266

BlockHash 00000000000000000000000000000004f7aeba4f2ea6027343e045e1b86120093d9cd32ebe7e ⓘ

### Summary

Number Of Transactions	563	Difficulty	3007383866429.732
Height	510266 (Mainchain)	Bits	175397dc
Block Reward	12.5 BTC	Size (bytes)	992316
Timestamp	Feb 21, 2018 8:50:46 AM	Version	536870912
Mined by	BTCC Pool	Nonce	2966277765
Merkle Root	38967c76b631ba5fee39ed849711... ⓘ	Next Block	510267
Previous Block	510265		

### Transactions

0e9e192e04d95a74d9b16abc85725fffd9754bf2c81bbb963995d6e7f867e56 ⓘ	mined Feb 21, 2018 8:50:46 AM
No Inputs (Newly Generated Coins)	
13TEtZNn1PK34HYAu01QjYMsOdJf3qeHx	12.63484184 BTC (U)
Unparsed address [0]	0 BTC (U)
8 CONFIRMATIONS	12.63484184 BTC
0a7e02a56f159388cc037ba38616179cb8b42cc5998d39a28aebd32f9709b ⓘ	mined Feb 21, 2018 8:50:46 AM
179kGoYHaiHgRlmPdmC6JQD43wBFUmfvfT	0.03619554 BTC
12lVTz5r1TpxKwWrWfAxNwGkXznYvsU	0.03619554 BTC (U)
FEE: 0 BTC	8 CONFIRMATIONS
0.03619554 BTC	
02ef404706fa4d7617d918c260a79de69d7d57a6ed4ea63843e6616525ae9707 ⓘ	mined Feb 21, 2018 8:50:46 AM
1H7n0k2j4RwdBuwCKDGwfFmnhNF7B65b9H	29.73215027 BTC
14p1QFZmXGPvNBAlbDmGj4qqeX5gyBywV	29.67034227 BTC (U)
17Dyli2SpuAShdnk254nHimhBAsEfLKx9	0.06 BTC (U)
FEE: 0.001808 BTC	8 CONFIRMATIONS
29.73034227 BTC	

# A TX

## Transaction

Transaction cb671e761a9ede0550af141c01d06b4f1475a8816ae60906f5064c2972d8c86c [\[P\]](#)

## Summary

Size	226 (bytes)
Fee Rate	0.0005899999999999999 BTC per kB
Received Time	Feb 21, 2018 10:03:44 AM
Mined Time	N/A
Included in Block	Unconfirmed

## Details

cb671e761a9ede0550af141c01d06b4f1475a8816ae60906f5064c2972d8c86c [\[P\]](#)

14dKmeMV4HmpbbSbkeeYG524nNuEFP6M5Q 4.43931125 BTC



18La8SWywMxm5ibVpzgWDVYkGwSj1HPPAi 0.03584102 BTC (U)

1KaTbQarFdLTkCUQ5beRaVyZ7UvRxKcnoE 4.40333689 BTC (U)

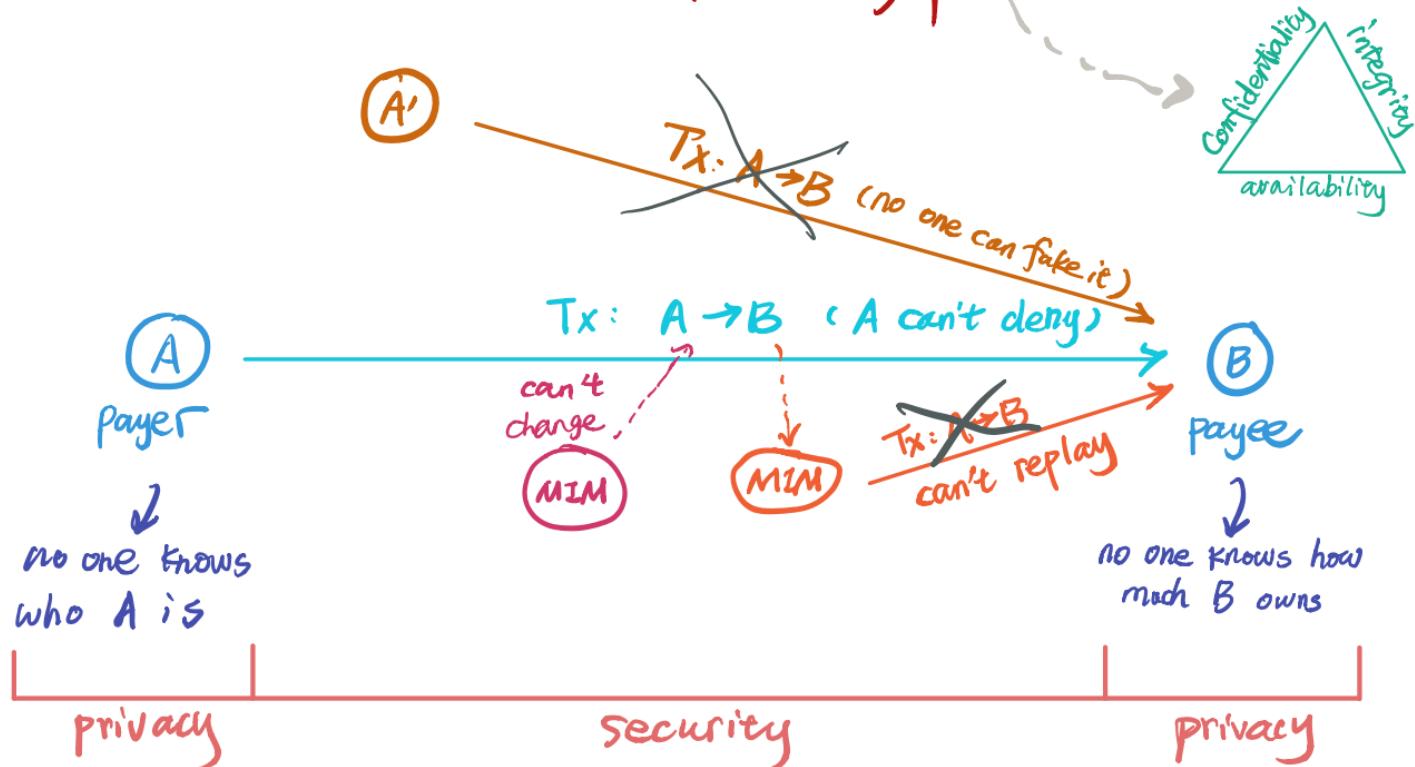
FEE: 0.00013334 BTC

UNCONFIRMED TRANSACTION!

4.43917791 BTC

$$\text{fee} = \text{input(s)} - \text{Output(s)}$$

What about { Security  
Privacy } of OL?

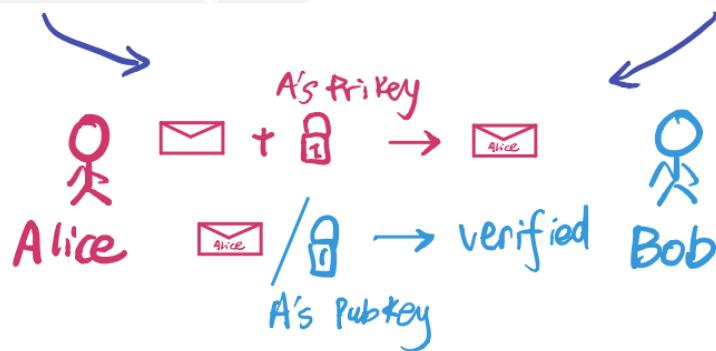


# What do we have?

## 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 length of  $e$ .
  3. Select a **cryptographically secure random integer**  $k$
  4. Calculate the curve point  $(x_1, y_1) = k \times G$
  5. Calculate  $r = x_1 \bmod n$ . If  $r = 0$ , go back to step 3.
  6. Calculate  $s = k^{-1}(z + rd_A) \bmod n$
  7. The signature is the pair  $(r, s)$ .
- $(x_1, y_1) = k \times G$
- 

Secp256k1



- Select two large prime numbers  $p, q$
- Compute **RSA**
$$n = p \times q$$

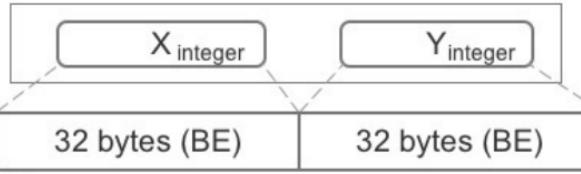
$$\varphi = (p-1) \times (q-1)$$
- Select small odd integer  $k$  relatively prime to  $\varphi$
$$\gcd(k, \varphi) = 1$$
- Compute  $d$  such that  $(d \times k) \% \varphi = (k \times d) \% \varphi = 1$
- Public key is  $(k, n)$
- Private key is  $(d, n)$

- example
 

$p = 11$
$q = 29$
$n = 319$
$\varphi = 280$
$k = 3$
$d = 187$
- public key  $(3, 319)$
- private key  $(187, 319)$

Private Key →  
single direction  
generate

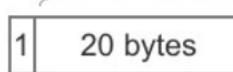
Public Key:



↓  
can we generate  
random private key  
so that the out  
put of wallet  
could look like:

0x40

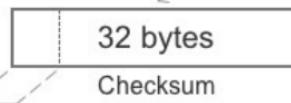
ripemd160(sha256( [ 1 32 bytes (BE) 32 bytes (BE) ] ))



Network ID Byte:

Main Network: 0x00  
Test Network: 0x6f  
Namecoin Net: 0x34

sha256(sha256( [ 1 20 bytes ] ))



25-byte binary address [ 1 20 bytes 4 ]

Base256-to-Base58 conversion  
(treat both quantities like big-endian)

1DS9U8Tjma2Kis3Z58ZKYbMUq4a8HXnuKp

I ssCynthia's Best ss . . . --

Who is responsible for accounts creation?

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

# How to verify that Tyr has enough coin?

- Alice got 50 btc
- Alice → Bob 10 btc
- Bob → Tyr 8 btc
- Alic → Tyr 10 btc
- ....
- Tyr → Cynthia 20btc

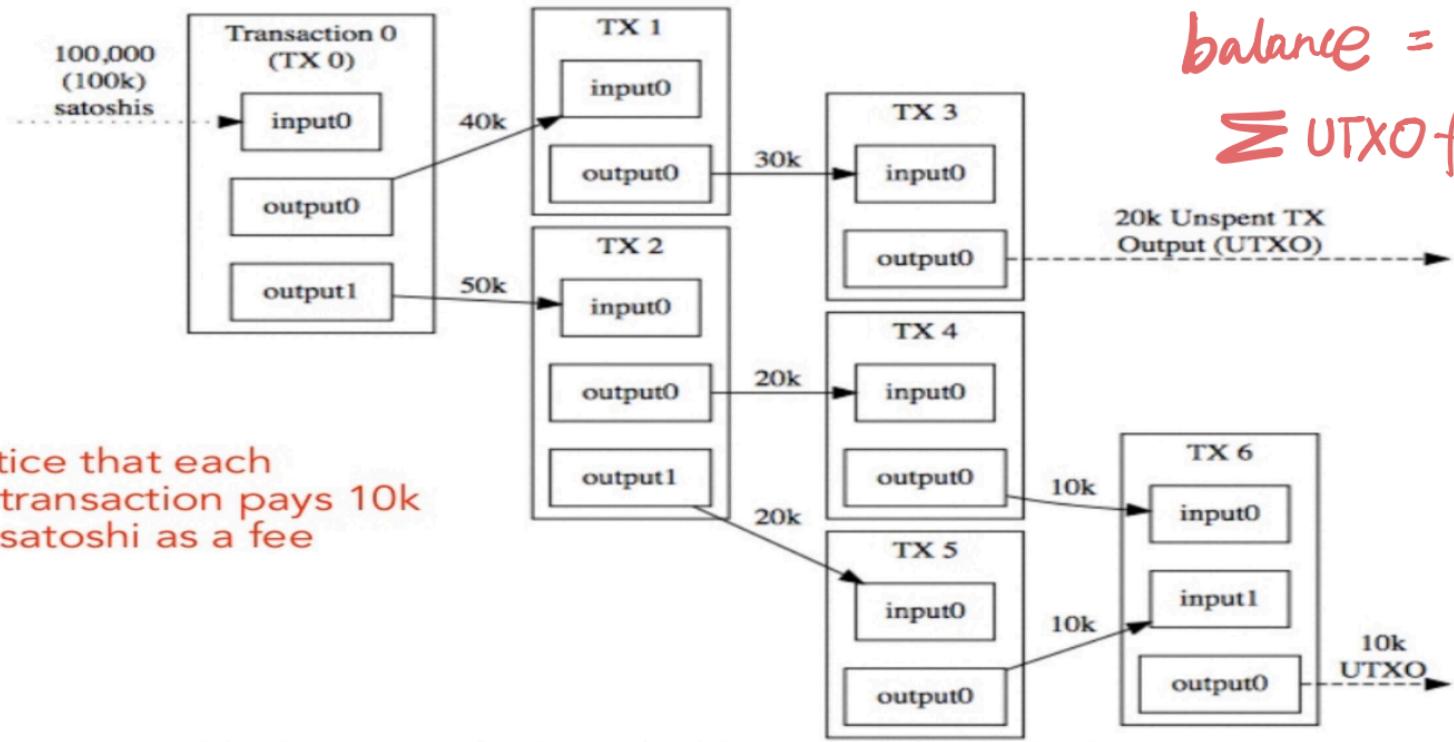
X

↑  
getBalanceOf(Tyr) = 18 btc

Problem: very inefficient  
solution: **UTXO**

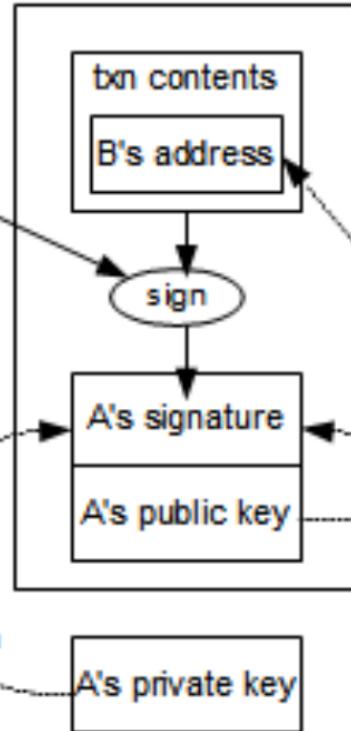
when Alice sent 10 btc to Bob  
she created a UTXO that only  
Bob can use it create new UTXO

# What is UTXO and why?

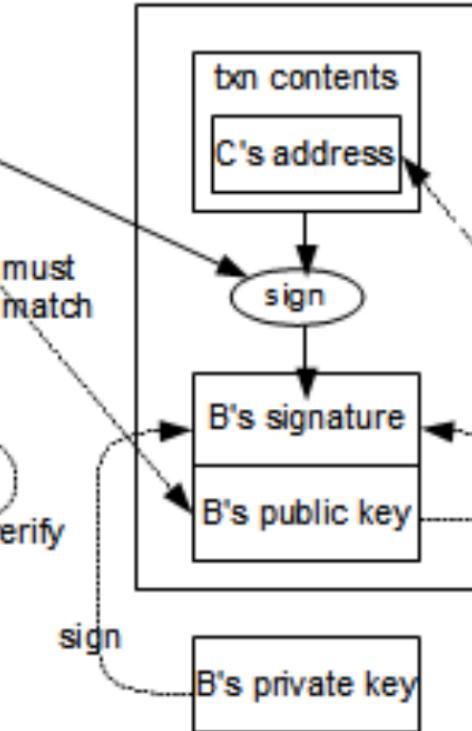


# How transactions work?

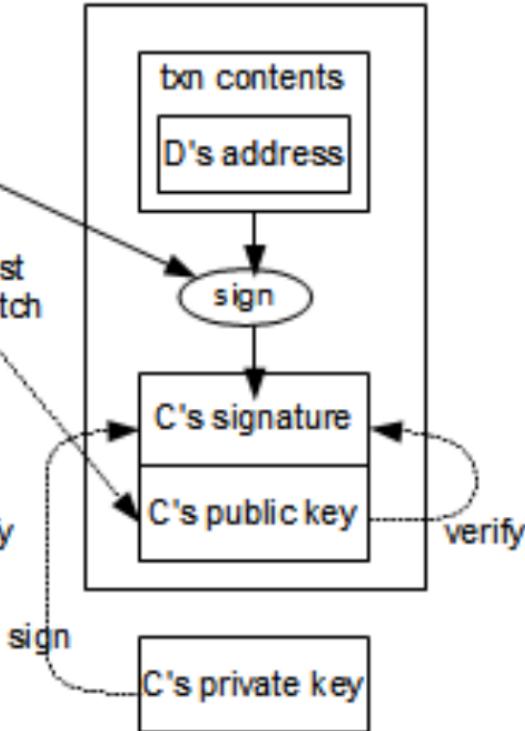
Transaction from A to B



Transaction from B to C



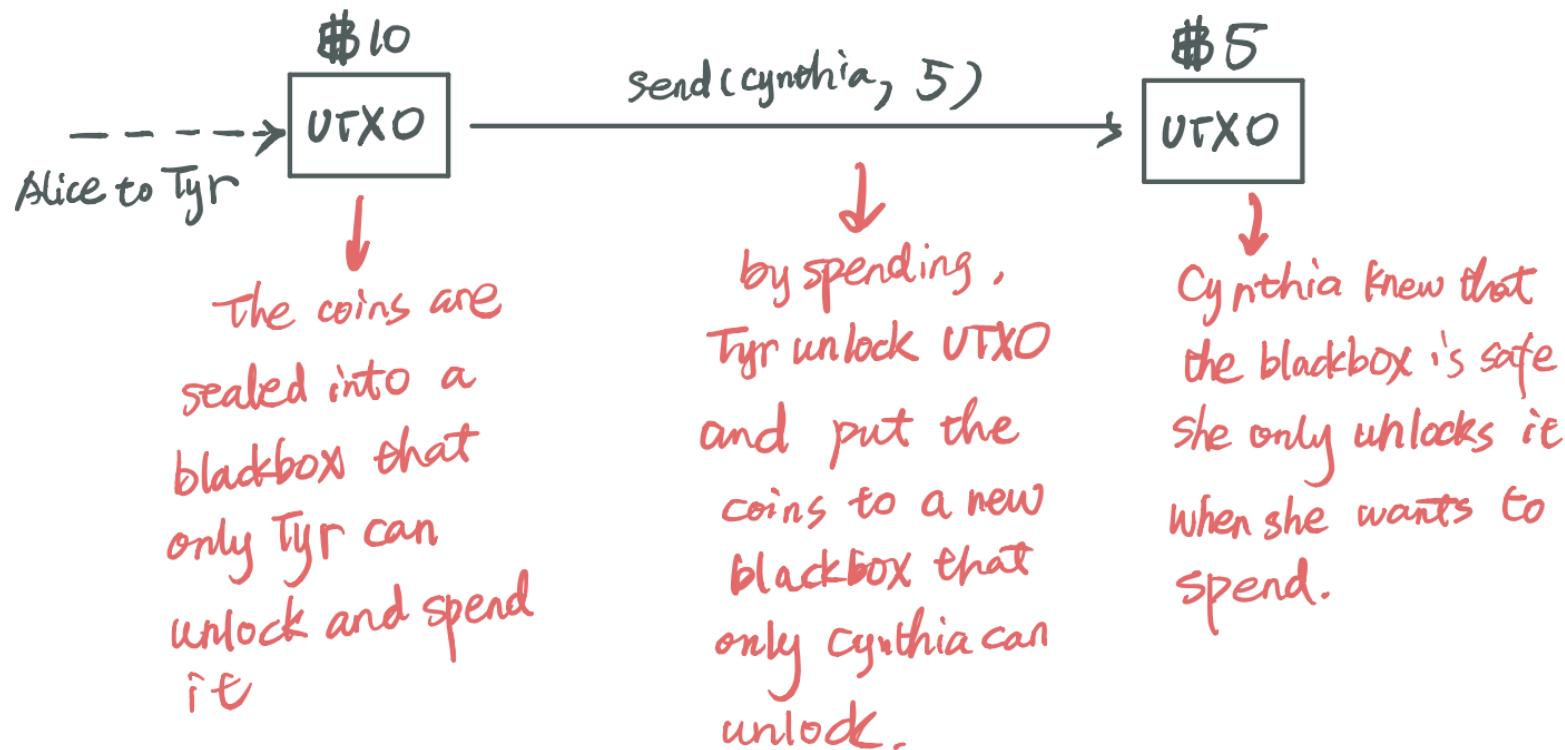
Transaction from C to D



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:

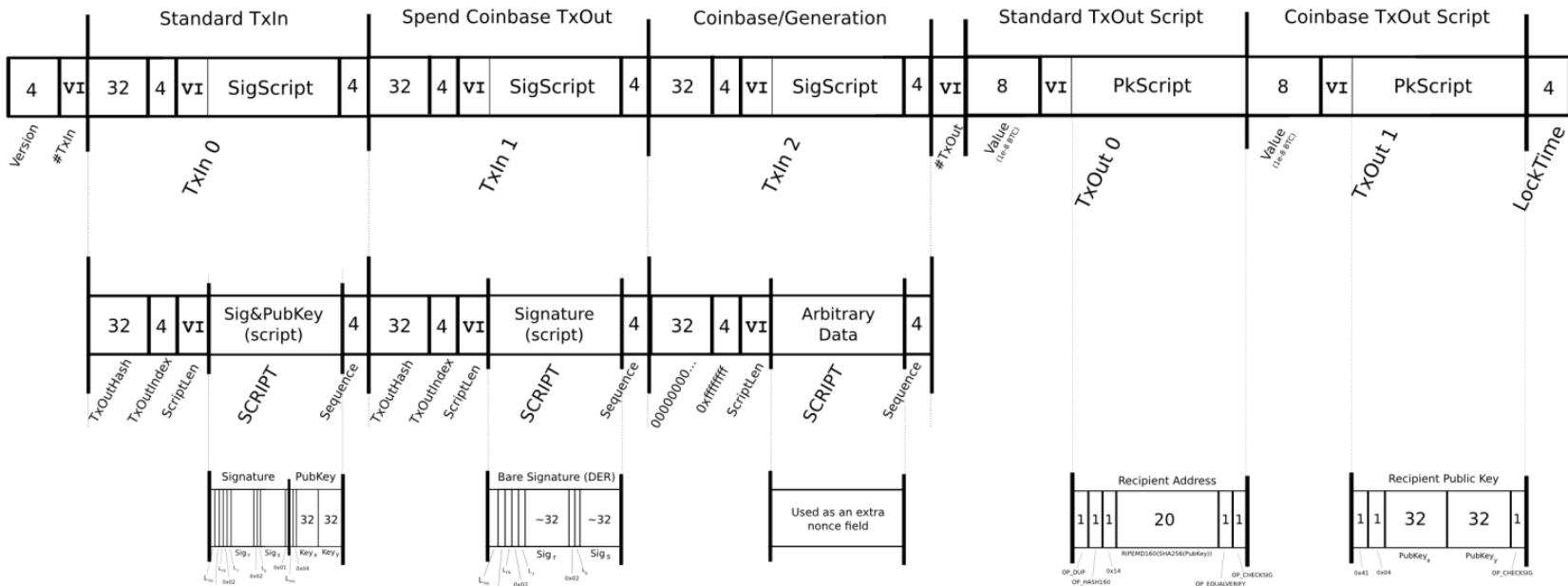
OP\_DUP OP\_HASH160 <Bob PK hash> OP\_EQUAL OP\_CHECKSIG

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>

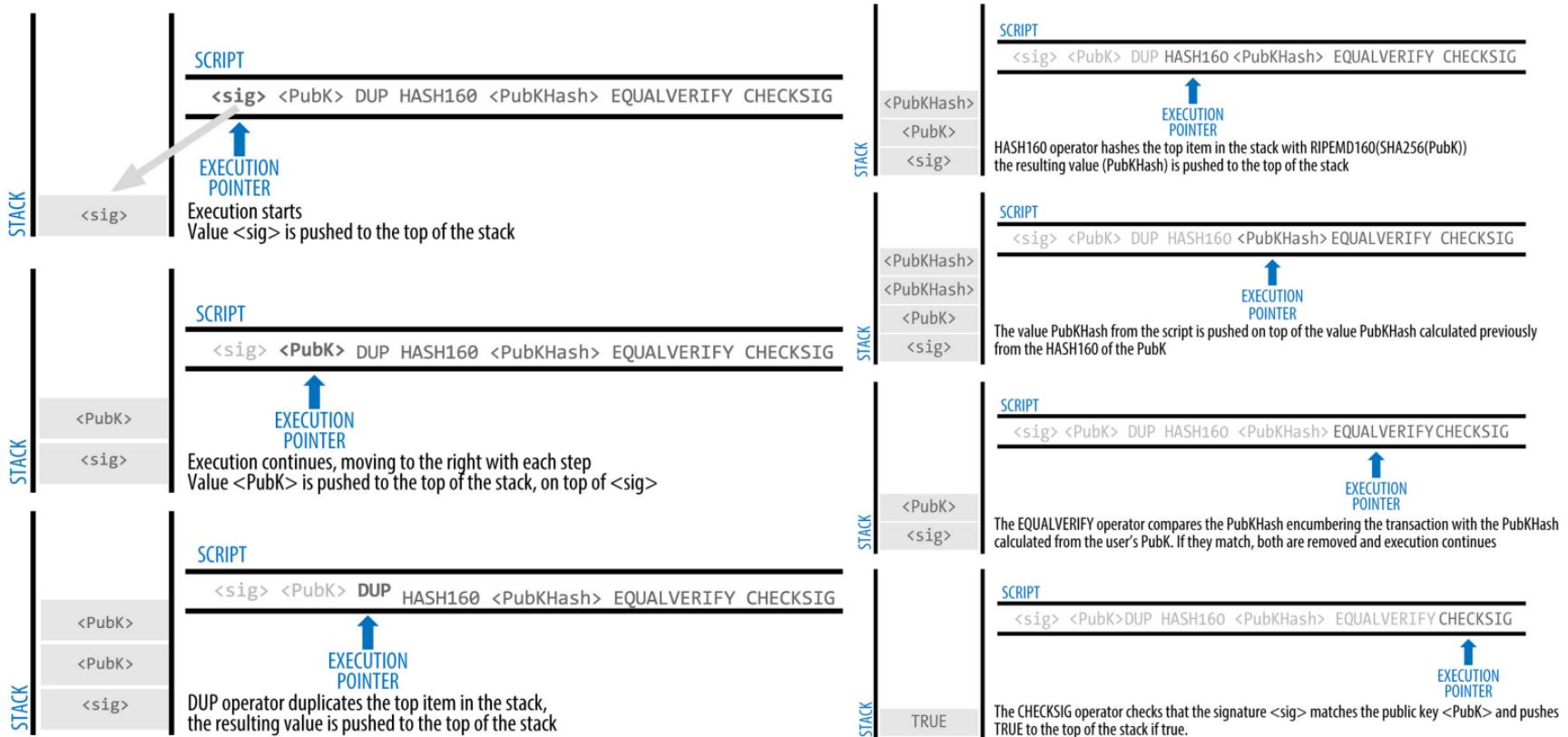
# What does a TX look like?



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

etotheipi@gmail.com / 1Gffm7LKXcNPFrtxy6yF4jBoe5rVka4sn1

# And how it gets executed?



What about security of the ecosystem?

- bad nodes
- bad nodes > 51% of computing power

SPV ←

Simplified  
Payment  
Verification

- chain is too big to fit a commodity node  
(availability)
- Threat from Quantum computing

Shor's algo

basics of

RSA

$$454243391453 \\ (299721) \times 349493$$

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

could be  
used in ECDSA

Grover's algo

given  $[a, b, \dots]$



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

Any problem

---

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

---

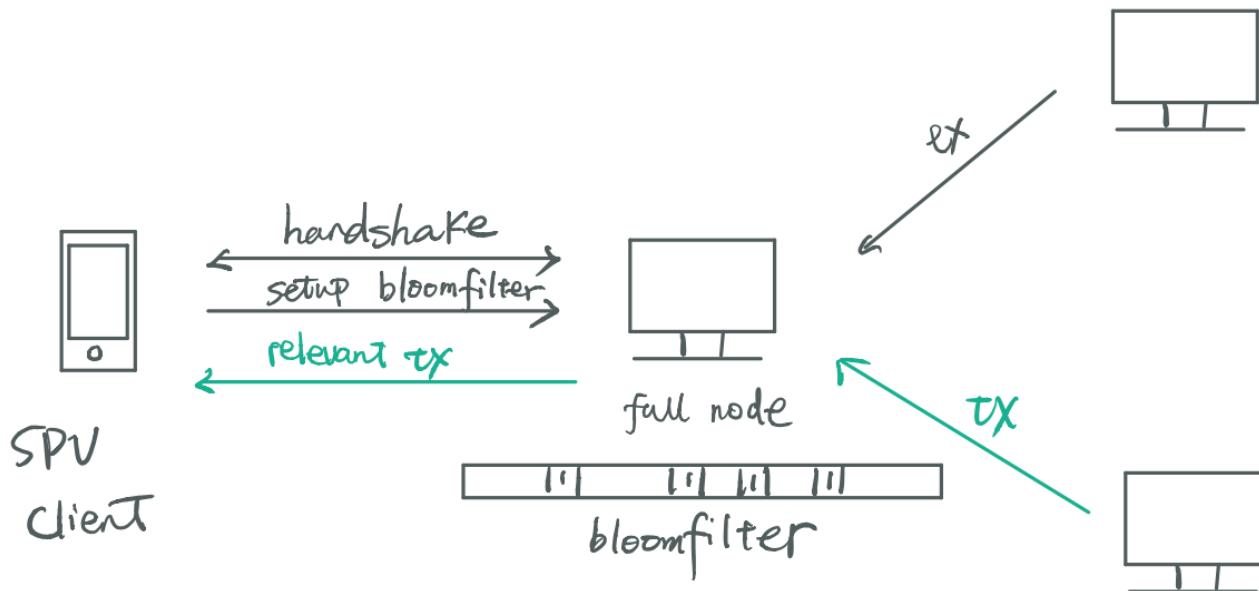
a few hundred Million

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?

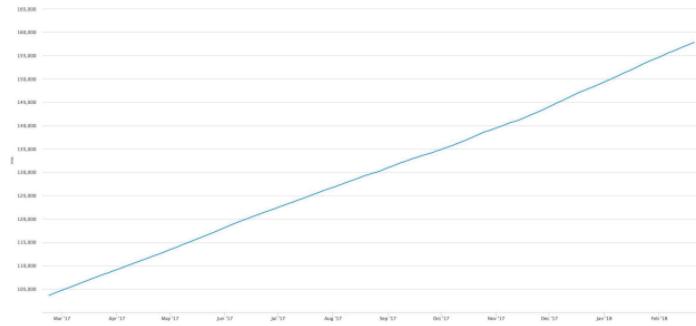
- 1 M block added 2010
- 10 Min a block  
 $(6 \times 24 \times 365 \times 9 = 474336)$  latest: 510438
- Wallet just need headers and related Merkle Tree path

why not bigger?

$1\text{MB} \times 8\text{bit} \times 7\text{peers}/30\text{sec} = 1.86\text{Mb/s}$

$8\text{MB} \times 8\text{bit} \times 7\text{peers}/30\text{sec} = 15\text{Mb/s}$

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
  - "
- $2 \cdot 1 \text{ quadrillion satoshis} \approx 2^{50.9} < 2^{53}$
- base58 - why why why!
  - L 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

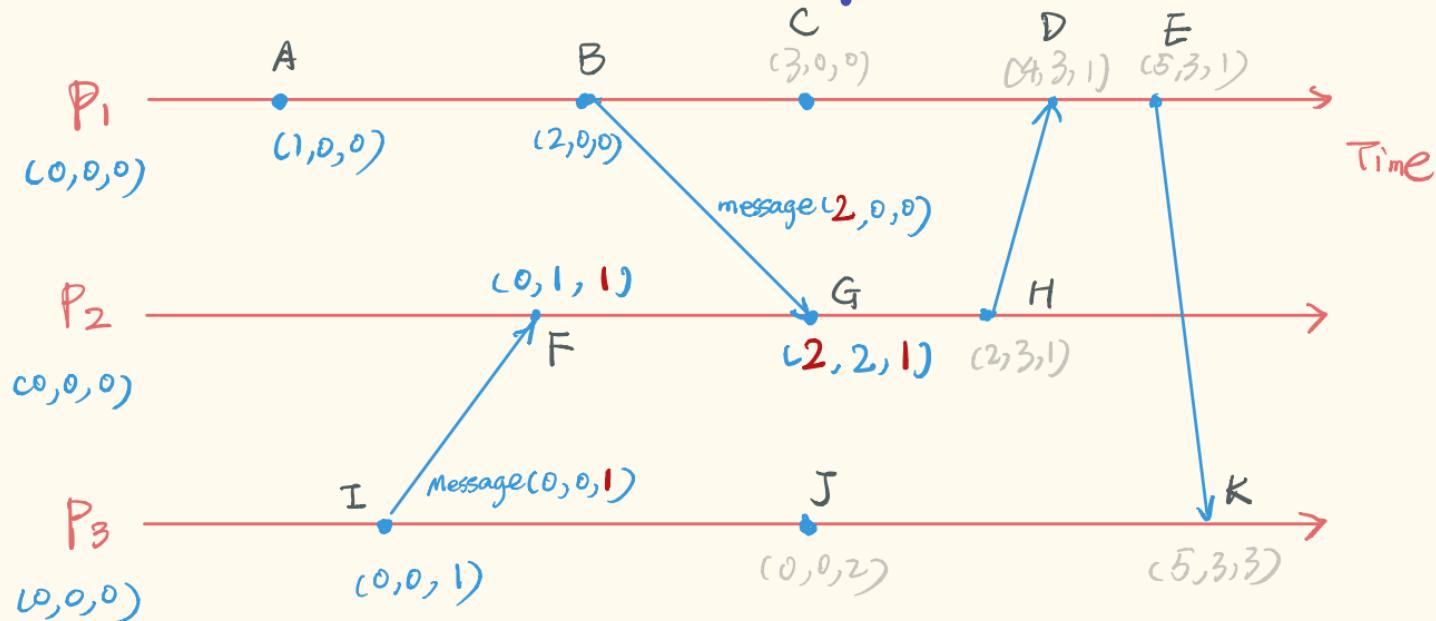
# Comparison

	Solo	Cloud computing	fully distributed
global clock	strict	yes	no
environment	simple, controlled	managed, a little moving pares	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

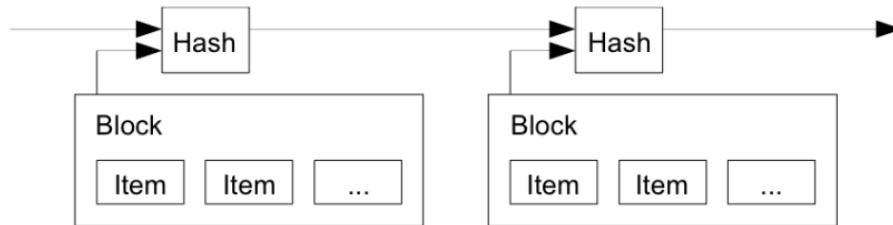
$$C \& G \quad (3, 0, 0) \parallel (2, 2, 1)$$

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

# 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.



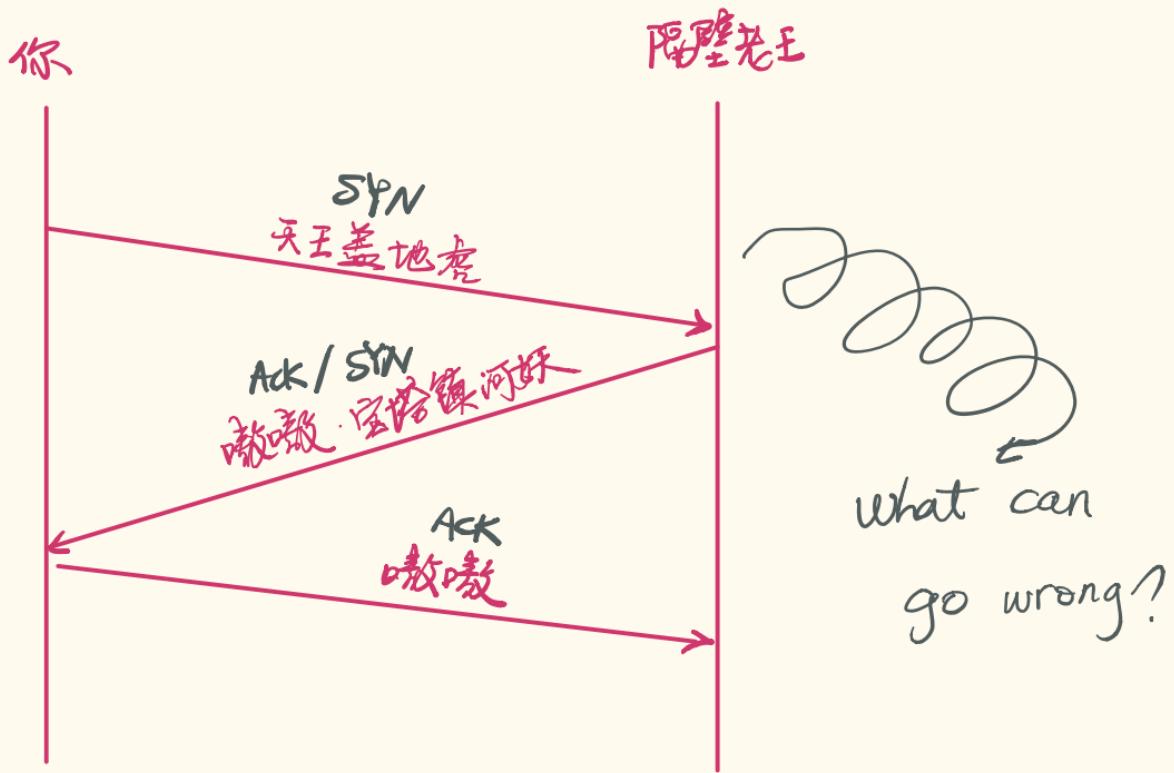
p2 , bitcoin paper

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

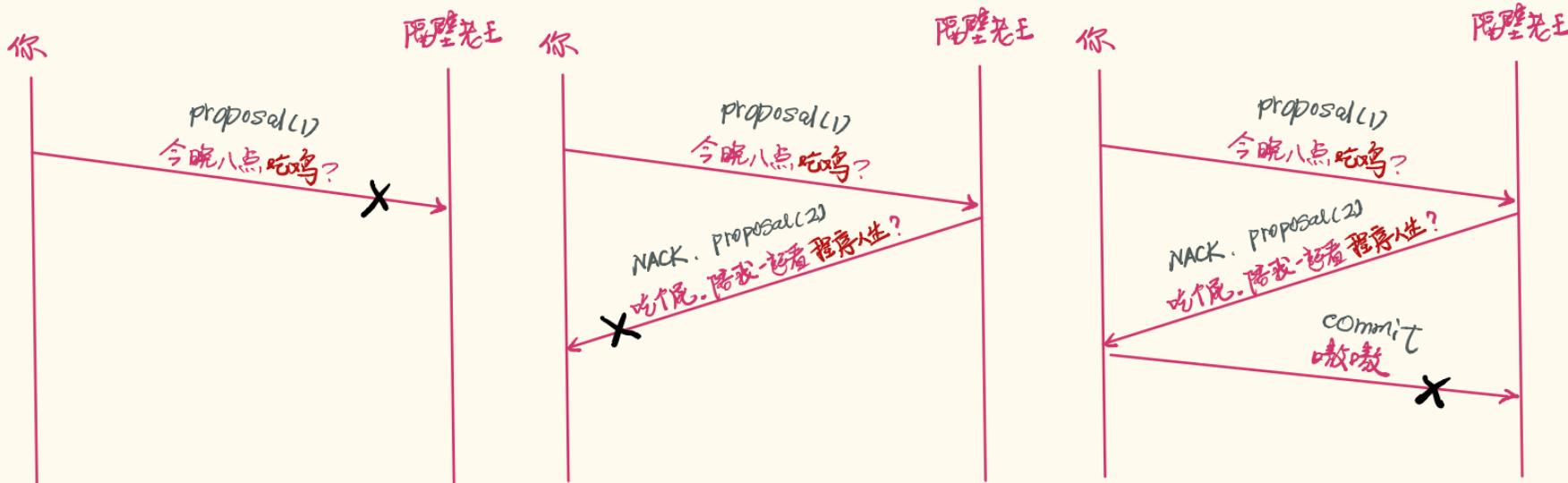
- DB (e.g. postgres)
- MQ (e.g. Kafka)
- service discovery tool (e.g. consul)
- . . .

hided those details for you

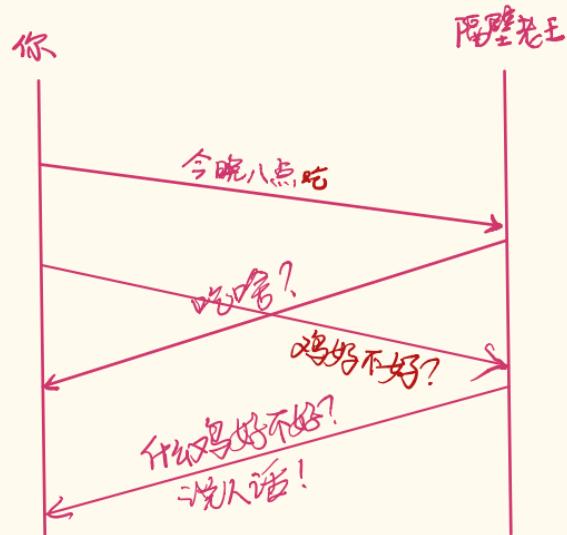
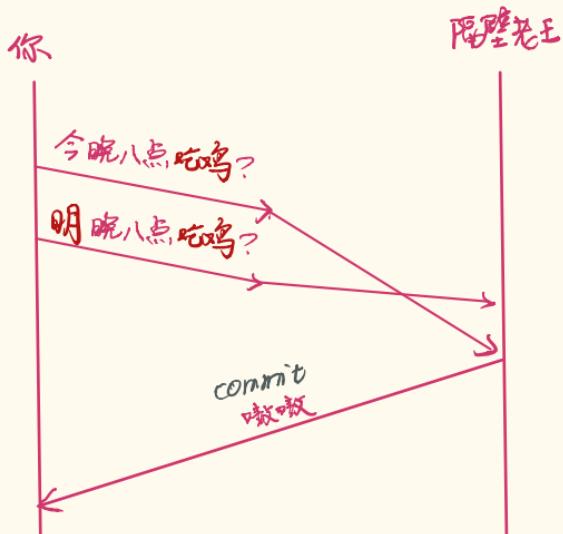
# Coordination & communication



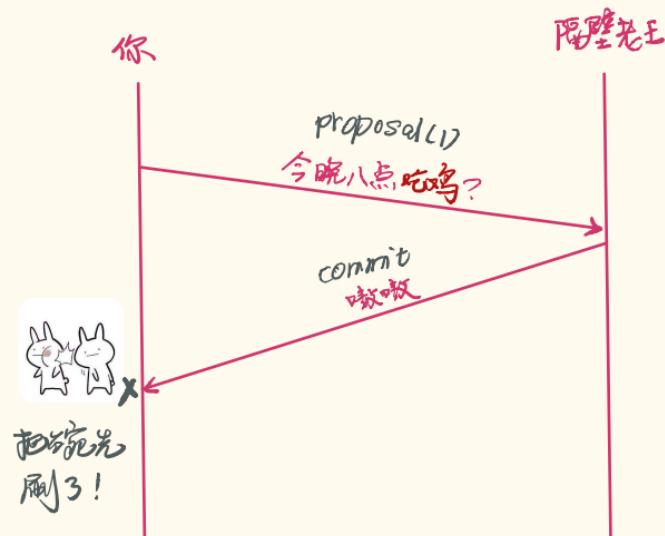
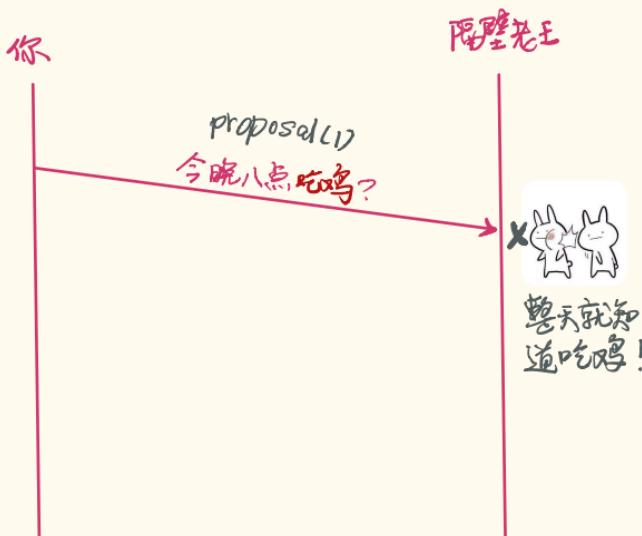
# Message Lost



# Out of order



# 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?

NACK. proposal(2)

吃饱了陪我一起看程序人生?

commit  
嗷嗷

consensus reached



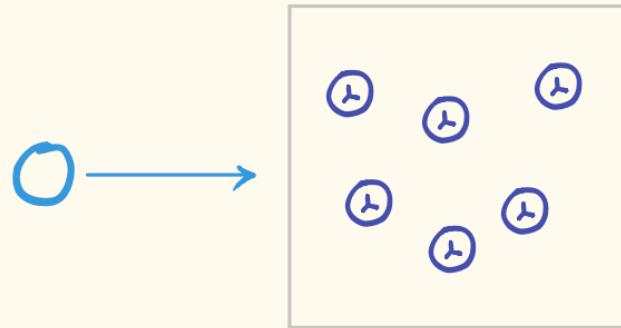
# Consensus

Traditional  
distributed systems  
(trust)

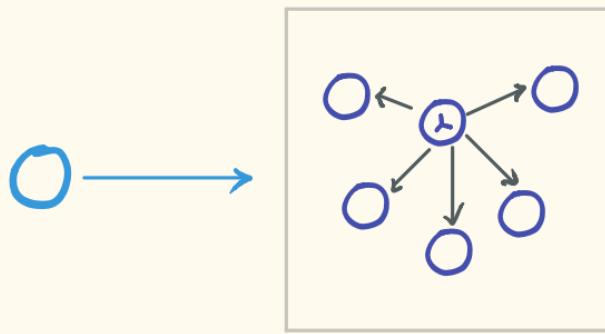
- Paxos
- Raft

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.

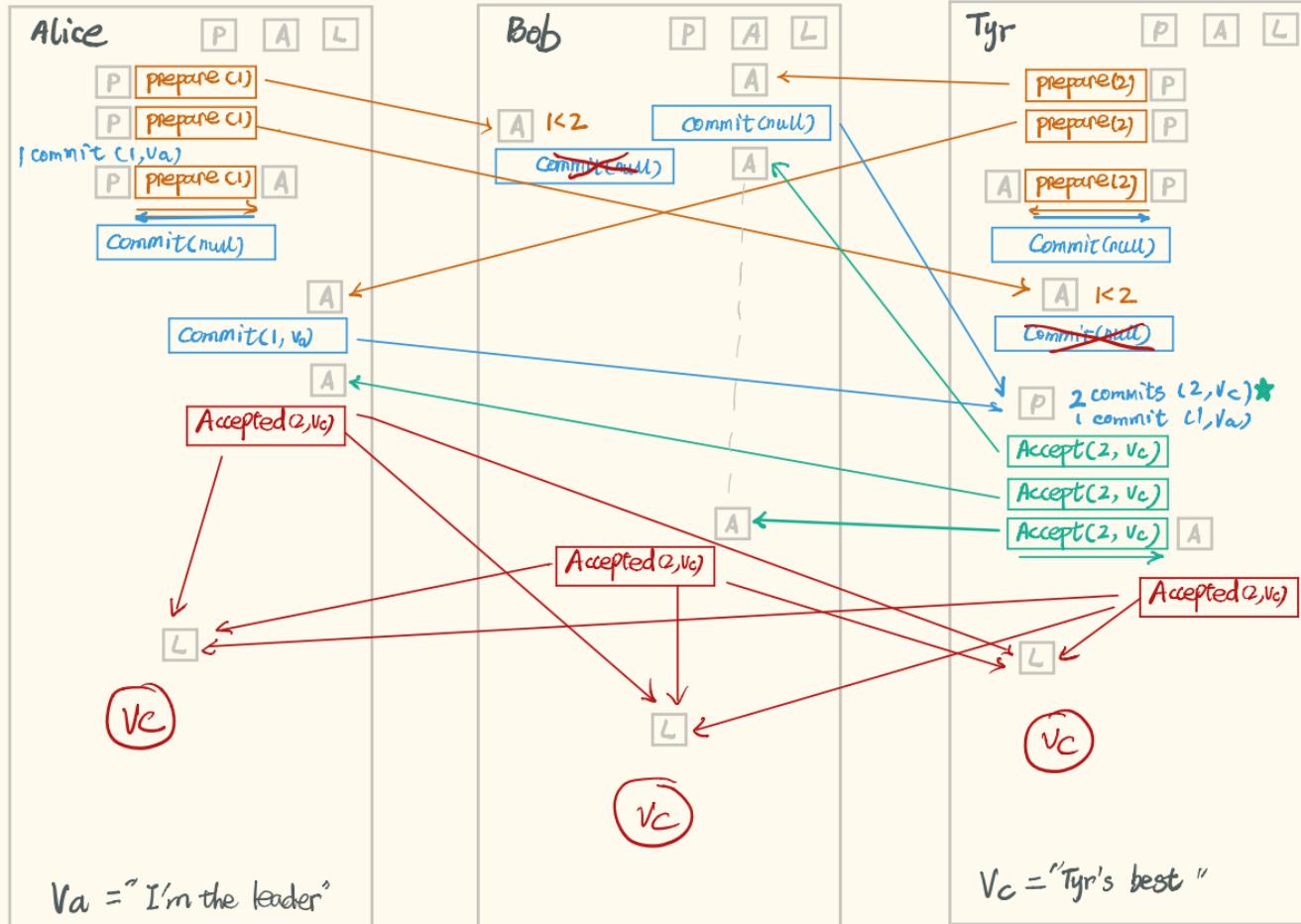
(b) If an acceptor receives an *accept* request for a proposal numbered  $n$ , it accepts the proposal unless it has already responded to a *prepare* request having a number greater than  $n$ .

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



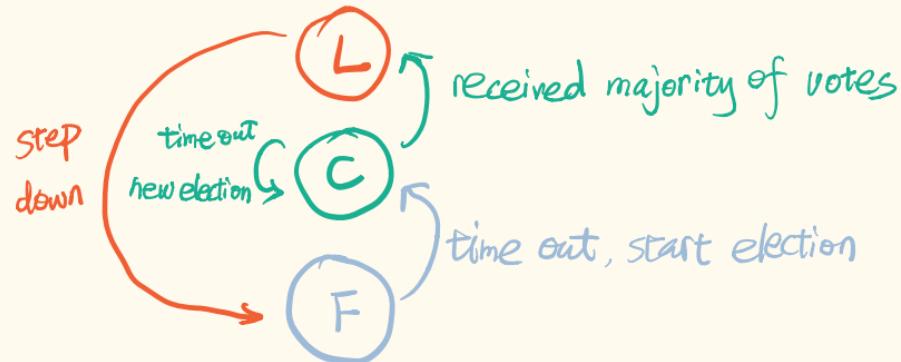
# Raft

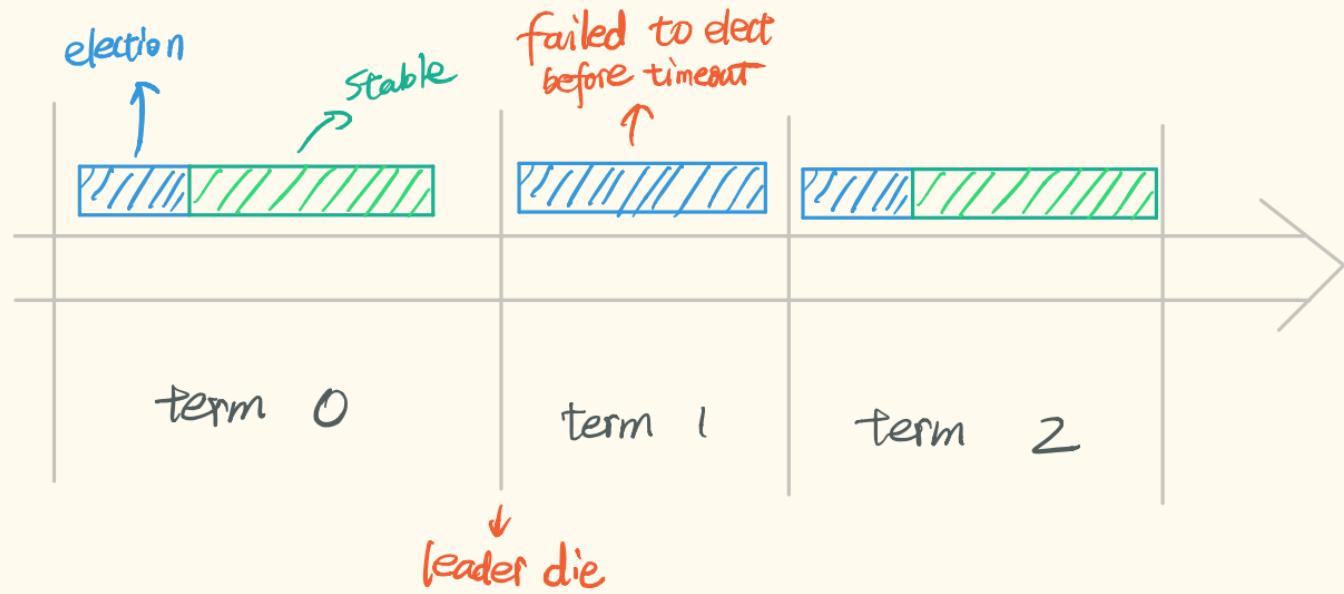
Goal: Have an available replicated state machine.

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

Assymmetric

- leader :
- Candidate :
- Follower





Stable >> election

# Election

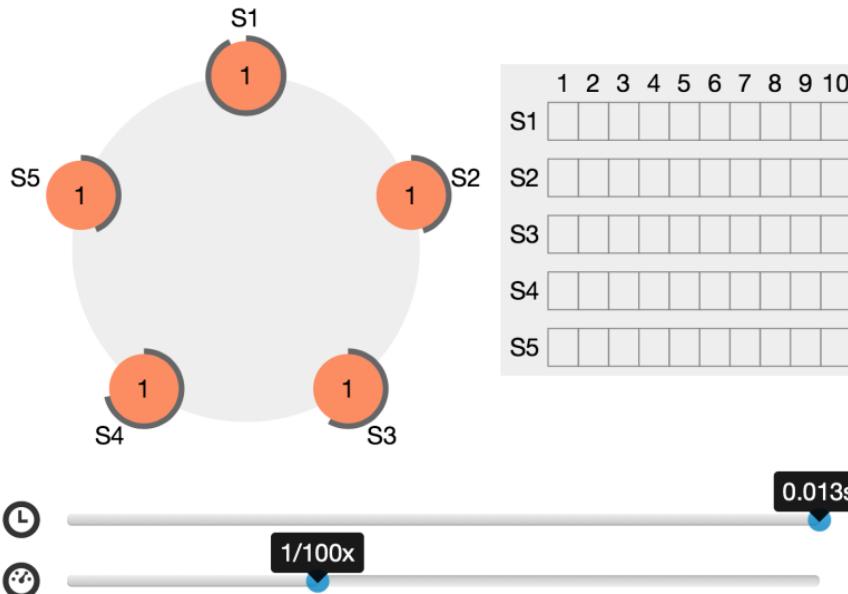
1. Increment current term
2. Vote for self
3. request votes

properties { safety : at most one winner per term.  
liveness : Someone eventually wins.

outcome { I'm leader : receive majority votes  
I'm not : receive valid heart beat  
no leader : election timeout

## Raft Visualization

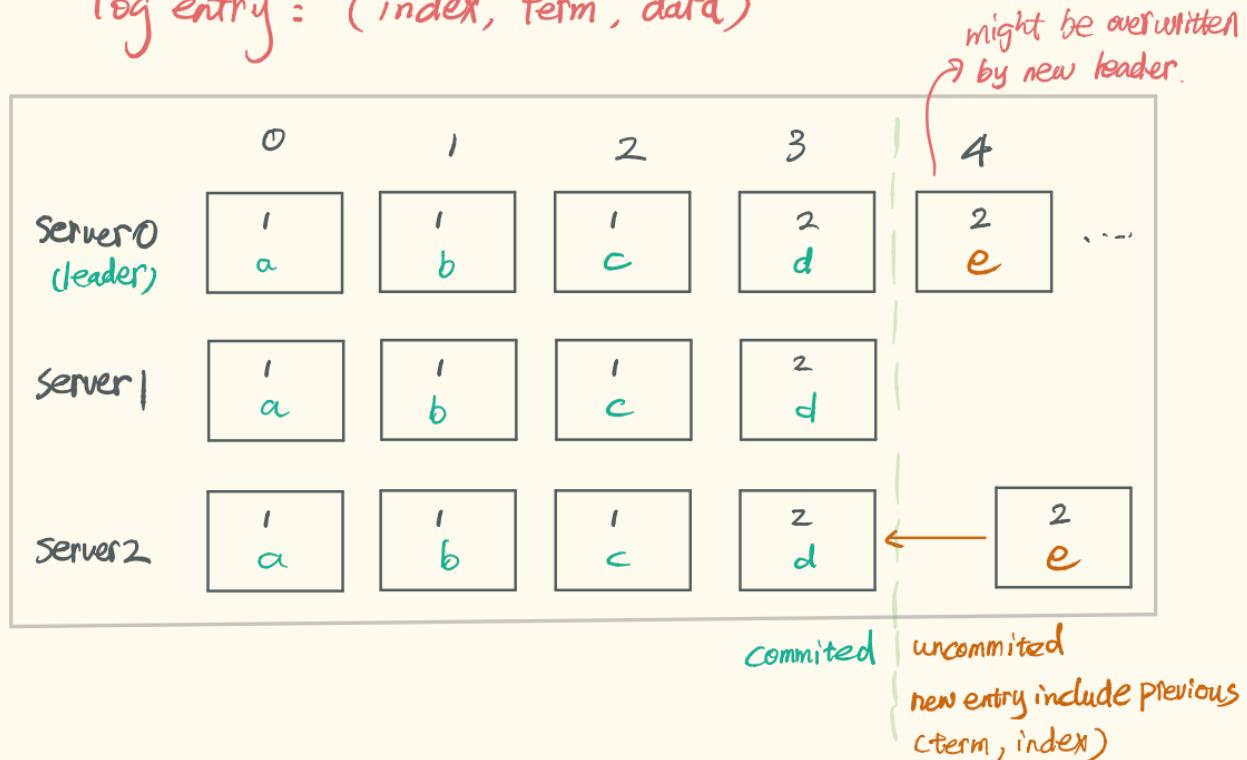
Here's a Raft cluster running in your browser. You can interact with it to see Raft in action. Five servers are shown on the left, and their logs are shown on the right. We hope to create a screencast soon to explain what's going on. This visualization ([RaftScope](#)) is still pretty rough around the 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 Foundry, Kubernetes, etcd, docker Swarm, 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  $\{ W = P t = \text{Hash Power} \times 10\text{ minutes}$   
something you pay people to do
- What is it solved

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
- incentivization
- side benefit

self reinforce  
lock in

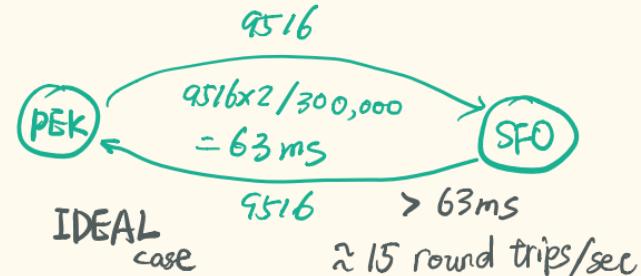
} get user onboard without buying it

# 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
  - $L = \text{distance} / c$



Lateney you should know

Latency Numbers Every Programmer Should Know

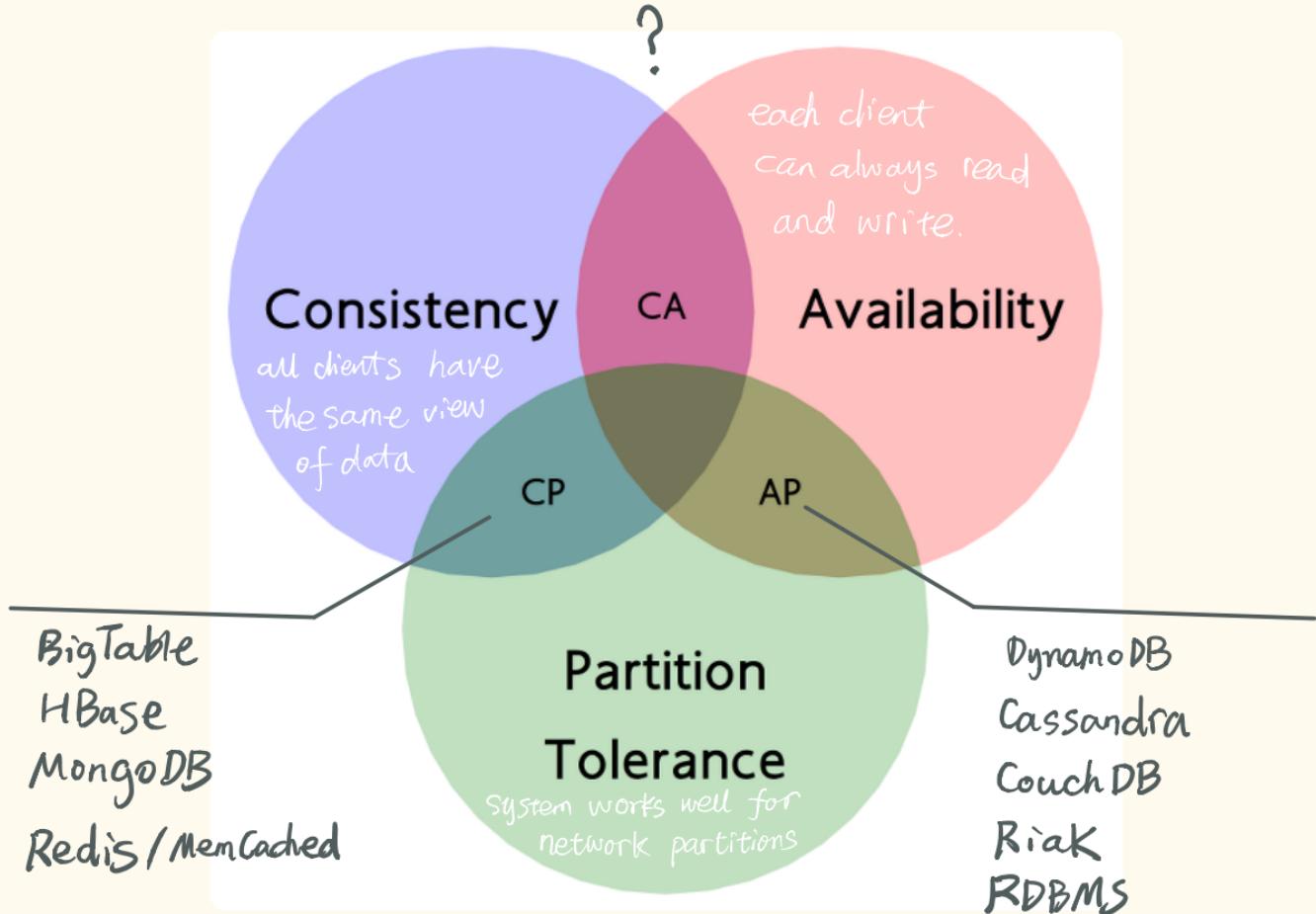
**Legend:**

- 1 ns
- L1 cache reference: 0.5 ns
- Branch mispredict: 5 ns
- L2 cache reference: 7 ns
- Mutex lock/unlock: 25 ns
- = ■ 100 ns
- Main memory reference: 100 ns
- = ■ 1 μs
- Compress 1KB with Zippy: 3 μs
- = ■ 10 μs
- Send 1KB over 1Gbps network: 10 μs
- SSD random read (1Gb/s SSD): 150 μs
- Read 1MB sequentially from memory: 250 μs
- Round trip in same datacenter: 500 μs
- = ■ 1 ms
- Read 1MB sequentially from SSD: 1 ms
- Disk seek: 10 ms
- Read 1MB sequentially from disk: 20 ms
- Packet roundtrip CA to Netherlands: 150 ms

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

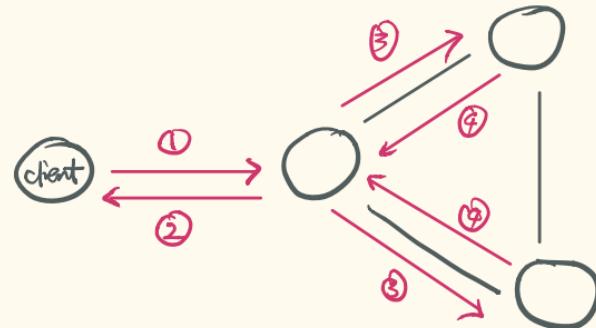
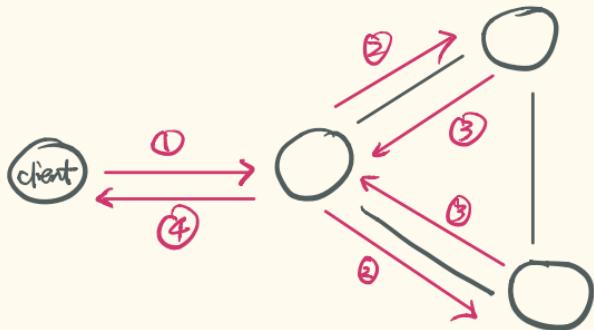


# Partition and Replication

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

## Algorithms

- Gossip
- Consistent hashing



from bitcoin to ethereum : evolution

# Alt coins

- Namecoin :

DNS: tubitr.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:  $\text{APPLY}(S, \text{TX}) \rightarrow S' \text{ or } \text{ERROR}$

metacoin:  $\text{APPLY}'(S, \text{TX}) \begin{cases} S' \\ S \text{ on } \text{ERROR} \end{cases}$

# Problems of bitcoin

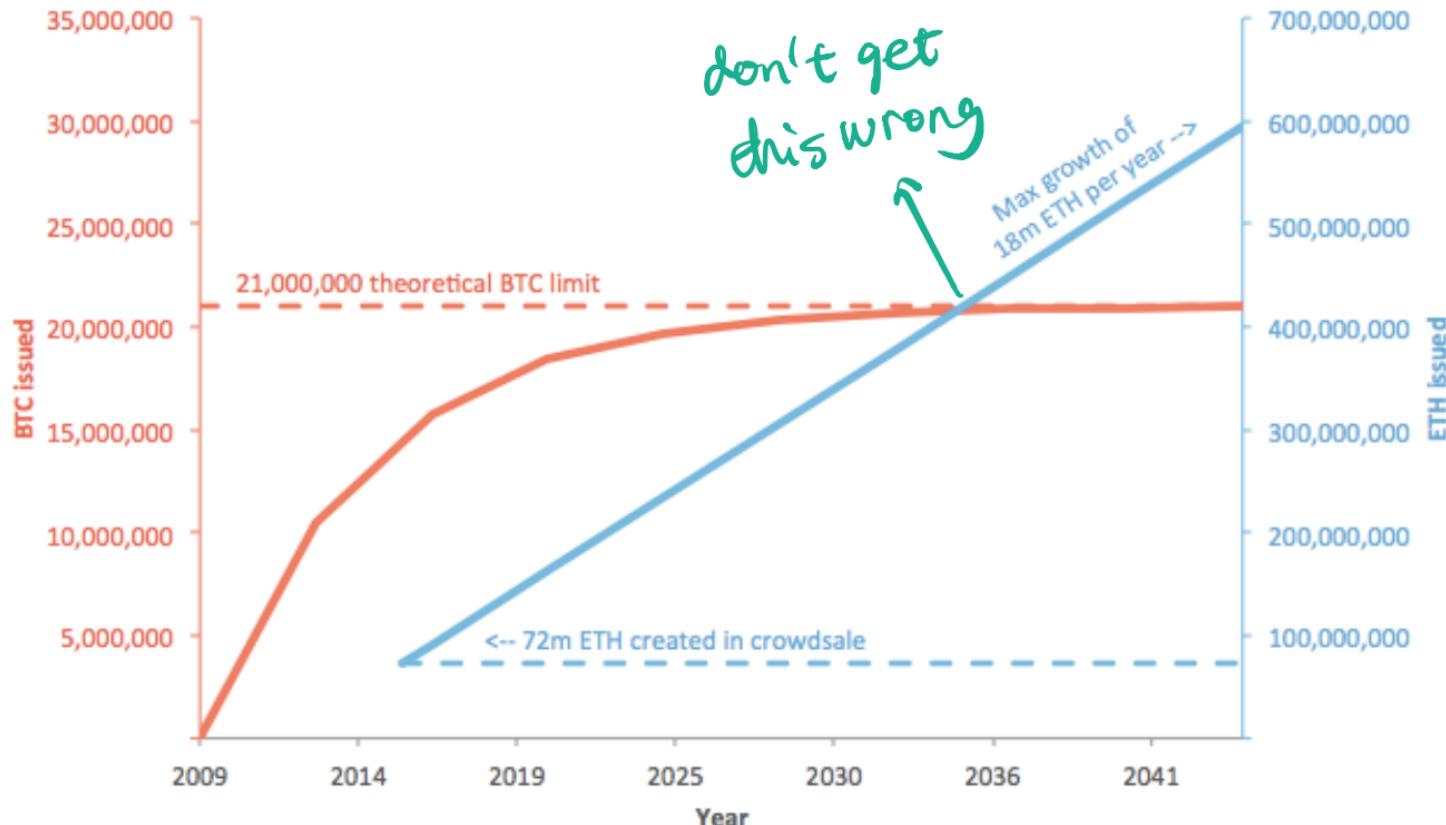
- slow
- scripting language is limited
  - Lack of turing completeness (e.g. no loop)
  - Lack of fine-grained control over UTXO
  - Lack of state (e.g. how to do multi-stage contract)
  - UTXO are blind to blockchain data

# Comparison

	bitcoin	ethereum
avg block time	10 min	12 sec (GHOST protocol)
consensus	PoW	PoW → PoS (Casper)
incentive	only the block creator	block creator & uncle
VM	simple & turing incomplete	turing complete ( <sup>use GAS</sup> to solve halting 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](http://www.bitsonblocks.net)



# Ethereum Account

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

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

UTXO vs balance  
||  
FP vs Imperative  
programming

# 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

## State

14c5f8ba:  
- 1024 eth

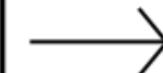
bb75a980:  
- 5202 eth

```
if !contract.storage[tx.data[0]]:  
    contract.storage[tx.data[0]] = tx.data[1]  
[0, 235235, 0, ALICE ...]
```

892bf92f:  
- 0 eth  
send(tx.value / 3, contract.storage[0])  
send(tx.value / 3, contract.storage[1])  
send(tx.value / 3, contract.storage[2])

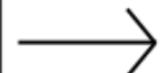
[ALICE, BOB, CHARLIE ]

4096ad65:  
- 77 eth



## Transaction

From:  
14c5f88a  
To:  
bb75a980  
Value:  
10  
Data:  
2,  
CHARLIE  
Sig:  
30452fdedb3d  
f7959f2ceb8a1



## State'

14c5f8ba:  
- 1014 eth

bb75a980:  
- 5212 eth

```
if !contract.storage[tx.data[0]]:  
    contract.storage[tx.data[0]] = tx.data[1]  
[0, 235235, CHARLIE, ALICE ...]
```

892bf92f:  
- 0 eth  
send(tx.value / 3, contract.storage[0])  
send(tx.value / 3, contract.storage[1])  
send(tx.value / 3, contract.storage[2])

[ALICE, BOB, CHARLIE ]

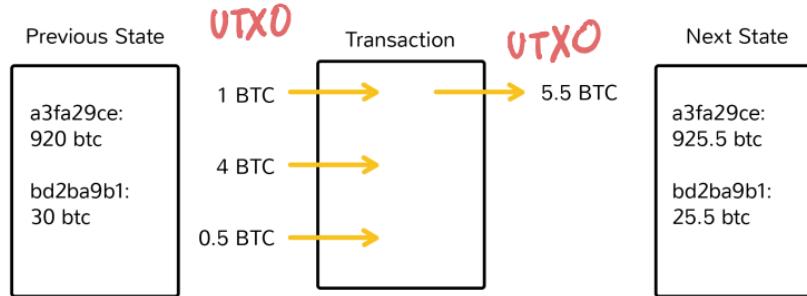
4096ad65:  
- 77 eth

## In detail

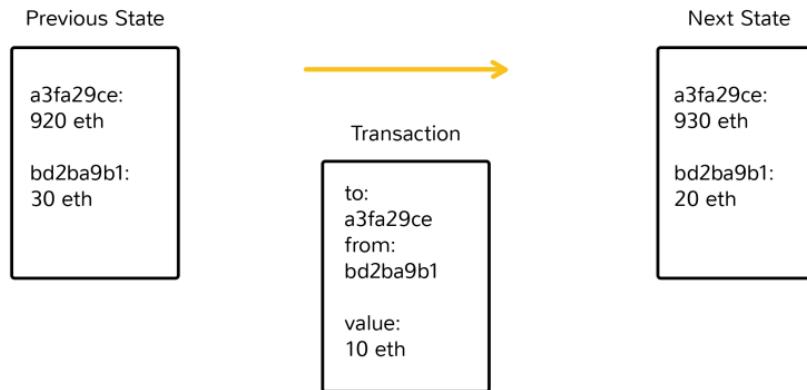
- Sanity check
- cost :  $\text{STARTGAS} \times \text{GASPRICE}$
- $\text{GAS} = \text{STARTGAS}$ . reduce GAS based on TX bytes
- transfer ether. If recipient not in state, create it  
if recipient is a contract account, run contract code
- rollback if GAS is used up.
- TX finished: return unused GAS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

# State Transition Comparison

Bitcoin



Ethereum

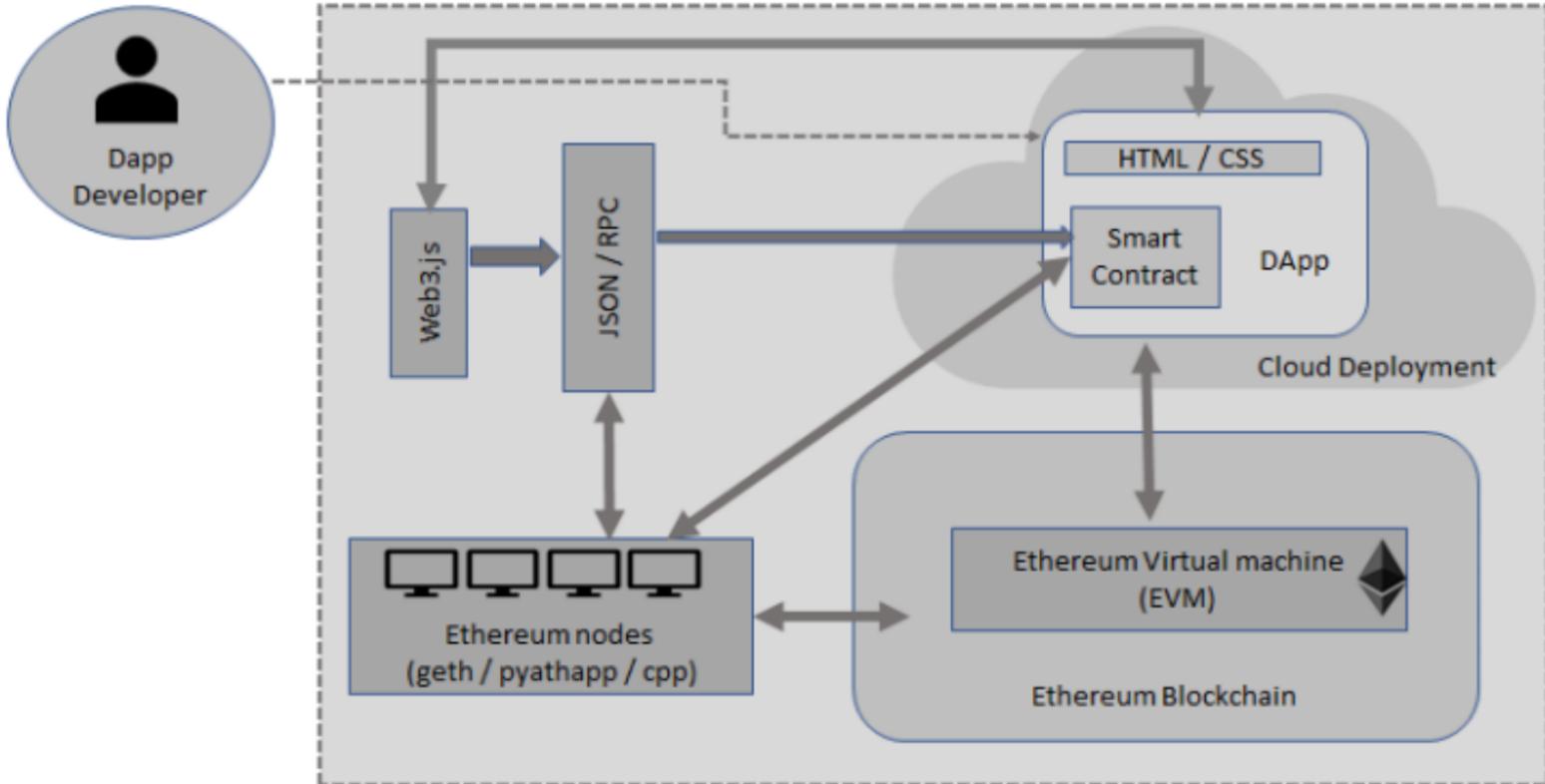


# EVM

- Stack based
  - Turing complete (e.g. introduced loop)
  - consumes GAS
  - To store data:
    - Stack
    - Memory (byte array)
    - Long-term storage (K/V store) - persistant
  - full computation state:  
(block.state, tx, message, code, memory, stack, PC, gas)
- deterministic  
is important

EVM, why not JVM?

# Ethereum Ecosystem



Smart Contract

# Vending Machines

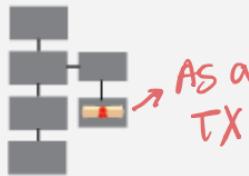
A Primitive Smart Contract



# 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;
    }
}
```

# 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

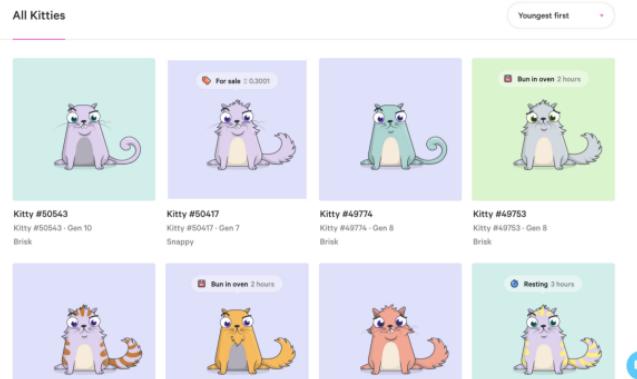
How much ERC20 tokens are currently lost (27 Dec, 2017):

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*

# ERC721 - Non-fungible

- unique, indivisible set of property
- Cannot be divided
- Scarcity: value might be different

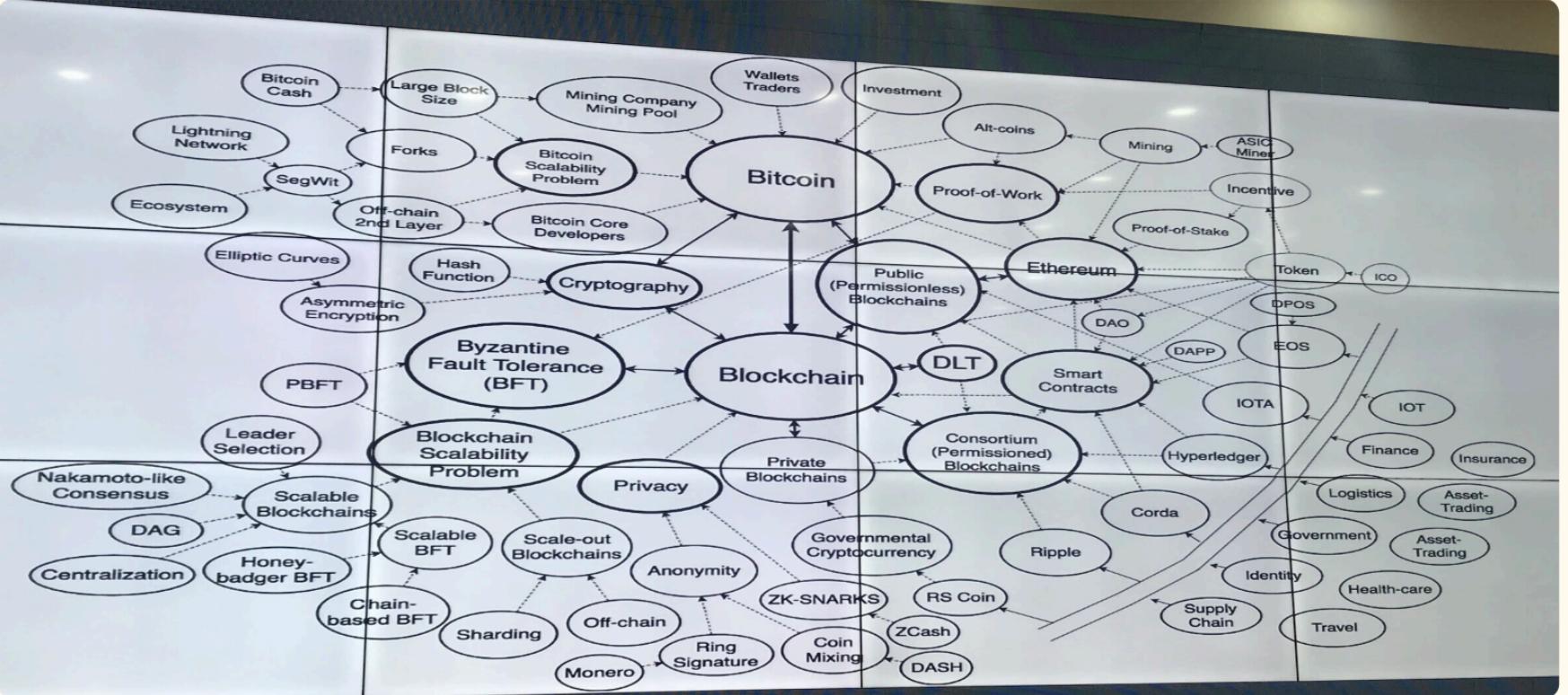


from ethereum to ?: what's next?

# Comparison

	Timestamp/ consensus	block time	use cases	features
Bitcoin	PoW/sha256	10min	coin transfer, multisig	UTXO, limited supply
Ethereum	PoW/ <u>ethash</u>	14-15 sec	ICO, dApps.	smart contract
bitshares	DPoS	3 sec	PEX, referral reward	fast/scalable tx/txs
Steemit			Social network	rewarding by activity, PoB
EOS	DPoS	500ms	OS for running app like Steemit	21 super node, WebAssembly

ethash: sha-3



# 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 Nutshell

- 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

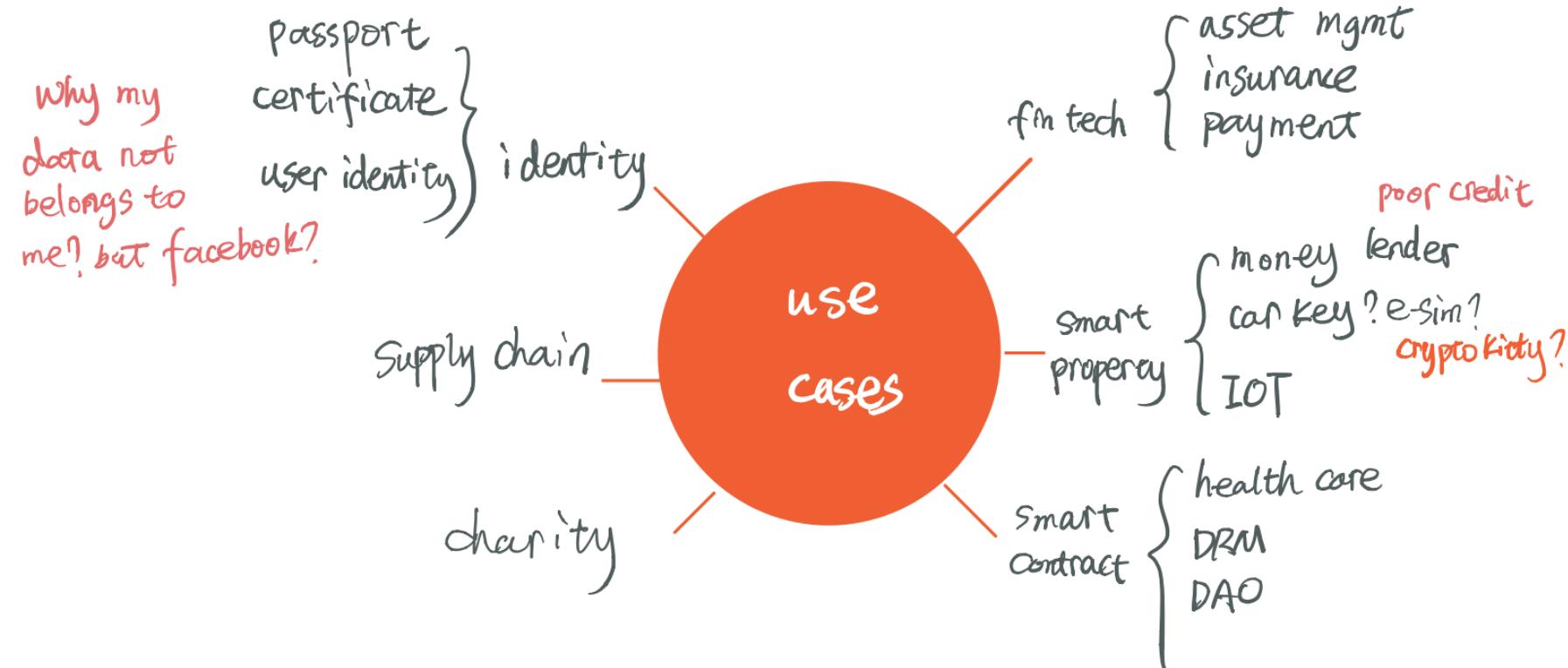


Account names	I don't want my users to remember random character strings. Know what a PK is. What a hash or 20 bytes means!	Nice human readable name that my user can unlock with a password
Account recovery	My user just forgot his password. Does he still have his private key saved? He does not seem to 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 transaction!
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 resources I need.
Speed	3 transactions per second? 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!

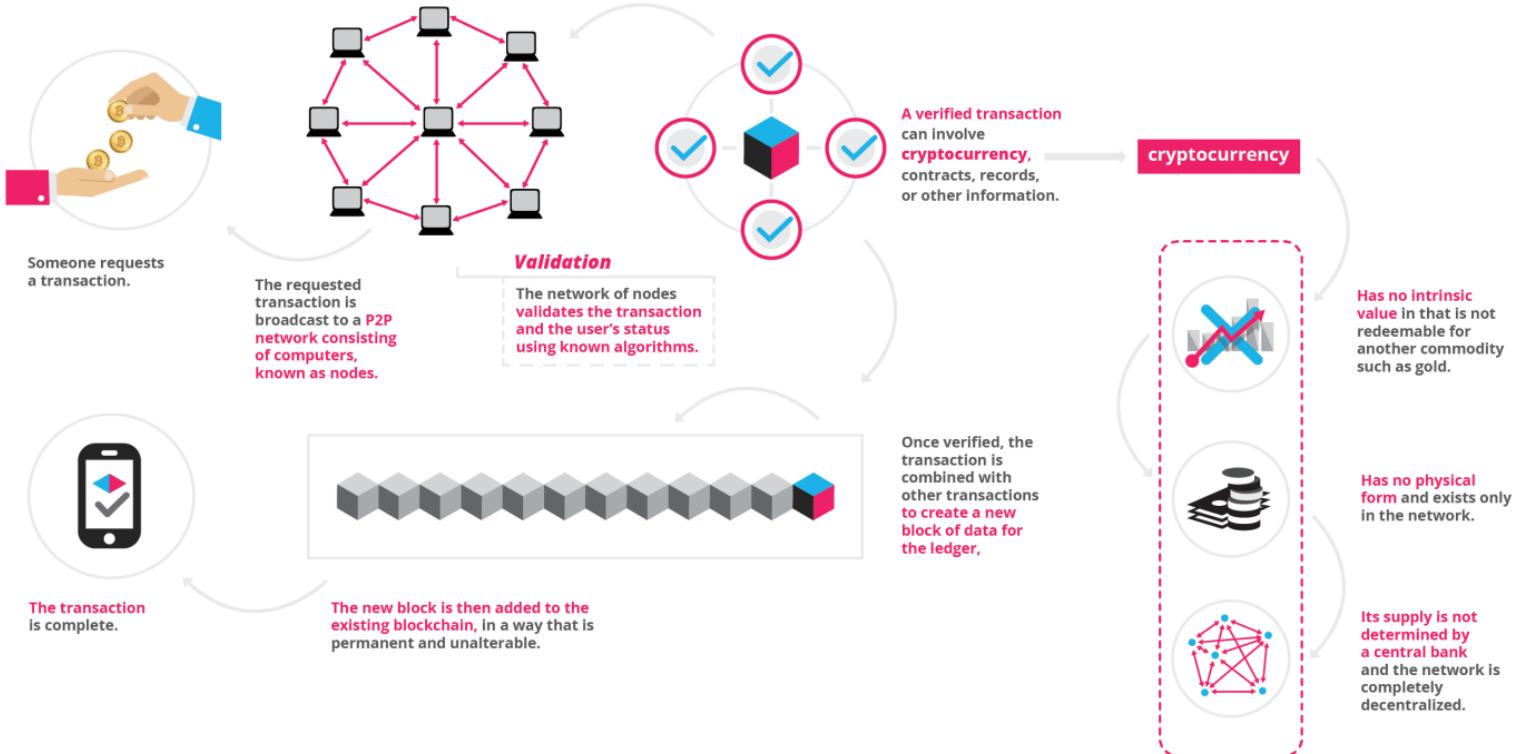
# Consensus

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

applications and innovations



# Payment



# Crypto Kitties



Kitty 2648



Kitty 2647



Kitty 2648



Kitty 2645



Kitty 2644



Kitty 2643



Kitty 2642



Kitty 2641



Kitty 2640



Kitty 2639



Kitty 2638



Kitty 2637



Kitty 2636



Kitty 2635



Kitty 2634



Kitty 2633



Kitty 2632



Kitty 2631



Kitty 2630



Kitty 2629



Kitty 2628



Kitty 2627



Kitty 2626



Kitty 2625



tokitties.co/kitty/2653



# 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  $\Rightarrow \max(X, Y) + 1$
- 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

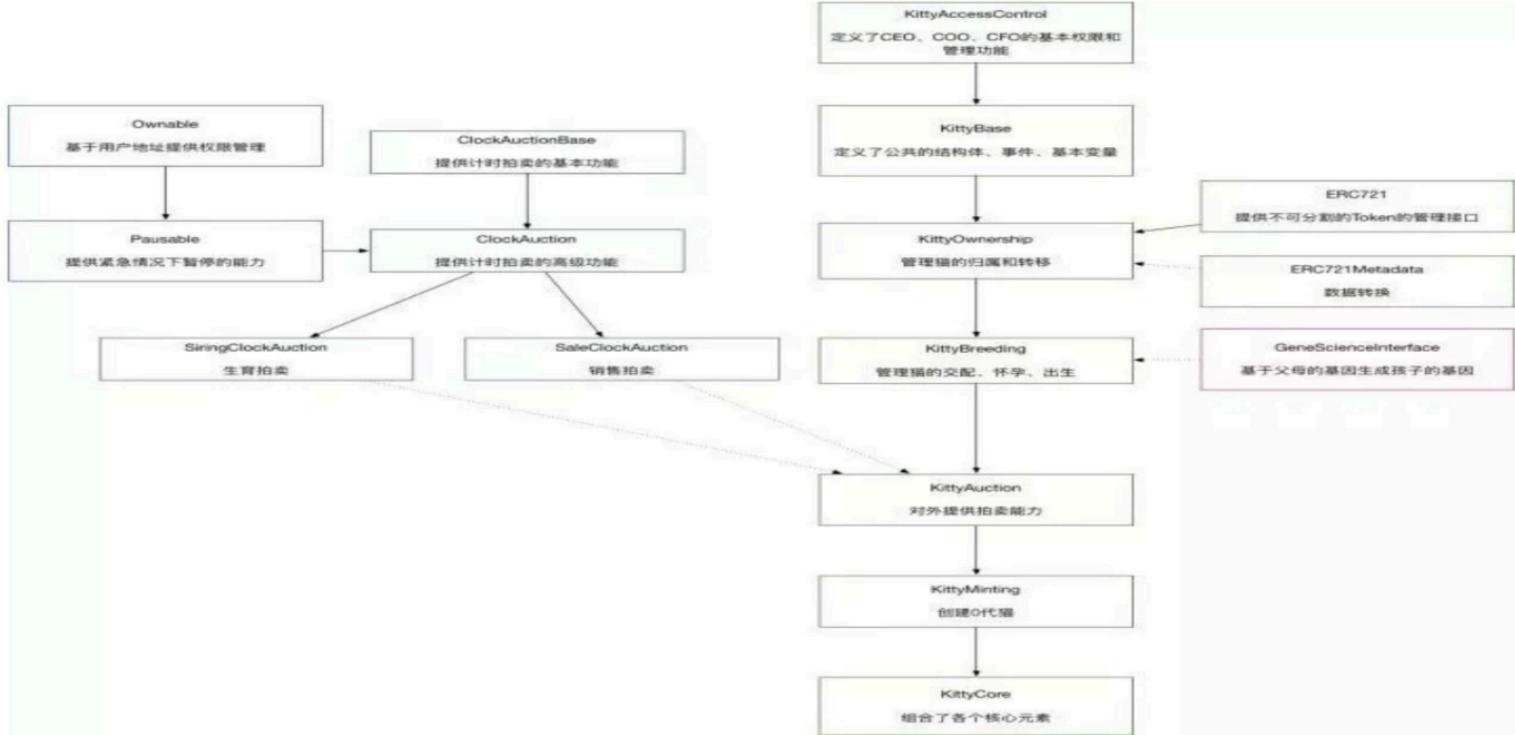
This kitty has no parents!

### Bebehs

This kitty has 0 bebehs

# Gene Code

- $256 \text{ bit} - 16 = 240 \text{ bit}$ 
  - 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



# Arc block



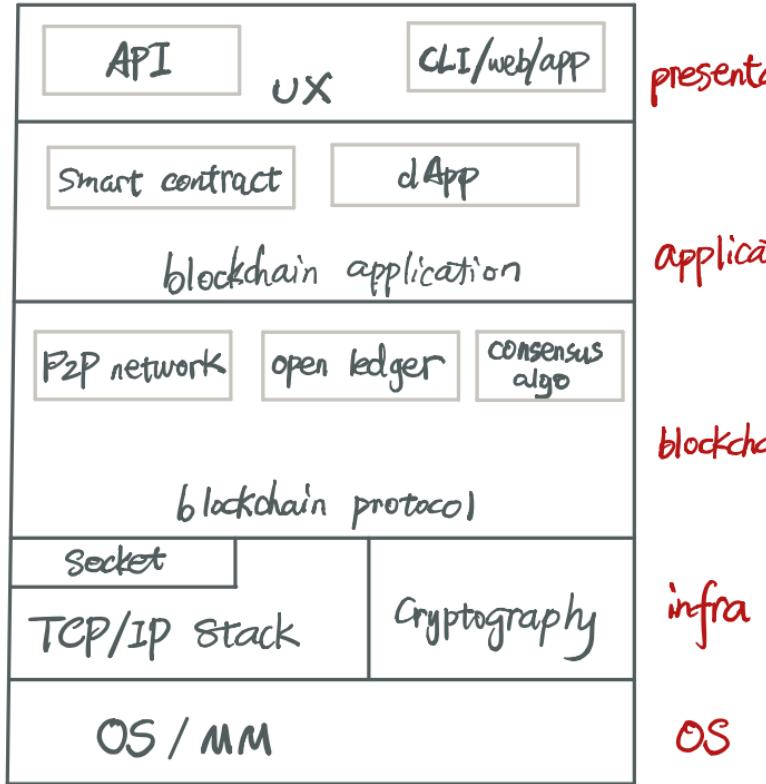
## Problem to solve

- 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 to a token?  
    , budget
  - 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 ?



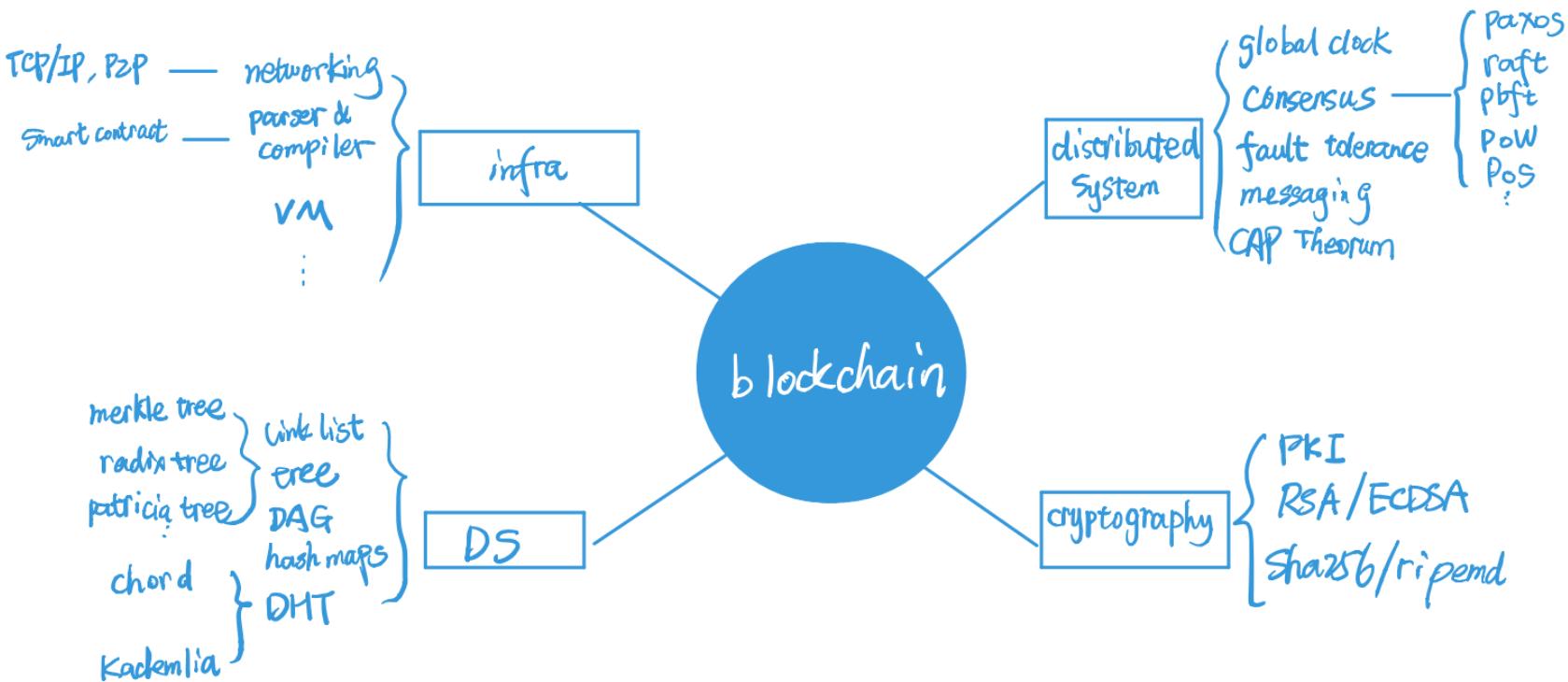
presentation layer

application layer

blockchain layer

infra layer

OS layer



## 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: belief is self-fulfilling prophecy





謝

謝！