Introduction to

Blockchain Science & Engineering

An informatics Master's level course

Aggelos Kiayias

Dionysis Zindros, Christos Nasikas

Consensus and Distributed Ledgers via the Blockchain Data structure

- The Consensus Problem
  - Properties
  - Honest Majority Assumption
- Ledger Consensus
  - Consistency
  - Liveness / Censorship resilience
- Blockchain Properties
  - Blockchain Addressing Notation
    - C[i], C[-i], C[i:j], C[i:], C[j:]
  - Common Prefix
  - Chain Growth
  - Chain Quality
- “Racing attacks”
- Consensus and distributed ledger (what is the relation)
- Difficulty Adjustment
  - Show difficulty graph
- The Bahack attack

Motivation for the consensus layer, I

- A transaction history and/or state of the service needs to be **agreed** by all servers.
- Servers may be operated by participants with **diverging interests** in terms of the history of transactions and/or state of the service.

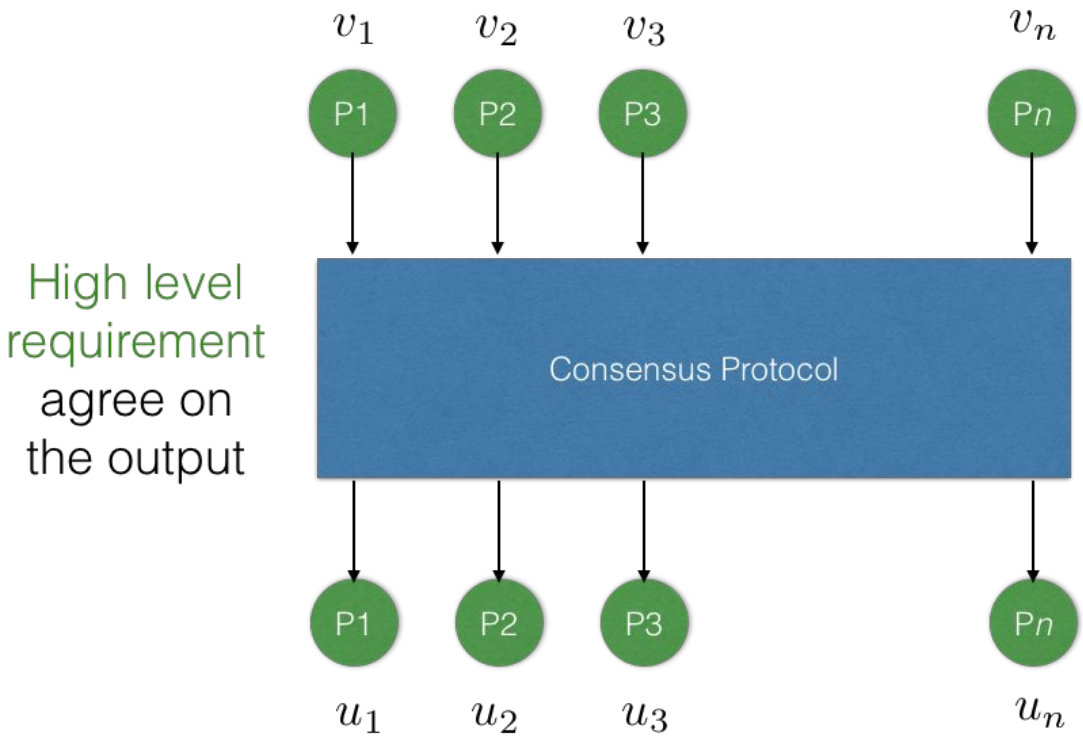
Motivation for the Consensus Layer, II

I prefer this!

I prefer this!

I prefer this!

The consensus problem



Study initiated by Lamport, Pease, Shostak 1982

Consesus : Problem Statement

- A number (say, *t*) of the participating entities can diverge from the protocol.
- This has been called Byzantine behaviour in the literature.
- The properties of the protocol are defined in the presence of this “malicious” coalition of parties that attempts to disrupt the process for the “honest” parties.

$$\mathrm{H}, |\mathrm{H}| = n - t$$

Consensus Properties

- Termination
- Agreement
- Validity
  - Strong Validity

$$\forall i \in \mathsf{H}(u_i \text{ is defined})$$

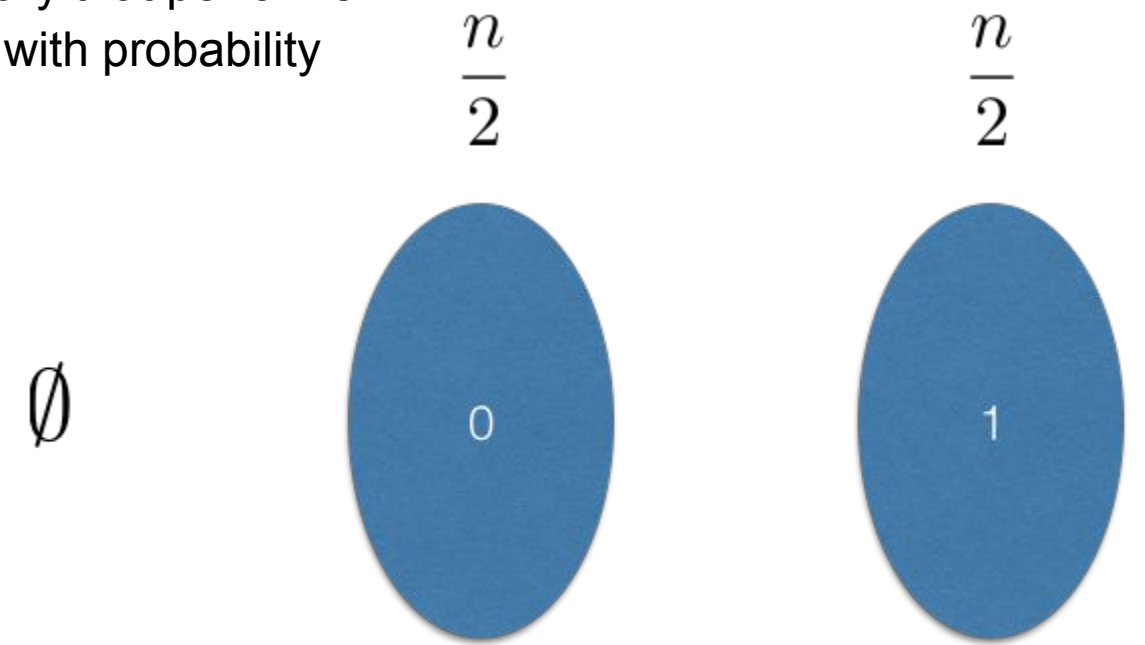
$$\forall i, j \in \mathsf{H}\,(u_i = u_j)$$

$$\exists v(\forall i \in \mathsf{H}\,(v_i = v)) \implies (\forall i \in \mathsf{H}\,(u_i = v))$$

$$\forall i \in \mathsf{H}\,\exists j \in \mathsf{H}\,(u_i = v_j)$$

Honest Majority is Necessary, I

Consider an adversary that performs one of the following with probability 1/3



Honest Majority is Necessary, II

- If the adversary corrupts A0, then output of honest parties (that belong to A1) should be 1.
- If the adversary corrupts A1, then output of honest parties (that belong to A0) should be 0.
- If the adversary corrupts no-one, then the output of all parties should be equal.

Is Honest Majority Sufficient?

- Two important scenarios have been considered in the consensus literature.
  - Point to point channels. **No setup.**
  - Point to point channels. **With setup.**

The setup enables provides a correlated private initialization string to each participant. It is assumed to be honestly produced.

Setup and Network

Setup/Network	Asynchrony / Partial Sync.	Synchrony
No Setup	t<n/3	t<n/3
With Setup	t<n/3	t<n/2

We know consensus can be achieved assuming the above bound on adversarial parties.

The typical setup and network in consensus

- Setup: a public-key directory
  - parties have signing and verification keys for a digital signature scheme.
  - Each party knows every other party’s verification key.
- Network: point-to-point channels

Enter Bitcoin (2008-09)

- Important concepts used
  - blockchain data structure.
  - proof of work (POW).
- both known and studied earlier, but put in combination for a novel application.

Blocks:

$$B = \langle s, x, ctr \rangle$$

pointer to previous

content

counter (proof of work witness)

$$(H(ctr, G(s, x)) < T) \wedge (ctr \leq q)$$

hash functions

target

upper bound

Blockchain

$$B_0 = \langle \perp, x_0, ctr_0 \rangle$$

$$B_1 = \langle s_1, x_1, ctr_1 \rangle$$

$$\vdots$$

$$B_n = \langle s_n, x_n, ctr_n \rangle$$

genesis block

$$s_i = H(ctr_{i-1}, G(s_{i-1}, x_{i-1}))$$

$$\mathcal{C} = \langle B_0, \ldots, B_n \rangle$$

head

$$\mathbf{x}_{\mathcal{C}} = \langle x_0, x_1, \ldots, x_n \rangle$$

$$\mathcal{C}^{\lceil k} = \langle B_0, \ldots, B_{n-k} \rangle$$

Blockchain Notation

C[i] = the i-th block from the start (0 is genesis)

C[-i] = the i-th block from the end (-1 is the tip)

C[i:j] = the range of blocks from [i,j)

C[:-k] = C with the k last blocks removed (or also $\mathcal{C}^{\lceil k}$)

$\mathcal{C}_1 \preceq \mathcal{C}_2$ prefix

The Bitcoin “backbone”

- The core of the bitcoin protocol
  - The chain validation predicate.
  - The chain selection rule (max-valid)
  - The proof of work function.
  - The main protocol loop
- Protocol is executed by “miners.”

Model

- Assume there are *n* parties running of the protocol
  - synchronously.
  - each one has a quota of *q* queries to the function H(.) in each round.
- A number of *t* parties are controlled by an adversary (a malicious coalition).

Algorithm 1 The *chain validation predicate*, parameterized by q, T, the hash functions $G(\cdot), H(\cdot)$, and the *content validation predicate* $V(\cdot)$. The input is $\mathcal{C}$.

function validate($\mathcal{C}$)
  $b \leftarrow V(\mathbf{x}_{\mathcal{C}})$
  if $b \wedge (\mathcal{C} \neq \varepsilon)$ **then** ▷ The chain is non-empty and meaningful w.r.t. $V(\cdot)$
    $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
    $s' \leftarrow H(ctr, G(s, x))$
    repeat
      $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$
      if $\text{validblock}_q^T(\langle s, x, ctr \rangle) \wedge (H(ctr, G(s, x)) = s')$ **then**
        $s' \leftarrow s$ ▷ Retain hash value
        $\mathcal{C} \leftarrow \mathcal{C}^{\lceil 1}$ ▷ Remove the head from $\mathcal{C}$
      else
        $b \leftarrow$ False
      end if
    until $(\mathcal{C} = \varepsilon) \vee (b =$ False$)$
  end if
  return (b)
end function

Algorithm 2 The function that finds the "best" chain, parameterized by function $\max(\cdot)$. The input is $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$.

```
1: function maxvalid(C_1, ..., C_k)
2:     temp ← ε
3:     for i = 1 to k do
4:         if validate(C_i) then
5:             temp ← max(C_i, temp)
6:         end if
7:     end for
8:     return temp
9: end function
```

Algorithm 4 The Bitcoin backbone protocol, parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$.

$\mathcal{C} \leftarrow \varepsilon$
$st \leftarrow \varepsilon$
$round \leftarrow 1$
while TRUE **do**
  $\tilde{\mathcal{C}} \leftarrow \mathsf{maxvalid}(\mathcal{C}, \text{any chain } \mathcal{C}' \text{ found in } \text{RECEIVE}())$
  if INPUT() contains READ **then**
    write $R(\tilde{\mathcal{C}})$ to OUTPUT() ▷ Produce necessary output before the POW stage.
  end if
  $\langle st, x \rangle \leftarrow I(st, \tilde{\mathcal{C}}, round, \text{INPUT}(), \text{RECEIVE}())$ ▷ Determine the x-value.
  $\mathcal{C}_{\mathsf{new}} \leftarrow \mathsf{pow}(x, \tilde{\mathcal{C}})$
  if $\mathcal{C} \neq \mathcal{C}_{\mathsf{new}}$ **then**
    $\mathcal{C} \leftarrow \mathcal{C}_{\mathsf{new}}$
    DIFFUSE($\mathcal{C}$) ▷ Broadcast the chain in case of adoption/extension.
  else
    DIFFUSE($\perp$) ▷ Signals the end of the round to the diffuse functionality.
  end if
  $round \leftarrow round + 1$
end while

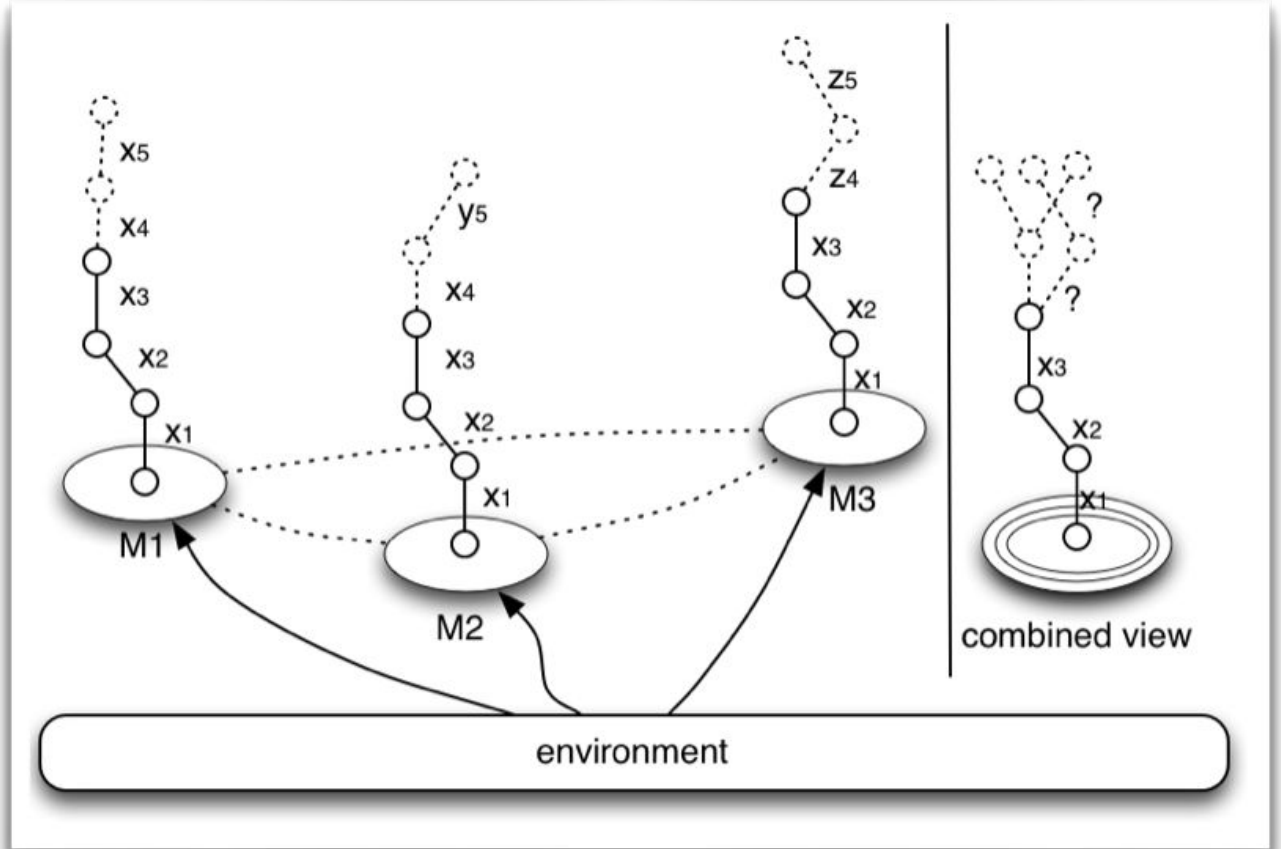
Algorithm 3 The *proof of work* function, parameterized by q, T and hash functions $H(\cdot), G(\cdot)$. The input is $(x, \mathcal{C})$.

1: **function** pow$(x, \mathcal{C})$
2: **if** $\mathcal{C} = \varepsilon$ **then** ▷ Determine proof of work instance
3: $s \leftarrow 0$
4: **else**
5: $\langle s', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$
6: $s \leftarrow H(ctr', G(s', x'))$
7: **end if**
8: $ctr \leftarrow 1$
9: $B \leftarrow \varepsilon$
10: $h \leftarrow G(s, x)$
11: **while** $(ctr \leq q)$ **do**
12: **if** $(H(ctr, h) < T)$ **then** ▷ This $H(\cdot)$ invocation subject to the q-bound
13: $B \leftarrow \langle s, x, ctr \rangle$
14: **break**
15: **end if**
16: $ctr \leftarrow ctr + 1$
17: **end while**
18: $\mathcal{C} \leftarrow \mathcal{C}B$ ▷ Extend chain
19: **return** $\mathcal{C}$
20: **end function**

Basic Properties

- Common Prefix
- Chain Quality
- Chain Growth

Common Prefix, I



Common Prefix, II

(strong common prefix / consistency)

$$\forall r_1, r_2, (r_1 \leq r_2), P_1, P_2, \text{ with } \mathcal{C}_1, \mathcal{C}_2 : \mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$$

- The property holds true in a probabilistic sense with an error that decays exponentially in *k*

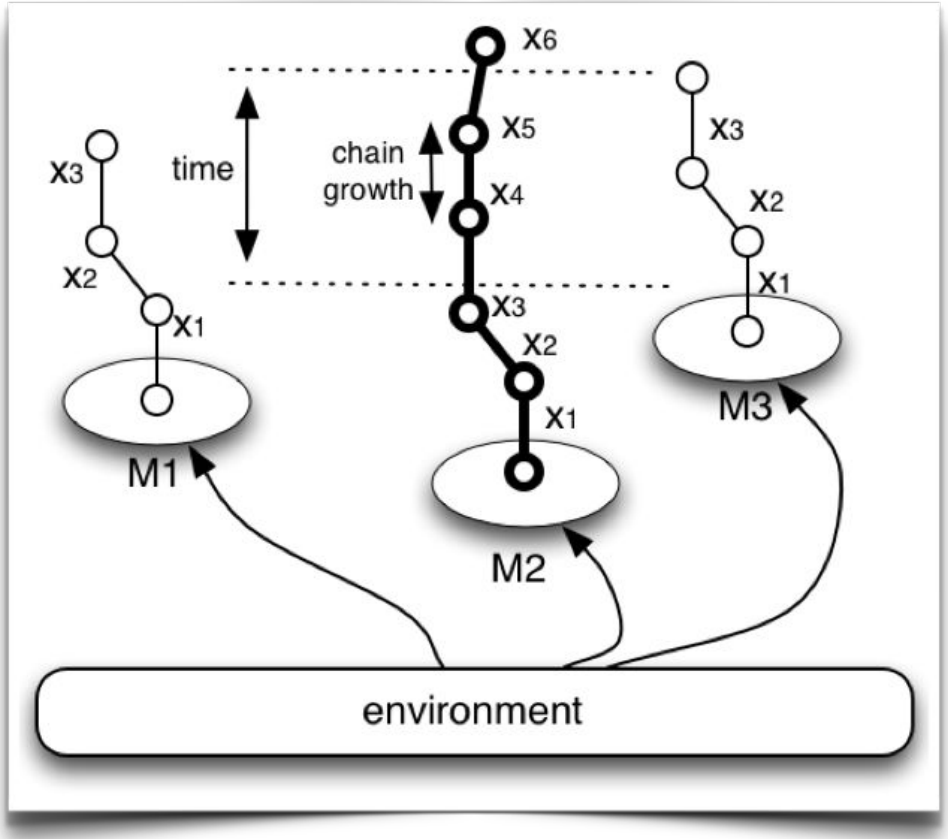
Racing Attacks

Attacher splits from the main chain and tries to overtake the “honest chain”

=> Common prefix breaks

Intuition why the attack is a small probability event: concentration bounds help honest parties.

Chain Growth, I



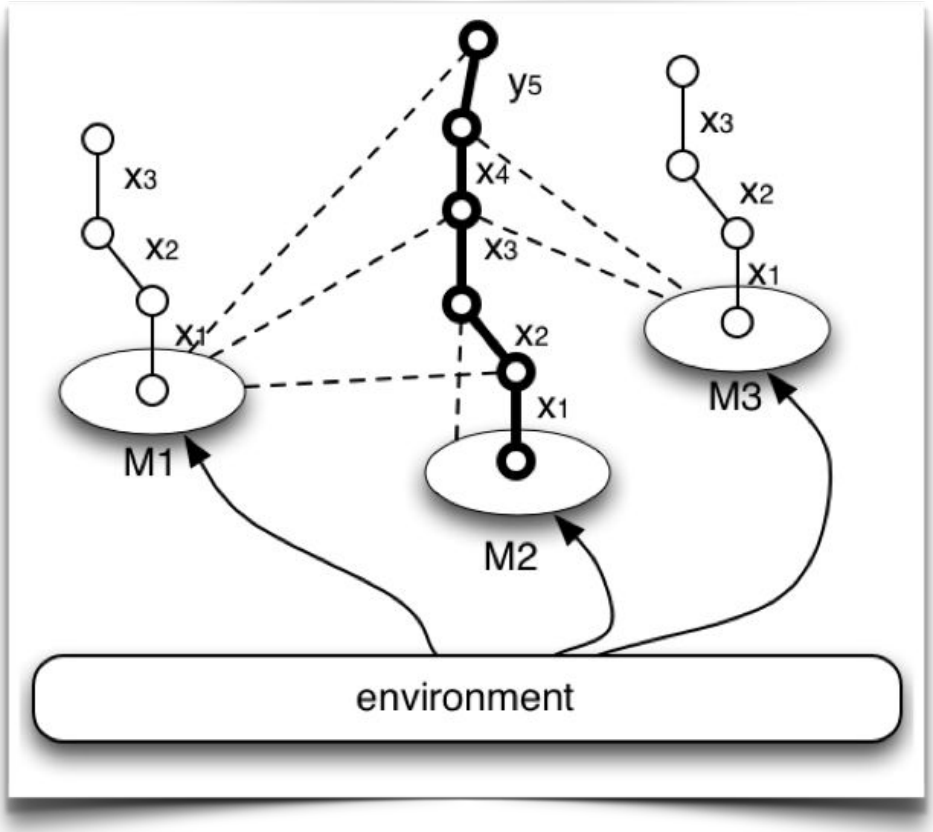
Chain Growth, II

Parameters $\tau \in (0,1), s \in \mathbb{N}$
In any period of s rounds at least τs blocks are added to the chain of an honest party P.

- The property holds true in a probabilistic sense with an error probability that exponentially decays in *s*

$\tau \approx$ probability at least one honest party finds a POW in a round

Chain Quality, I



Chain Quality, II

Parameters $\mu \in \{0,1\}, \ell \in \mathbb{N}$
The ratio of blocks of an ℓ-long segment of an honest chain produced by the adversary is bounded by $(1-\mu)\ell$

- The probability holds true probabilistically with an error that exponentially decays in ℓ

$$\mu \approx \frac{n-2t}{n-t}$$

Block withholding attack

Attacker mines privately and releases block at the time an honest party releases its own block.

Assuming honest propagation favours the adversary, the honest party block is dropped reducing chain quality

Still over time the adversary may not have enough blocks to completely eliminate chain quality.

Robust Transaction Ledger - Ledger Consensus

- It can be shown that the three properties can provide a ledger with the following characteristics.
  - persistence: Transactions are organized in a “log” and honest nodes agree on it.
  - liveness: New transactions are included in the log nodes, after a suitable period of time.

Establishing a RTL from a Blockchain

- Persistence follows from (strong) Common Prefix.
  - (need to exclude *k* most recent blocks)
- Liveness from Chain Growth and Chain Quality.
  - (leave sufficient time for chain to grow and then apply chain quality to ensure that at least one honest block is included).

Ledger Consensus vs. Consensus

- What is the connection?
  - ledger is an ever-going protocol with inputs (e.g., transactions) continuously coming from (also) external sources.
  - Consensus is a one-shot execution.
- Is it possible to reduce consensus to the ledger? Is it possible to reduce the ledger to consensus?

The Bitcoin Setting for Consensus

Sometimes also referred to as the “Permissionless” setting.

- The bitcoin setting is a different compared to what has been considered classically for the consensus problem.
  - Communication is by **diffusion**. (no point-to-point channels).
    - Message delivery is assumed but message origins and recipient list are not specified.
  - The protocol setup is not a private correlated setup (digital signatures are not used to authenticate miners)
    - A public setup is assumed (in the form of the genesis block)

Consensus <= Ledger, I

- Using a ledger (or directly the underlying blockchain) to solve consensus.
- Let’s focus on *binary* consensus for simplicity.

Consensus <= Ledger, II

Re: Bitcoin P2P e-cash paper

Satoshi Nakamoto | Thu, 13 Nov 2008 19:34:25 -0800

James A. Donald wrote:
> It is not sufficient that everyone knows X. We also
> need everyone to know that everyone knows X, and that
> everyone knows that everyone knows that everyone knows X
> - which, as in the Byzantine Generals problem, is the
> classic hard problem of distributed data processing.

The proof-of-work chain is a solution to the Byzantine Generals' Problem. I'll try to rephrase it in that context.

A number of Byzantine Generals each have a computer and want to attack the King's wi-fi by brute forcing the password, which they've learned is a certain number of characters in length. Once they stimulate the network to generate a packet, they must crack the password within a limited time to break in and erase the logs, otherwise they will be discovered and get in trouble. They only have enough CPU power to crack it fast enough if a majority of them attack at the same time.

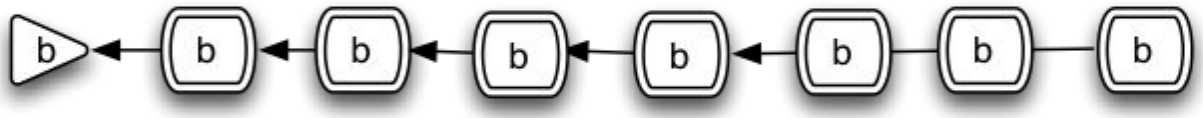
They don't particularly care when the attack will be, just that they all agree. It has been decided that anyone who feels like it will announce a time, and whatever time is heard first will be the official attack time. The problem is

https://www.mail-archive.com/cryptography@metzdowd.com/msg09997.html

Consensus <= Ledger, III

Nakamoto's suggestion

Valid chains have a single bit

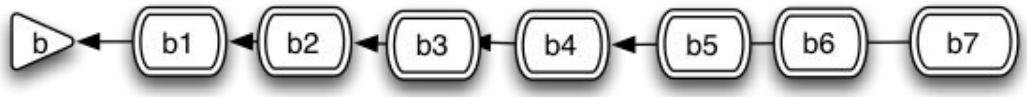


It works .. but only with constant probability of success
(not overwhelming)

Consensus <= Ledger, IV

A (1/3) “consensus protocol” (GKL protocol 1)

Valid chains have different bits with each block containing the bit of the block producer



It works .. but only up to 1/3 adversarial power.

Consensus <= Ledger, V

- Main obstacle (intuitively)
  the blockchain protocol does not provide sufficiently high chain quality.
- … we cannot guarantee that we have enough blocks originating from honest parties.
- How to fix this?

Consensus <= Ledger, VI

- The *n* parties build a ledger but now generate transactions based on POW that contain their inputs.
- Once the blockchain is long enough the parties' prune the last *k* blocks and output the majority of the values drawn from the set of transactions in the ledger.

Beware! given that POW's are used for two different tasks how do we prevent the attacker from shifting its hashing power from the one to the other?

2-for-1 POW

$h \leftarrow G(s, x)$
if $H(h, ctr) < T \ldots$

$h' \leftarrow G(s', x')$
if $H(h', ctr') < T' \ldots$

given $((s, x), ctr)$
verify:
$H(G(s, x), ctr) < T$
given $((s', x'), ctr')$
verify:
$H(G(s', x'), ctr') < T'$

Not Secure

parallel composition of POW protocols

$h \leftarrow G(s, x)$
$h' \leftarrow G(s', x')$
$w \leftarrow H(h, h', ctr)$
if $w < T \ldots$
if $[w]^{\mathsf{R}} < T' \ldots$

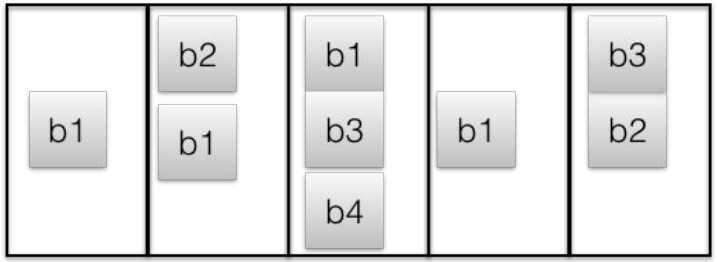
given
$((*, *), (s', x'), ctr')$
verify:
$[H(G(*, *), G(s', x'), ctr')]^{\mathsf{R}} < T'$

Key Lemma

- **Lemma.** Finding POW solution for either “side” of the POW protocol is an independent event.
  - [note that it works only for a suitable choice of T and T’]

GKL Consensus Protocol #2

parties **mine** POWs for each block (as in bitcoin backbone)



parties **mine** POWs for their input in {0,1} (input+"nonce")

they keep **transmitting** POW-inputs, until they are accepted.

Finally, after the blockchain **grows sufficiently**,
they **chop** the last *k* blocks and return the **majority among unique inputs** in the (common) prefix.

Why it works:

Key proof idea

recall : the output of the protocol is the **majority** bit from the inputs in the **common prefix**

The **(minuscule) chain quality** of the protocol, given parties transmit inputs until they are accepted, it guaranteeing that they **will be included** eventually.

Moreover, because each input, has a POW, the **majority** of unique POW-inputs will be **originating from honest parties** in any **sufficiently long** part of the chain (such as the common prefix)

Ledger <= Consensus

- (Potentially) simpler:
  - run consensus for each set of transactions (blocks) with each party proposing a set of transactions.
  - plain **validity** is insufficient.

Dynamic Availability

So far we dealt with the case where *n* nodes maintain the blockchain.

However this number may change over time: new users enter the system while existing users may go away.

The change over time can be dramatic !

The Bitcoin Blockchain handles this by adjusting the difficulty of the proof of work algorithm.

Recall: PoW algorithm

```
int counter;
counter = 0
        while Hash( block_header, counter) > Target
increment counter
```

block_header contains a coinbase transaction which contains an extraNonce parameter

if transactions remain the same extraNonce has to be modified to avoid repeating work

Targe Difficulty over time

- Initially required approximately 2^{32} hashing operations.

- Currently, [Oct.2017], 2^{72} hashing operations
    - [Oct.2018], 2^{74} hashing operations

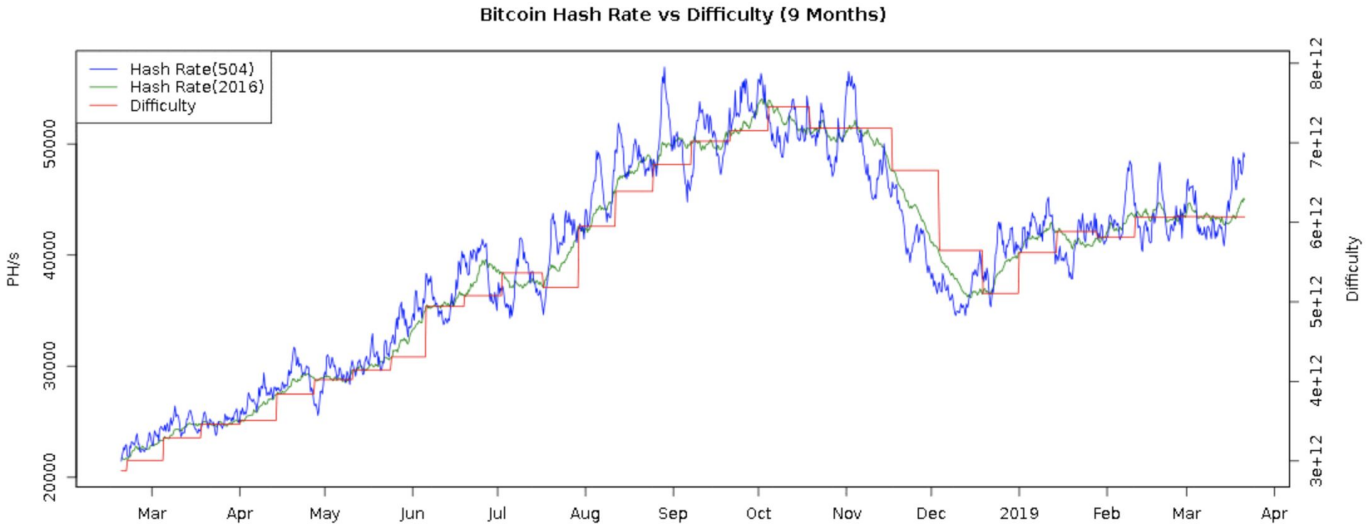
https://bitcoinwisdom.com/bitcoin/difficulty

Target difficulty Now

Bitcoin Difficulty:	6,068,891,541,676
Estimated Next Difficulty:	6,341,537,193,069 (+4.49%)
Adjust time:	After 389 Blocks, About 2.5 days
Hashrate(?):	48,835,562,309 GH/s
Block Generation Time(?):	1 block: 9.2 minutes 3 blocks: 27.4 minutes 6 blocks: 54.9 minutes
Updated:	14:45 (4.3 minutes ago)

Difficulty: 6068891541676 BTC/USD: 4122.415731

1000000	KH/s	1.726e-9	BTC/hour	0.000007117	USD/hour
1000	MH/s	4.143e-8	BTC/day	0.0001708	USD/day
1	GH/s	2.9e-7	BTC/week	0.001196	USD/week
0.001	TH/s	0.000001243	BTC/month	0.005124	USD/month



Hash operations

- Consider a regular PC that can do 30 MHash / sec
- With expectation of 2^{74} hashing operations, mining a block will require ~ **20 million years**.

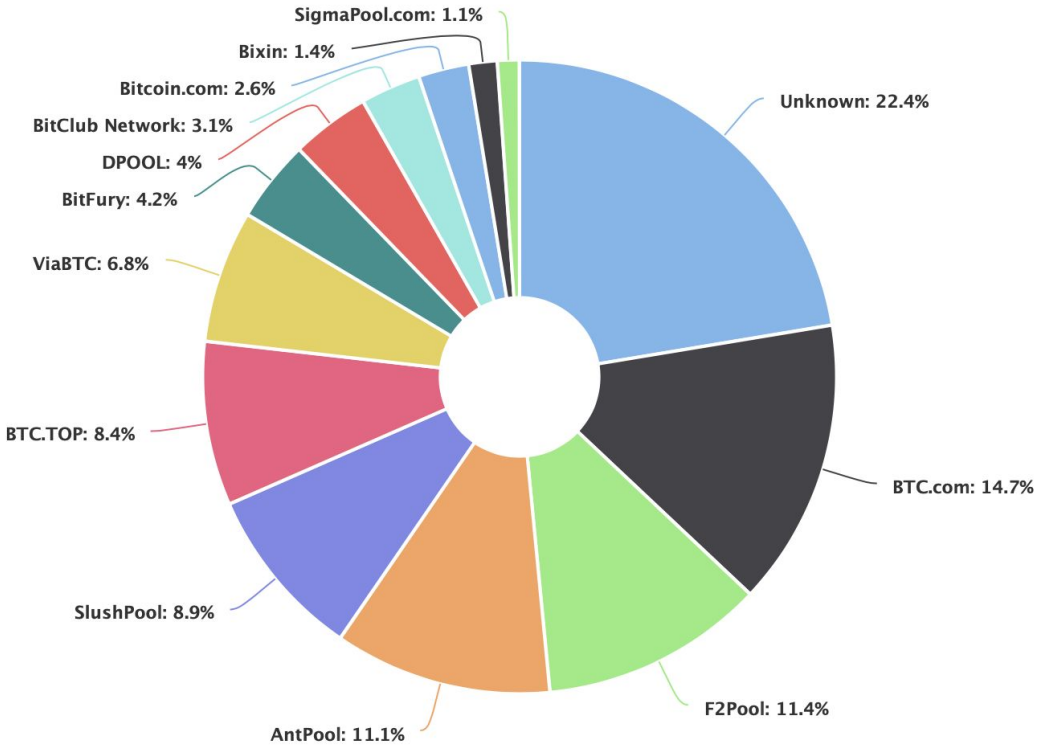
https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison

Parallelising mining

- Bitcoin’s Proof of Work can be parallelized
  - Mining pools
    - Instead of working separately, work **together** to solve the same block.
    - By collecting **“shares”** (small hashes of the block that are not quite as small as needed) one can prove how much they contributed.

https://www.blockchain.com/pools

Current bitcoin mining pools



Adjusting the difficulty

“maxvalid” rule
is changed so that
parties adopt chain with highest difficulty
linearly related to

$$\sum_i \frac{1}{T_i}$$

The *f* parameter [GKL15]

f = probability of producing a block in a round of interaction
(depends on target *T,* # of miners *n, and duration of round*)

- If *f* becomes too small, parties do not do progress; chain growth becomes too slow. [liveness is hurt]

- if *f* becomes too large, parties "collide" all the time; an adversary, exploiting network scheduling, can lead them to a forked state. [persistence is hurt]

To resolve this in a dynamic environment, bitcoin **recalculates the target** *T* to keep *f* constant

Target recalculation

$n_0 =$ estimation of the number of ready parties at the onset

$T_0 =$ initial target

[recall in this context "party"= single CPU]

$m =$ epoch length in blocks (In Bitcoin) = 2016

$\tau =$ recalculation threshold parameter (In Bitcoin) = 4

$T =$ target in effect

$pT =$ prob of a single miner getting a POW in a round

$$\text{next target} = \begin{cases} \frac{1}{\tau} \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 < \frac{1}{\tau} \cdot T; \\ \tau \cdot T & \text{if } \frac{n_0}{n} \cdot T_0 > \tau \cdot T; \\ \frac{n_0}{n} \cdot T_0 & \text{otherwise} \end{cases}$$

$\Delta =$ last epoch duration based on block timestamps

$$n = \frac{m}{pT\Delta}$$ the "effective" number of parties of the epoch

Bahack's attack

- The recalculation threshold is essential.
  - Without it, an adversary can create a private, artificially difficult chain that will increase the variance in its block production rate; overcoming the chain of the honest parties becomes a non-negligible event.

Understanding the attack : clay pigeons

clay pigeons

Clay pigeon shooting game

- Suppose you shoot on targets successively from 10m against an opponent
  - your success probability 0.3 vs. 0.4 that of your opponent.
  - You shoot in sequence 1000 targets. The winner is the one that got the most hits.
- What is your probability of winning?

Chernoff Bounds

Let

$$\delta > 0, \mathbf{Prob}[X_i = 1] = p_i, \mu = \sum_{i=1}^{n} p_i$$

Then

$$\mathbf{Prob}[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/(2+\delta))$$

$$\mathbf{Prob}[\sum_{i=1}^{n} X_i \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2), \delta \in (0,1)$$

Analysis, I

- You have an expectation of 300 hits and your opponent has an expectation of 400 hits.
- What is your probability of winning?
- Denote by X, whether you hit a target, and similarly Y for your opponent. From Chernoff bounds

$$\mathbf{Pr}[\sum_{i=1}^{1000} X_i \geq 345] \leq \exp(-(0.15)^2 300/2.15) < 4.3\%$$

$$\mathbf{Pr}[\sum_{i=1}^{1000} Y_i \leq 348] \leq \exp(-(0.13)^2 400/2) < 3.5\%$$

Analysis, II

- If the negation of both these events happens you will certainly loose

$$\mathbf{Pr}[X_{<345} \wedge Y_{>348}] = (1 - \mathbf{Pr}[X_{\geq 345}])(1 - \mathbf{Pr}[Y_{\geq 348}]) \geq 92.3\%$$

- Thus the probability of you winning is below 8%

Analysis, III

- Now you are given a choice: you can decrease the size of the clay pigeon target by a ratio β and augment your “kills” by multiplying with 1/β.
- Suppose your accuracy is just linear with β.
  - do you accept to play like this (while your opponent will keep playing in the same way) ?

Analysis, IV

Each shot has success $\mathbf{Pr}[X'_i = 1] = \beta \cdot \mathbf{Pr}[X_i = 1]$

- The score expectation of each shot remains the same:
  $E[(1/\beta)X'_i] = (1/\beta)\beta E[X_i] = E[X_i]$

$$\mathbf{Pr}[\sum_{i=1}^{1000} X'_i \geq 345\beta]$$

$$\leq \exp(-(0.15)^2 300\beta/2.15)$$

decreasing β results in increased variance and our previous concentration argument will fail

β	bound
1,	~4.3%
0.5,	~20.8%
0.25,	~45.6%
0.10,	~73.1%