Braiding the Blockchain

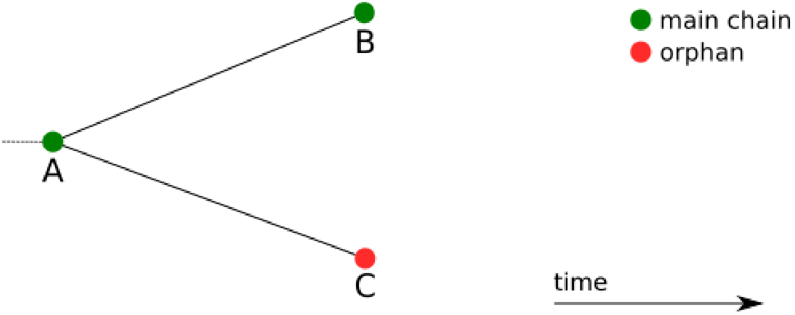
Bob McElrath, Ph.D.

Scaling Bitcoin
Hong Kong
Dec 7 2015

bmcelrath@sldx.com

Save the Orphans!

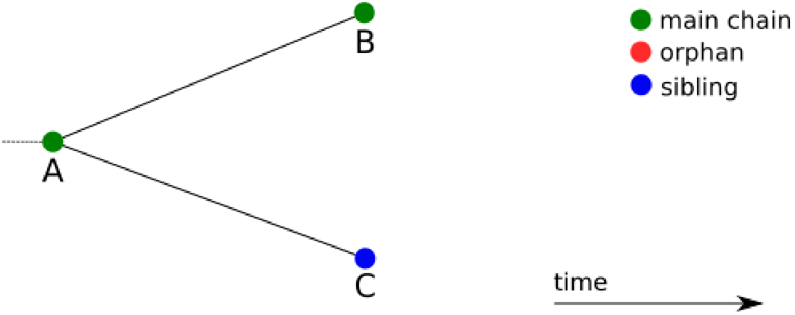
Orphans are *not* a necessary component of the operation of bitcoin!



- Orphans occur when miners do not know about the existence of another block (B) before generating theirs (C)
- Simultaneous block generation is *unavoidable*
- It doesn't require us to deprive a miner of profit

Save the Orphans!

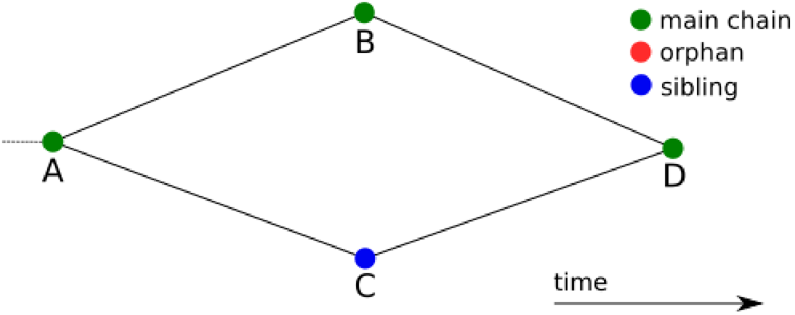
Orphans are *not* a necessary component of the operation of bitcoin!



- What if block C contains no conflicting transactions?
- What if block C contains a duplicate transaction?
- *There is no conflict* so let us call C a *sibling*.

Save the Orphans!

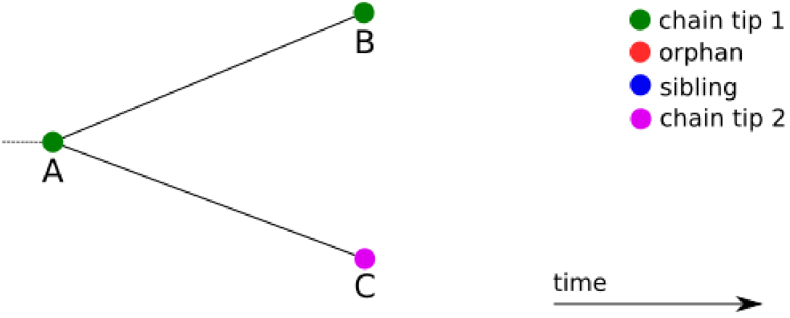
Orphans are *not* a necessary component of the operation of bitcoin!



- A future block must be able to tie up B and C, indicating that there is no conflict.
- To get rid of orphans, *blocks must have multiple parents*

Save the Orphans!

Orphans are *not* a necessary component of the operation of bitcoin!



- If C contains a double-spend relative to B, then C forms a new chain tip.

Down the Rabbit Hole

- Allowing blocks to have multiple parents creates a data structure called a *Directed Acyclic Graph*.
- What if I just throw out blocks as fast as I desire, do algorithms exist that could make sense of the chaos and define a highest work "tip"?
- The blockchain is an over-simplified data structure, with some unfortunate consequences (orphans, selfish mining).

Down the Rabbit Hole

- Allowing blocks to have multiple parents creates a data structure called a *Directed Acyclic Graph*.
- What if I just throw out blocks as fast as I desire, do algorithms exist that could make sense of the chaos and define a highest work "tip"?
- The blockchain is an over-simplified data structure, with some unfortunate consequences (orphans, selfish mining).

Down the Rabbit Hole

- Allowing blocks to have multiple parents creates a data structure called a *Directed Acyclic Graph*.
- What if I just throw out blocks as fast as I desire, do algorithms exist that could make sense of the chaos and define a highest work "tip"?
- The blockchain is an over-simplified data structure, with some unfortunate consequences (orphans, selfish mining).

The Directed Acyclic Graph

Allowing blocks to have multiple parents creates a:

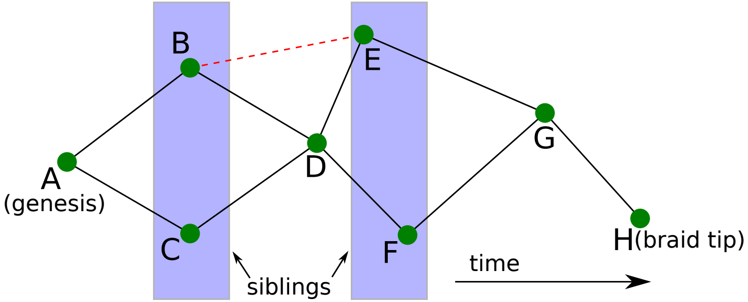
Directed Blocks have parents, parents cannot refer to children

Acyclic A cycle is cryptographically impossible

Graph Structure is non-linear (no “height”)

- A DAG can be *partial ordered* in linear time.
- We have to make a restriction relative to a more general dag, so I’m going to name this data structure a *braid*.

Braid Terminology



Braid A Directed Acyclic Graph having no *incest* (no triangles)

Bead Analog of Bitcoin's blocks (green circles)

Sibling A *bead* that cannot be partial ordered relative to myself: the pairs (B,C) and (E,F)

Incest A parent that is simultaneously an ancestor of another parent (disallowed)

Our Approach

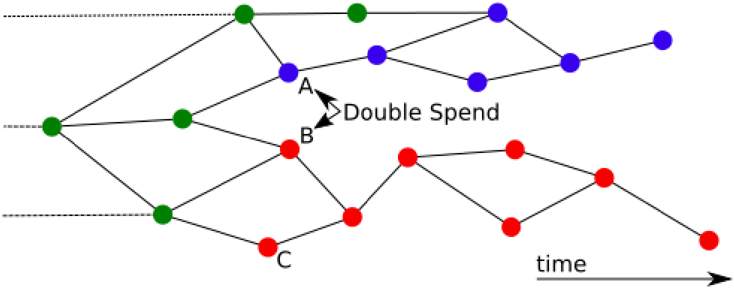
- Because of *selfish mining*[1] we will incentivize miners to quickly transmit beads
- Because GHOST[2] worsens some attacks, we will require that *parents must not contain conflicting transactions*
- Allow all of these to be decided per-node:
  - Bead time; Bead target difficulty; Bead size
- Assume Braids will be a parallel, faster layer to Bitcoin blocks:
  - Beads will be constructed such that they are valid Bitcoin blocks, if they meet bitcoin's difficulty target.
- Publish beads *ex-post-facto* because knowing who the current leader is (Bitcoin-NG[3]) opens a new vulnerability

[1] Eyal, Sirer, arXiv:1311.0243

[2] Sompolinsky, Zohar, ia.cr/2013/881

[3] Eyal, Gencer, Sirer, Renesse, arXiv:1510.02037

Braid Fork Example



- A double-spend occurs in A and B
  - $\Rightarrow$ We must evaluate which braid has the most work
- Beads in each fork reference either A or B as a parent
- The highest-work braid will be decided by evaluating the work in the combined work of all beads in the red and blue subgraphs.

How to Incentivize Miners

Miner incentives must be aligned with correct operation of the network

- Consensus is *created* by the profit-maximizing miners
- Let the *reward* be proportional to a miner's target difficulty
  - ⇒ Miners can individually choose target and block rate based on *other considerations* (e.g. bandwidth)
  - ⇒ Bandwidth and CPU is now the only limiting factor for the network!
- The existence of siblings/orphans means *we cannot decide miner coin allocation until all beads are seen by all nodes*
  - ⇒ Coin allocation will be calculated 100 blocks later

We will define several quantities that we will use in a new miner incentive formula (a.k.a. How many bitcoins do I get?)

Siblings

A sibling S is an analog to Bitcoin *orphans*. It is defined as

A bead that cannot be ordered to come before or after mine using only the DAGs partial order

- Siblings are defined *per braid tip*
- Siblings *must not* contain conflicting transactions
- Siblings *may* contain duplicate transactions

If siblings share the same transaction, each sibling will be allocated a work-weighted fraction of the tx fee. (e.g. 2 siblings at the same target difficulty will each recieve 1/2 of the tx fee)

Cohort Difficulty

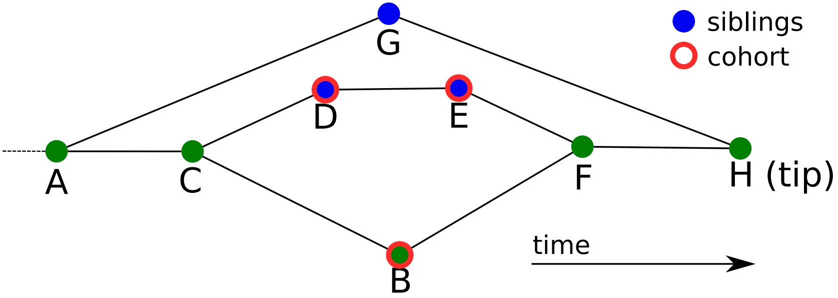
The *cohort difficulty* D is the work of other miners during the time window in which I was mining. It is defined as:

The combined work of all beads between
my youngest parent and my oldest child.

- A miner with large cohort difficulty relative to his own is playing games or following a perverse incentive
  - Trying to steal fees by becoming everyone's sibling
  - Withholding blocks (children are late)

$\Rightarrow$ Incentivizing small cohort difficulty incentivizes fast block transmission

Cohorts and Siblings



The cohort of bead B is (D,E,G) while its siblings are (D,E)

Quiz:

- Siblings of G? Cohort of G?
- Siblings of D? Cohort of D?

Miner Incentive Formula

The miner of block i receives a reward R_i:

$$R_i = \sum_t^{T_i} f_t \frac{d_i}{D_i}\left(\frac{1}{N_t}\right) + C\frac{d_i}{D_i}\left(\frac{1}{N}\right); \qquad N_t = \sum_s^{S_t} \frac{d_s}{D_s}; \qquad N = \sum_j^{N_c} \frac{d_j}{D_j}$$

i,j	bead indices
t	transaction index
d_i	difficulty = 1/target
D_i	the cohort difficulty
T_i	number of transactions in bead i
C	block reward = 25 BTC
f_t	Transaction fee for tx t
S_t	number of siblings containing tx t ($S_t = 1$ if no siblings)
N_t	Sum of weighted difficulty over siblings contining tx t
N	Sum of weighted difficulty over all beads (normalization)

Miner Incentive Formula

The miner of block i receives a reward R_i:

$$R_i = \sum_t^{T_i} f_t \frac{d_i}{D_i}\left(\frac{1}{N_t}\right) + C\frac{d_i}{D_i}\left(\frac{1}{N}\right); \qquad N_t = \sum_s^{S_t} \frac{d_s}{D_s}; \qquad N = \sum_j^{N_c} \frac{d_j}{D_j}$$

This miner incentive formula is constructed such that it is:

- *linear* in miner difficulty d_i (miners set their own target)
  - For $\frac{d_i}{D_i} \ll N$ miner income is independent of target
  - Smaller d_i means *smoother* income distribution over time
- Fair (difficulty-weighted) split of fees f_t among siblings

Miner Incentive Formula

The miner of block i receives a reward R_i:

$$R_i = \sum_t^{T_i} f_t \frac{d_i}{D_i}\left(\frac{1}{N_t}\right) + C\frac{d_i}{D_i}\left(\frac{1}{N}\right); \qquad N_t = \sum_s^{S_t} \frac{d_s}{D_s}; \qquad N = \sum_j^{N_c} \frac{d_j}{D_j}$$

Consequences of this incentive structure:

- We're incentivized to *optimize the p2p topology* to quickly propagate blocks
- Use of cohort difficulty D_i incentivizes fast transmission of blocks
- Small miners can mine without joining pools: coinbase has many outputs (like p2pool)

Evaluating the Best Braid

The sub-braid containing the most work can be determined by estimating the hash rate

$$H = \sum_{i}^{\text{miners}} H_i$$

The best way to do this is using a likelihood function:

$$W_\alpha(\{H_i\}) = -\log L = -\log \prod_{i}^{\{\mathrm{x_i}\}} P_{x_i}(H_i t, k_i)$$

where $P_x(h, k)$ is the Poisson distribution

- Miners are incentivized to include all chain tips as parents because it gives the sub-dag they're mining on more work.

Confirmation Times

How do I know when a transaction is “confirmed”?

- Satoshi’s analysis still applies, and we must keep Bitcoin’s payout schedule
  - ⇒ Counting six bitcoin blocks is still resonable
- Much better analyses are possible
  - Do there exist other braid-tips with similar work?
  - What’s the ratio of work in my braid tip and the next closest?
  - Has the hash power recently changed?

We have much more data: I’d like to see a whole class of risk evaluation methods added for different users and use cases.

Confirmation Times

How do I know when a transaction is “confirmed”?

- Satoshi’s analysis still applies, and we must keep Bitcoin’s payout schedule
  - $\Rightarrow$ Counting six bitcoin blocks is still resonable
- Much better analyses are possible
  - Do there exist other braid-tips with similar work?
  - What’s the ratio of work in my braid tip and the next closest?
  - Has the hash power recently changed?

We have much more data: I’d like to see a whole class of risk evaluation methods added for different users and use cases.

Conclusions

- Gettting rid of orphans forces us to the Braid structure.
- Transaction volume is limited only by bandwidth and CPU!
- Confirmation times can be *much* faster, limited only by the propagation time to reach the entire network (e.g. the size of the Earth).
- Algorithms are more complex, but seem to all be O(N): We don't have to solve the Travelling Salesman Problem.
- Miner income becomes much smoother and more predictable
- Many ways to put this into bitcoin many incentive models: simulation and testing is necessary.
- Smaller miners do not have to use pools
  $\Rightarrow$ mining decentralization!