Braiding the Blockchain

Bob McElrath, Ph.D.

On Chain Scaling Conference
Jun 24 2015
bmcelrath@sldx.com

Why is "simple" scaling so hard?

- Bitcoin is the first Satoshi-derived coin to achieve 1500 tx/block. We're in uncharted territory.
- A block "chain" cannot tolerate multiple writers!
  - $\Rightarrow$ We must move to a data structure capable of handling *multiple, simultaneous writers*.
- The orphan rate is directly proportional to the validation time, and therefore the *block size*.
  - $\Rightarrow$ Bitcoin contains a serious *design flaw*: it assumes zero latency or equivalently that the network is *synchronous*.
- This *design flaw* leads to a serious security reduction called *selfish mining*[1]
  - $\Rightarrow$ We must ensure that miners receive equal pay for equal (Proof-of) Work.

[1] Eyal, Sirer, arXiv:1311.0243

Eight Fallacies of Distributed Computing

A set of assumptions that L. Peter Deutsch and others at Sun Microsystems originally asserted programmers new to distributed applications invariably make.

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- Topology doesn't change
- The network is secure
- There is one administrator
- Transport cost is zero
- The network is homogeneous

"These assumptions ultimately prove false, resulting either in the failure of the system, a substantial reduction in system scope, or in large, unplanned expenses required to redesign the system to meet its original goals."

Eight Fallacies of Distributed Computing

A set of assumptions that L. Peter Deutsch and others at Sun Microsystems originally asserted programmers new to distributed applications invariably make.

- The network is reliable
- Latency is zero (scaling)
- Bandwidth is infinite
- Topology doesn't change
- The network is secure
- There is one administrator
- Transport cost is zero
- The network is homogeneous

"These assumptions ultimately prove false, resulting either in the failure of the system, a substantial reduction in system scope, or in large, unplanned expenses required to redesign the system to meet its original goals."

Sources of Latency

Latency comes from:

- Validating blocks (a few seconds, today)
- Propagating blocks (bandwidth limited)
- Relay delays (thanks, Great Firewall)
- Relay delays (Tor)
- Physics (the size of the Earth and the speed of light)

It is *impossible* to remove all sources of latency.

$\Rightarrow$ We must redesign the “chain” to tolerate multiple, simultaneous writers.

Sources of Latency

Latency comes from:

- Validating blocks (a few seconds, today)
- Propagating blocks (bandwidth limited)
- Relay delays (thanks, Great Firewall)
- Relay delays (Tor)
- Physics (the size of the Earth and the speed of light)

It is *impossible* to remove all sources of latency.

$\Rightarrow$ We must redesign the “chain” to tolerate multiple, simultaneous writers.

Block size and diminishing returns

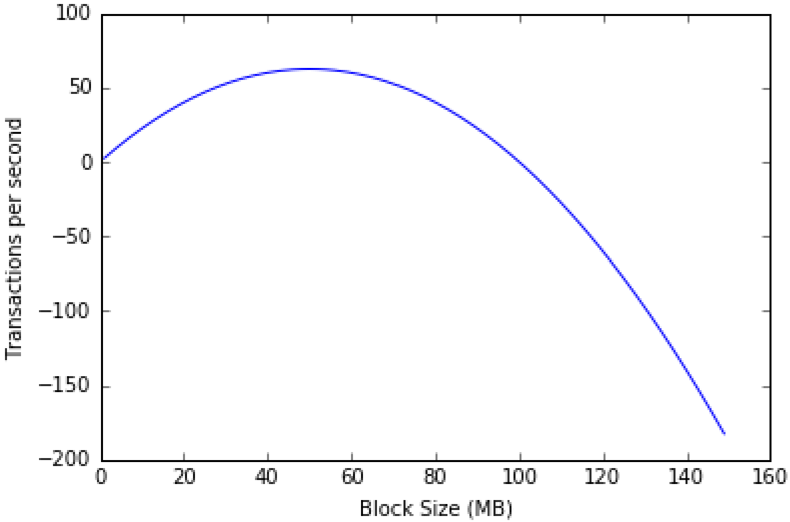
Let's assume we simply increase the block size B, resulting in an increased number of transactions per block T. We can then write the following formula for the transaction rate:

$$Rate = T\left(1 - \mathrm{S}\right) \quad (1)$$

in terms of the orphan rate (stale block fraction) S. Today with 1MB blocks this is about 1%. But this is directly proportional to the validation time. The longer a block takes to validate, the more likely another block is produced while nodes are validating and propagating it.
Therefore we can extrapolate $S = 1\% T/1500$ today and

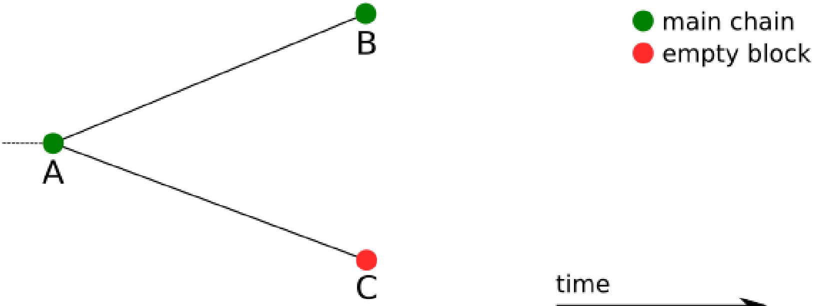
$$Rate = T\left(1 - 0.01\frac{T}{1500}\right) \quad (2)$$

Kicking the can down the road



Adding a block

In the process of adding a block, a miner must choose what to do *while validating the new block*

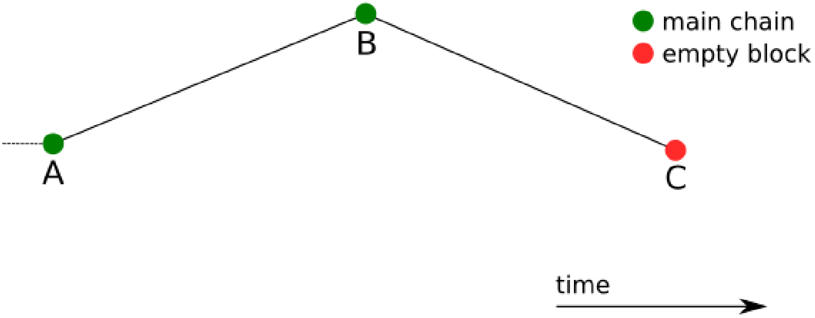


Choice 1 (today): add a block at A

- Miner risks losing if B is valid
- The block C must be *empty*

Adding a block

In the process of adding a block, a miner must choose what to do *while validating the new block*

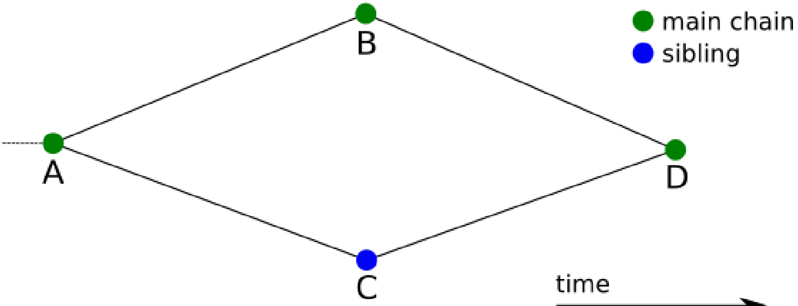


Choice 2 (today): add a block at B

- Miner risks losing if B is invalid
- The block C must be *empty*

Adding a block

In the process of adding a block, a miner must choose what to do *while validating the new block*



Choice 3 (the future):

- Allow miners to both *mine* and *validate* at the same time!
- A future block must tie up both
- Duplicated transactions must be tolerated

Block size and diminishing returns II

The point at which increasing the block size actually **decreases** the number of transactions that can be accepted by the network is around 50 MB blocks.

$\Rightarrow$ Increasing block size would work today,
but *will not work* tomorrow.

The implied maximum transaction rate of 125 transactions/sec is far below the peak capacity of competitors like Visa at 60,000 transactions/sec.

There's a much deeper underlying problem causing this.

Block size and diminishing returns II

The point at which increasing the block size actually **decreases** the number of transactions that can be accepted by the network is around 50 MB blocks.

$\Rightarrow$ Increasing block size would work today,
but *will not work* tomorrow.

The implied maximum transaction rate of 125 transactions/sec is far below the peak capacity of competitors like Visa at 60,000 transactions/sec.

There's a much deeper underlying problem causing this.

Block size and diminishing returns II

The point at which increasing the block size actually **decreases** the number of transactions that can be accepted by the network is around 50 MB blocks.

$\Rightarrow$ Increasing block size would work today,
but *will not work* tomorrow.

The implied maximum transaction rate of 125 transactions/sec is far below the peak capacity of competitors like Visa at 60,000 transactions/sec.

There's a much deeper underlying problem causing this.

Miners are unwilling to lose blocks

Satoshi's original analysis used a notion of the *highest work chain* to select among double spends.

- Satoshi assumed transactors and miners were the same
  - ⇒ Today we have a practical separation between miners and transactors.
- A block is worth around $15,000 USD. It is inappropriate to use this much value to select among double-spends using Satoshi's "highest work chain" rule.
- Practically, miners are selecting among double-spends before they ever get put into blocks. (See "thin blocks", "Xthin blocks" and other methods of "mempool synchronization" in this conference)

The Miner Income Asymmetry Problem[2]

The block size debate hides a more serious problem: miner income asymmetry.

Selfish Mining

If some miners make more than others, for doing exactly the same job, that income asymmetry can be gamed, at expense of the security of the system.

I'd go further than Eyal at all to suggest that **any** income asymmetry among miners *can be exploited*, maximizing miner profits, at the expense of the security of Bitcoin.

$\Rightarrow$ *We must remove the asymmetry.*

[2] Eyal, Sirer, arXiv:1311.0243

Equal Pay for Equal (proof-of) Work

Mining today is:

- the means of tying the value of the coin to real-world value
- NOT the means by which double-spends are selected

Larger blocks, and more orphans *increases the* **temptation** to selfish mine.

Let's recognize mining for what it is:

Mining is the means of coupling real-wold cost to the coin.

Let's ensure that equal (proof-of) Work receives equal pay!

Equal Pay for Equal (proof-of) Work

Mining today is:

- the means of tying the value of the coin to real-world value
- NOT the means by which double-spends are selected

Larger blocks, and more orphans *increases the* **temptation** to selfish mine.

Let's recognize mining for what it is:

Mining is the means of coupling real-wold cost to the coin.

Let's ensure that equal (proof-of) Work receives equal pay!

Equal Pay for Equal (proof-of) Work

Mining today is:

- the means of tying the value of the coin to real-world value
- NOT the means by which double-spends are selected

Larger blocks, and more orphans *increases the* **temptation** to selfish mine.

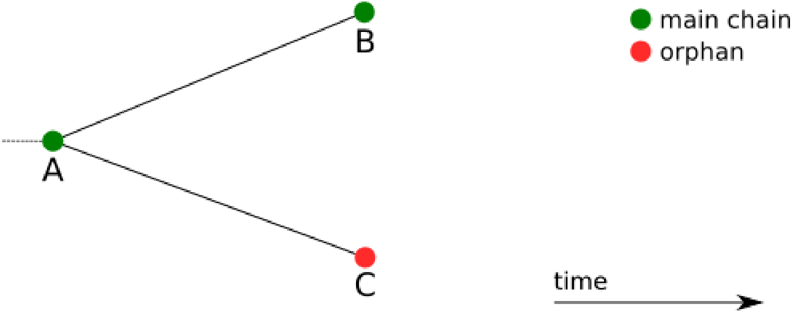
Let's recognize mining for what it is:

Mining is the means of coupling real-wold cost to the coin.

Let's ensure that equal (proof-of) Work receives equal pay!

How to Mine and Validate at the same time

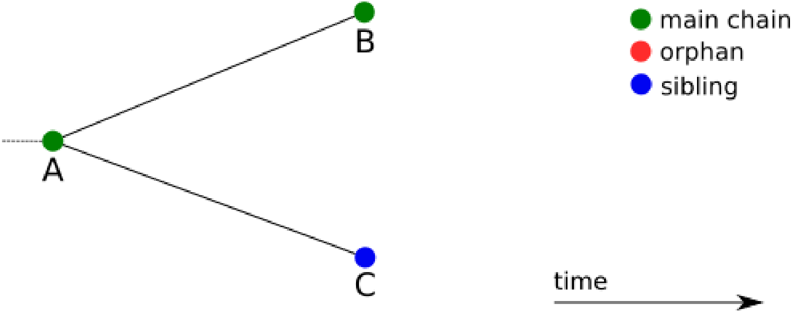
Orphans are *not* necessary for the operation of bitcoin!



- Simultaneous block generation is *unavoidable*
- Miners must be able to validate and mine

How to Mine and Validate at the same time

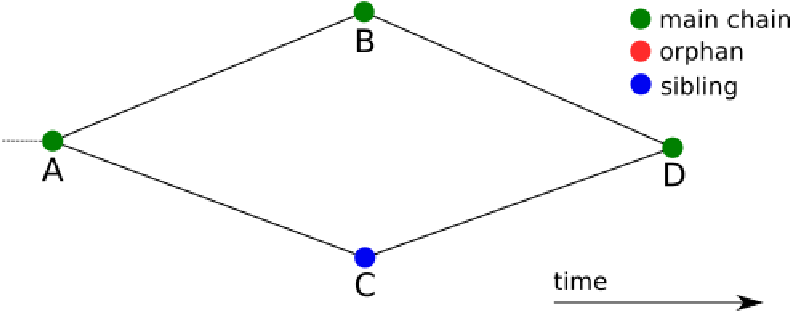
Orphans are *not* necessary for the operation of bitcoin!



- 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*.

How to Mine and Validate at the same time

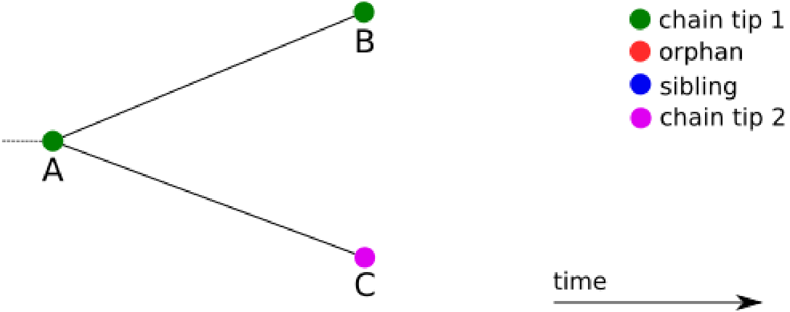
Orphans are *not* necessary for the operation of bitcoin!



- 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*

How to Mine and Validate at the same time

Orphans are *not* necessary for 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:

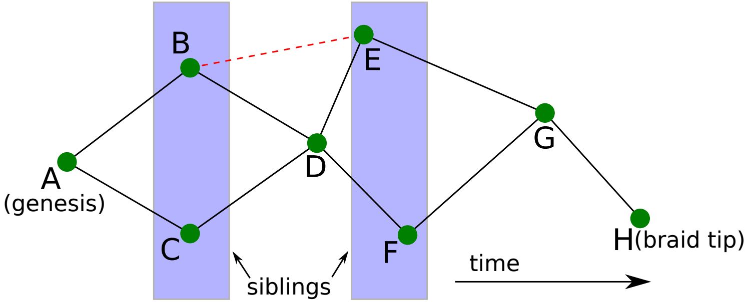
Directed Blocks have parents, parents cannot refer to children

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)

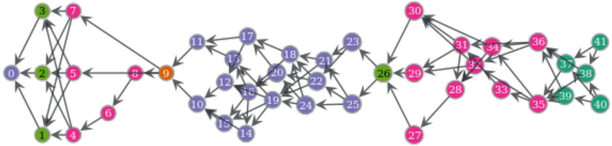
Bead Analog of Bitcoin's blocks (green circles)

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)

The Cohort

Cohorts define a *total order* such that *all* beads are ancestors of the next cohort, and descendants of the previous cohort:



Major contribution of this work: the algorithm to find cohorts.

A cohort is the analog of a block

Let's Simulate!

Let's build a "toy model" of the Bitcoin network.

- 25 node network
- 4 connections per node
- Nodes randomly distributed on the surface of (the Earth)
- Inter-node latencies decided by arc distance on sphere

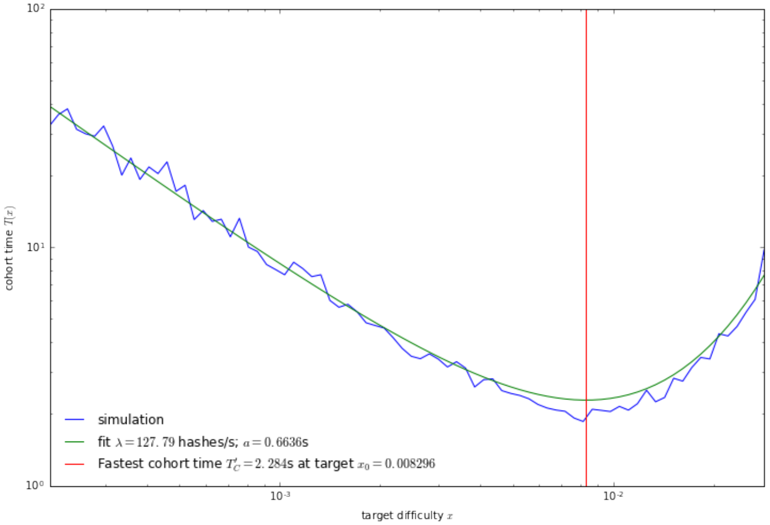
Python code available at

`http://github.com/mcelrath/braidcoin`

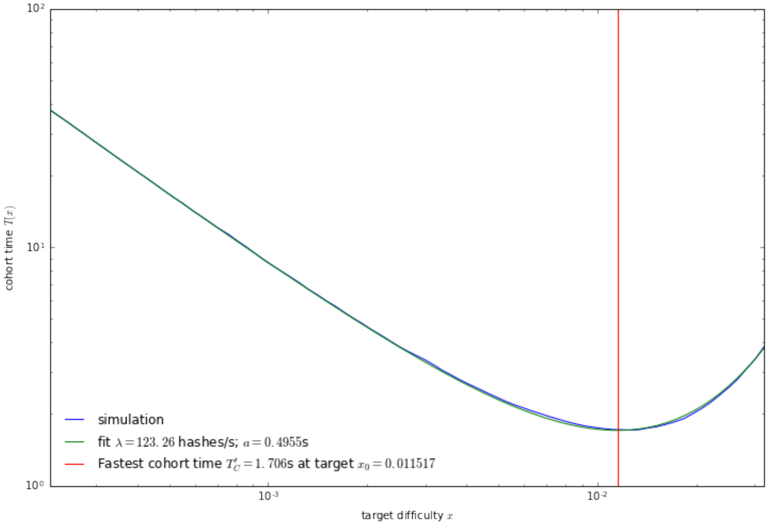
Another "toy model" for simulation is Simbit (javascript)

`https://github.com/ebfull/simbit`

Cohort Time vs Difficulty



Cohort Time vs Difficulty



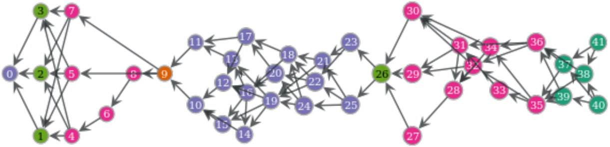
Analysis

There exists a *fastest possible cohort time!*

- HIGH difficulty (blockchain limit - LEFT side of graph)
- Beads are produced so rarely that they can almost always be ordered into a chain.

Analysis

There exists a *fastest possible cohort time!*



- LOW difficulty (braid limit - RIGHT side of graph)
- Beads are produced so fast that there's almost always one in propagating through the network!
- Poisson nature of mining makes finding a cohort exponentially less likely

Analysis II

The curve resulting from this simulation is extremely well approximated by

$$T(x) = \frac{1}{\lambda x} + a e^{a\lambda x}$$

T Cohort time	x Target hash value
λ Hash rate (hashes per second)	a Network "size" parameter (in seconds)

Let's approximate this in the blockchain limit $x \to 0$:

$$T(x) = \frac{1}{\lambda x} + a + \mathcal{O}(x)$$

the quantity a is the *increase in effective block time* due to network latency effects. This is to say that the actual cohort time is slightly longer than expected from the hashrate. a is the orphan rate, or "miner utilization".

Analysis III

The parameter λ is the hash rate and can be obtained along with a:

$$\lambda = \frac{N_B}{x T_C N_C}; \qquad a = T_C W\left(\frac{T_C}{T_B} - 1\right)$$

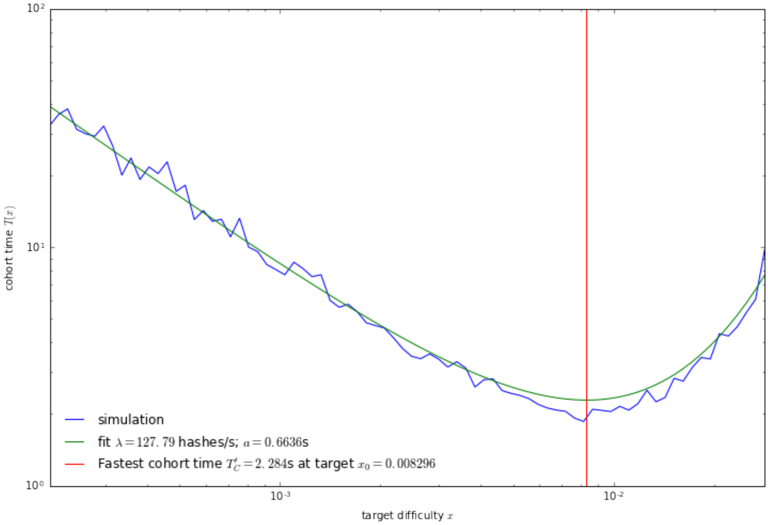
where N_B is the number of beads and N_C is the number of cohorts. $W(z)$ is the *Lambert W function*[3]. With these in hand we can compute the location of the minimum

$$x_0 = \frac{2W\left(\frac{1}{2}\right)}{a\lambda} = \frac{0.7035}{a\lambda}$$

This is relatively independent of network topology.

[3] https://en.wikipedia.org/wiki/Lambert_W_function

Cohort Time vs Difficulty



Analysis IV

We can target the *fastest possible* block time!

10 minutes is arbitrary, and slow.

We can have a **zero parameter retarget algorithm**: instead of targeting a fixed block time, targets the fastest possible block time. Evaluating the above for the Bitcoin network, which has had an orphan rate of 1.16 orphans/day on average over the last 2 years with a block time of 600s, we find that $a = 4.79$s and the cohort time at the minimum is 11.64s.

By targeting the fastest block time, we automatically compensate for network topology changes!

Analysis V

The network cannot measure time more accurately than a. This implies:

- The network cannot make state transitions faster than a
- The network cannot change difficulty faster than a
- The network cannot change rewards faster than a

$\therefore$ let us consider allowing a *range* of difficulties and rewards that are acceptable!

δx acceptable range of targets

δR acceptable range of rewards

Reward Analysis

Bitcoin's reward schedule is:

$$R(t) = R_0 e^{-t/\tau} \qquad (3)$$

where R_0 is the initial reward (50 BTC) and τ is the time constant of the decay, which is related to the halving time by $\tau = T_H / \ln 2$ where T_H is the 4-year halving, resulting in $\tau = 1.82 \times 10^8 s$ for Bitcoin.

At $t = 0$, the uncertainty on the reward, due to the uncertainty on time, is about 132 Satoshis.

It's unlikely that miners would game this, as it represents only a 0.00000026% change in reward.

Summary/Conclusions

- We present code and algorithms to allow *blocks to have multiple parents*, thereby allowing miners to both *validate* and incoming block and *mine* a new one at the same time.
  - ⇒ Our algorithm organizes a chain allowing multiple parents into a ordered set of cohorts (blocks) *without additional assumptions*
- We suggest that *all* blocks satisfying the proof-of-work condition be accepted to the main chain,
  - ⇒ We must find an alternative to Satoshi's "highest work chain" rule to select among double-spends.
- We suggest that the consensus rules allow for a *range* of target and reward values
  - ⇒ Accept the fundamental fact: the network is asynchronous, and has a consensus time of a.