# IOP on its Way to a New Consensus

Traditional BFT, DAGs and virtual voting

## The Current Situation

Most consensus mechanisms based on one or more Sybil prevention ideas:

- Proof of Work
- Proof of Stake
- Delegated Proof of Stake

IOTA, NANO, etc. aren’t fully self-sufficient

## Proof of Work

Bitcoin is running at 35 million Th/s

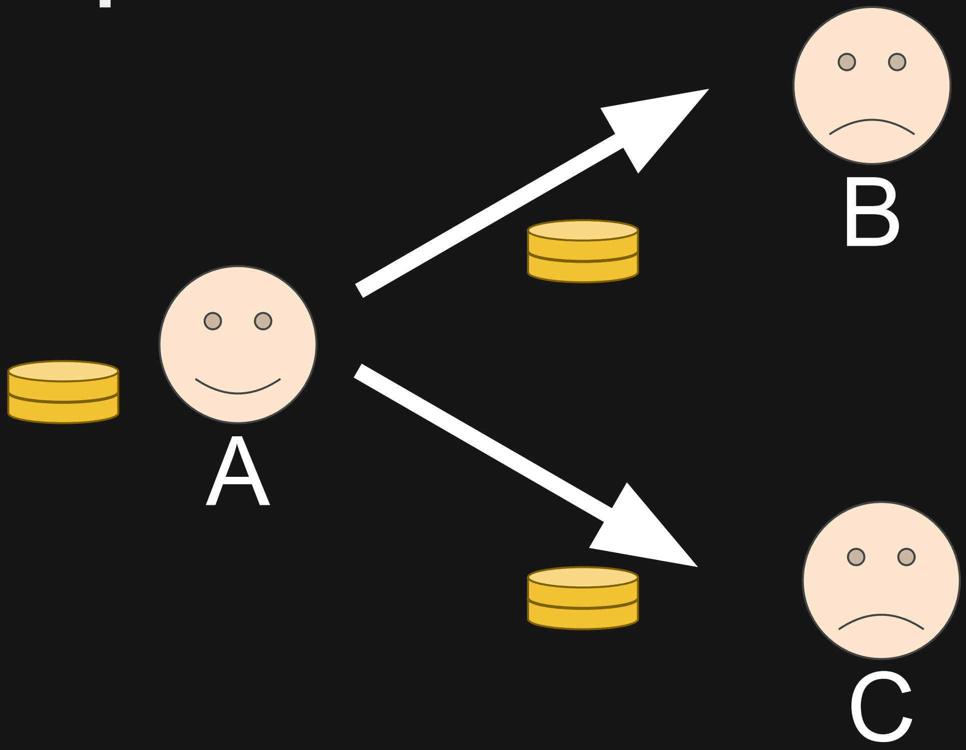
Yearly energy consumption of 71.12 billion kWh

At 5ct/kWh that is 3.5 billion USD in electricity cost

All a big waste?

## Proof of Work

## Double Spend



## Proof of Work

Artificial slowdown of the network

Time between blocks must be much larger than network latency so that honest users can collaborate and coordinate efforts

Miners decide about:

- Consensus changes
- Protocol updates

Longest chain always wins → No true consensus

Small but finite possibility of a rollback → The blockchain is not immutable!

## Proof of Stake

Probability to be selected as forger of the next block

$$\propto$$

Amount of coins in wallet

Staking on multiple concurrent chains

“Nothing at stake” needs to be addressed using some form of slashing mechanisms

## Delegated Proof of Stake

Coin owners vote on representatives

Representatives cooperate by taking turns → Proof of Authority

Very decentralized and fast in theory

Next forgers are known long before it’s their turn to create blocks
→ dDoS countermeasures necessary

Voter apathy, a “few party system”, bribing, threats, …

## A better system

Every system is designed

Paradox: We try to make up strict rules to create a free economy

Theoretical proofs of feasibility are not the end, but the beginning

Hydra does egalitarian, provably random and “hidden” round-robin block proposal

IOP does Directed Acyclic Graphs

## The Byzantine Generals

*A group of generals, each commanding a part of the byzantine army, encircle a city. The generals want to formulate a plan to attack the city. In its simplest form they need to decide whether to attack or retreat. Some of them may prefer to attack, while others prefer to retreat. The important thing is that the generals must agree on a common decision, for a halfhearted attack by only a few generals would become a rout, and would be worse than either a coordinated attack or a coordinated retreat.*

*The problem is complicated by the presence of treacherous generals who may not only cast a vote for a suboptimal strategy, they may do so selectively. For instance,if nine generals are voting, four of whom support attacking while four others are in favor of retreat, the ninth general may send a vote of retreat to those generals in favor of retreat, and a vote of attack to the rest. Those who received a retreat vote from the ninth general will retreat, while the rest will attack (which may not go well for the attackers). The problem is complicated further by the generals being physically separated and having to send their votes via messengers who may fail to deliver votes or may forge false votes.*

From *Byzantine Fault Tolerance,* Wikipedia

## The Byzantine Generals

Each solution must guarantee:

1. Liveness
2. Safety or Agreement
3. Non-triviality or Consistency

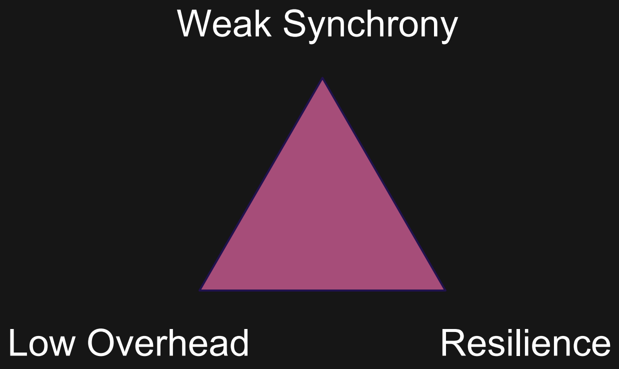
Cryptography allows us to relax assumption about forgeable messages.

An algorithm becomes  extremely powerful by introducing  “coin tosses”

## Byzantine Fault Tolerance

Let **n** be the Number of Generals and **t** be the number of Byzantine Nodes

Three opposing requirements:



## BBA*

Proposed by Micali at CSAIL, MIT, March 19, 2018

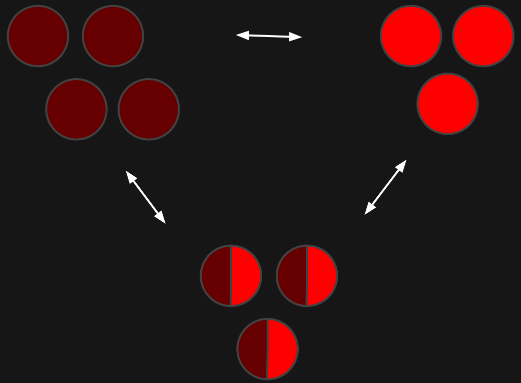
Very Simple algorithm

Works for $n = 3t + 1$

Assumes synchrony

Uses “common coin” and cryptographic signatures

$O(n^2)$ messages per step, with expected 9 steps



## BBA*

Idealized Protocol

1. Every player sends his estimate to everyone, including himself
2. A random bit **c** appears in the sky
3. Every player counts the estimates and chooses his strategy  as follows:
    a. If **$\#(0) > 2t + 1$**, estimate 0 next round
    b. If **$\#(1) > 2t + 1$**, estimate 1 next round
    c. Else change your estimate to c

We come to an agreement in the next round with probability ½

Random bit like this is called “magic coin”

## Concrete Coins

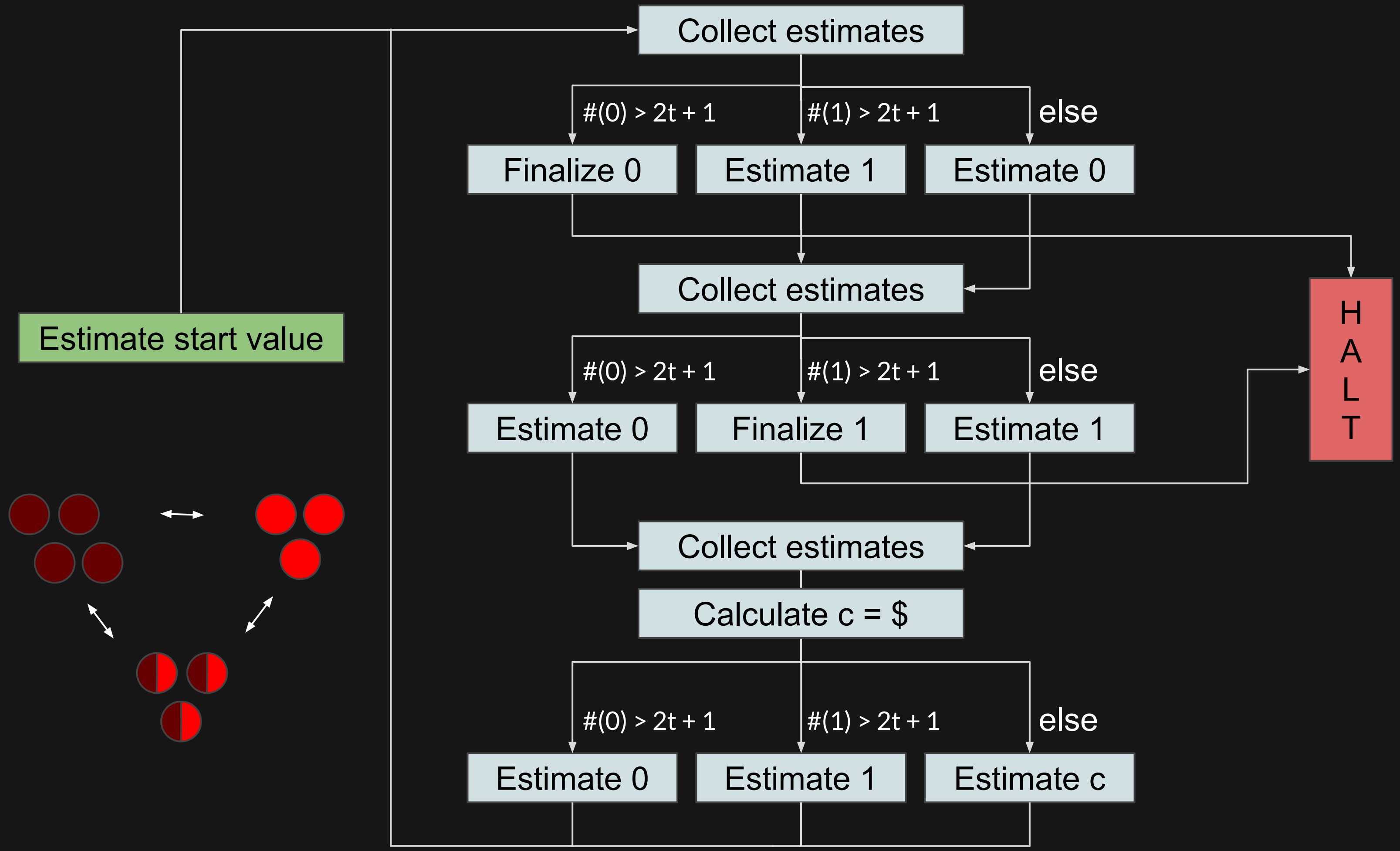
We don’t need perfect coins

If the coin is “magic” with probability ⅔, we come into into agreement with probability ⅓

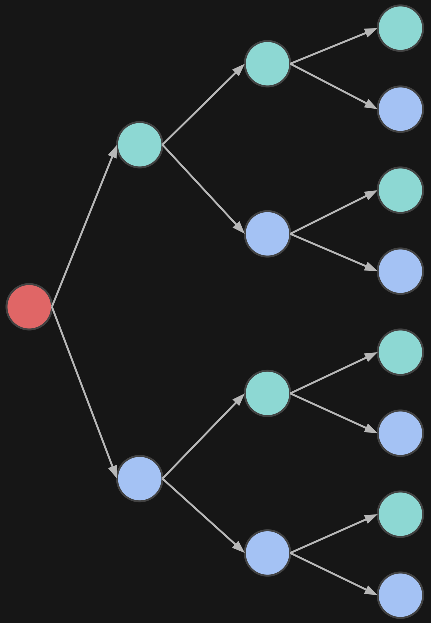
Let R be some previously agreed on value (e.g. public network ID in the HYD protocol):

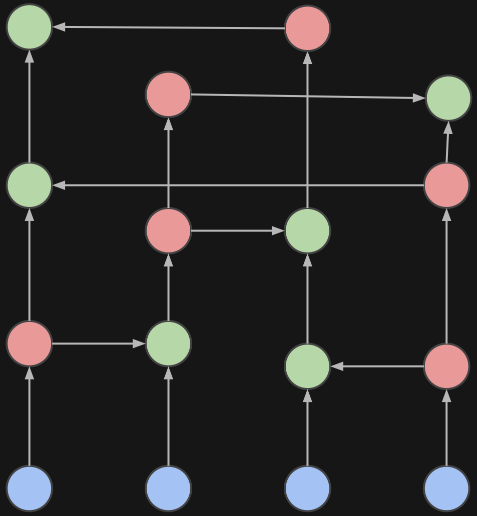
1. Every player signs R, call his signature v
2. Find the player who has the numerically lowest hash value for his signature
3. Let c be the least significant bit of this player’s signature

c is now a random coin with probability >⅔


Estimate start value
Collect estimates
#(0) > 2t + 1
#(1) > 2t + 1
else
Finalize 0
Estimate 1
Estimate 0
Collect estimates
#(0) > 2t + 1
#(1) > 2t + 1
else
Estimate 0
Finalize 1
Estimate 1
Collect estimates
Calculate c = $
#(0) > 2t + 1
#(1) > 2t + 1
else
Estimate 0
Estimate 1
Estimate c
HALT

# Directed Acyclic Graphs





## Introducing DAGs to BFT

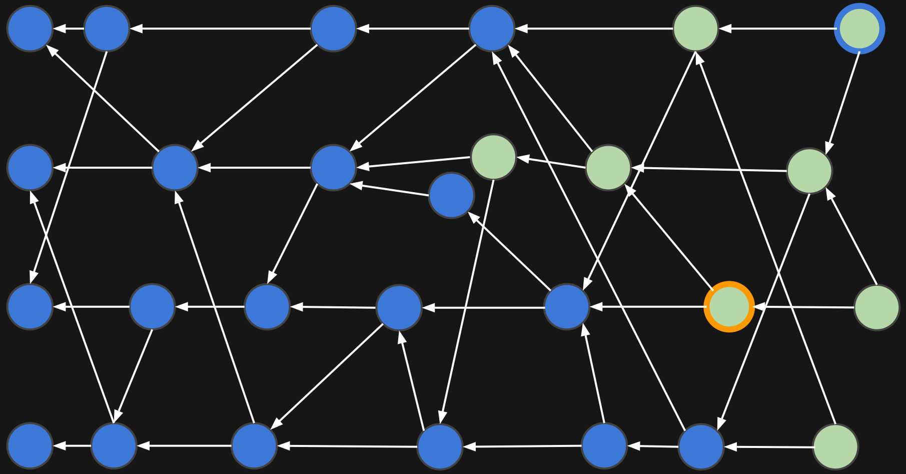
Every message I send

- is signed → No one can forge my messages
- references my previous message → No one can selectively hide some of my messages
- references the last message I received → No one can selectively hide some users

DAG formed by the parent-child relationship between messages

If every message contains on average 1 transaction, the number of messages sent is $O(n)$

# The Gossip Graph

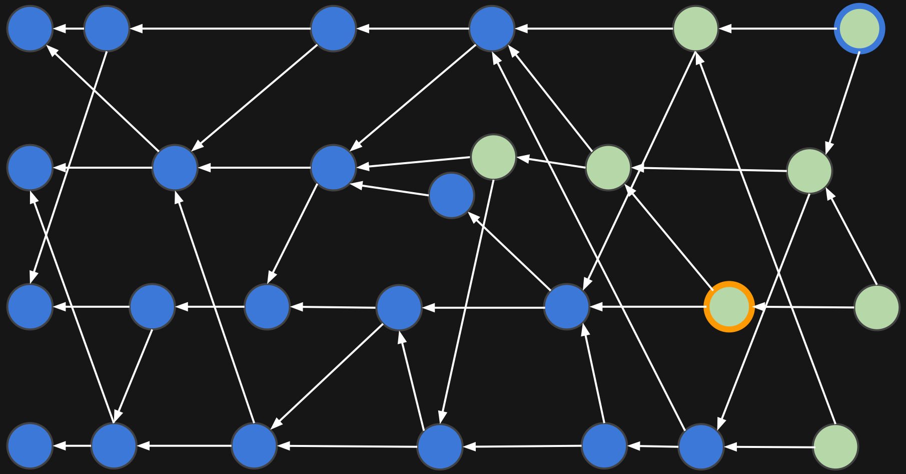


Event A is an *ancestor* of Event B if A = B or A is an ancestor of a parent of B.

Two events X and Y form a *fork* if they are created by the same member but neither is an ancestor of the other.

Event A *sees* Event B if B is an Ancestor of A and there are no forks from the creator of B in the ancestors of A.

## The Gossip Graph



Event A *strongly sees* Event B if A sees B and A sees Events by more than ⅔ of all creators that each see B.

Two graphs are consistent if for any event X they contain the same ancestors of X with the same parent-edges.

If (X,Y) is a fork and X is strongly seen by an Event A in a graph G, then there are no events in any consistent graph H that strongly see Y.

## Virtual Voting

Strongly seen events are reliable indicators of another members view of the gossip graph

Using strongly seen events, we can calculate what another member of the network would have answered if we asked him a question about the transaction history without actually asking.

→ Consensus mechanism without additional overhead

Algorithms already proposed: Swirlds Hashgraph (Hedera), PARSEC (MaidSAFE)

## First Prototype

Written in Python

Aim: Compare and/or combine multiple strategies (Graph traversal is non-trivial problem)

Nodes are separate, independent processes
→ when computational feasibility is established, extension to non-local testing with minimal effort

Color code:

- Fresh events
- Strongly seen events
- Virtual votes

## First Prototype

Current state:

- Refusal of inconsistent information
- Two algorithms for “strongly seen” events
- Casting of virtual votes for transactions

Still missing:

- Collecting votes and deciding on order of events
- Addition and removal of nodes on the fly

## Other Algorithms

Swirlds (Leemon Baird):

- Closed Source, Patented
- Closed network
- Completely asynchronous
- Orders events in “bunches”
- Fully virtual

PARSEC:

- Open Source (GPLv3)
- Ostensibly open network
- Assumes weak synchrony
- Orders events one after the other
- Message sending for concrete coins and blockvotes

The ultimate goal: Fully virtual algorithm which orders events sequentially, while being completely asynchronous, running on a variable set of nodes

## Possible Incentives on a DAG

Members of the network are competing in terms of network latency and throughput → Optimal hardware is multi-purpose, competition is directly enhancing network quality

Leading participants will be taking part in consensus, runner-ups will stand by to replace non-responding nodes on the fly.

Rewards can be calculated based on “quality of service”, taking into account number of connections, median latency, and “age” of a node, because relevant information is verifiable “on graph”.

Thank You!