Keeping TABS

A Canon-Arbitration Algorithm for *Modified-GHOST* Proof-of-Work Blockchain Protocols

Isaac Ardis

Daniel Aronoff
MIT Economics
MIT Media Lab

April 22, 2022

Abstract

Canon-arbitration algorithms are used to build consensus in decentralized state machines. The consensus characteristics of mutually implemented canon-arbitration algorithms enables a race between block authors and thereby facilitate the trust-minimized security characteristics of the blockchain state. In Proof-of-Work (PoW) block validation protocols canon-arbitration algorithms typically depend on dynamic emission parameters like Difficulty, facilitating a near-constant block emission rate for the network.

We proposed and evaluate a modification to the *Modified-GHOST* [1] algorithm used by Ethereum. This proposal introduces two novel features: a capacity for segment-specific transactions, and a positive scalar value representing a general measurement of active capital per block. From these features we construct a novel canon-arbitration strategy which preserves the incumbent PoW chain-growth game while improving the expected block finality rate by 50% and raising the cost of double-spend attacks.

We evaluate associated additional risk to the network, which we show to be practically nominal. A proof-of-concept code implementation of the protocol is drafted for Ethereum's primary client *go-ethereum*[2], and is included with or easily referenced from this document.

[1] https://ethereum.org/en/whitepaper/#modified-ghost-implementation

[2] https://github.com/ethereum/go-ethereum

1 Model Definitions

Table 1: List of Model Definitions

Model Elements	
t	Time. Standard unit: seconds.
λ	Network mean block time; the average number of seconds between block timestamps.
η	Network block-message latency (measurable or assumed) between two nodes. The time elapsed between the transmission of a message containing block information and its reception.
B_i	A block with number i.
H_i	A block header with number i.
H_d	Block (and header) annotated difficulty.
H_k	Block (and header) annotated TABS value.
$TAB(B)$	A scalar value representing aggregated account balances for a given block B. "TAB" stands for Total Active Balance.
$TABS(B_i, B_{\text{i}-1})$	A scalar value derived from the TAB value of block B_i and recursively its parent B_{i-1}. This value is annotated in a block's header. "S" stands for *synthetic* or *synthesis*.
$\sigma[a]$	Global account state σ at address a.
T	A transaction.
T_a	A proposed transaction field: Ancestor Hash. See Glossary.
$\delta(T_a)$	The depth (magnitude of ancestry) of a block cited by some transaction as T_a.
τ	A network-level aggregate over time or segment (assume: average) of Ancestor Hash depth δ values. Transactions *not* asserting an Ancestor Hash value can be defined as having $\delta = H_i$. Smaller values represent a greater dependence on blocks nearer a common (consensus) chain head (a greater degree of transaction/segment exclusivity), while larger values signal less exclusive transaction assertion on chain state dependence.
CS	Canon Score. A scalar value representing canon preference weight of a single block. Comparable to Difficulty.
TCS	Total Canon Score. A scalar value representing protocol canonical preference. Derivation of this value for any block relies on the antecedent calculation of that of its parent. Comparable to Total Difficulty.

2 Verbal Glossary

Table 2: Definitions of Common Terms

Glossary	
Ancestor Hash	A proposed transaction field whose value references a block by hash, specifying the existence of that block in the transaction's chain segment as a condition of the transaction's validity. Proposed. Use optional. Synonyms: Segment ID, HFC (Header Field Context) Hash. Related: Chain ID.
Canonical-Arbitration Algorithm	A program of comparative conditions resulting in the selection of one prioritized segment from any two segments. In context, the prioritized segment is used as the basis for determining a canonical chain state, and, as such, for focusing mining effort (via establishing a *parent*. Synonyms: Canonical-Preference Algorithm. Related: GHOST, *Modified-GHOST* , Inclusive Protocol, Nakamoto Protocol.
Segment-Agnostic (Transaction)	A property on a transaction such that the transaction can be applied to one or more chain histories. Related: Segment-Specific.
Segment-Specific (Transaction)	A property on a transaction such that the transaction can be applied on a subset of available chain histories. Synonyms: Segment-Exclusive, Chain Context Constrained. Related: Segment-Agnostic.
Chain State Finality	The permanence of chain state (or series of states). PoW blockchain states have characteristically 'soft' finality characteristics. PoW finality expectations increase as relative state increases; newer states are more vulnerable to change than older states.
Block Emissions	The (rate of) production of blocks. Blocks are metadata associated with chain state modification sequences.
Decidability	The frequency of ties in canon-arbitration during chain growth. Narula et al. call this *symmetry-breaking*. Decidable scenarios are characterized by eventual network-level chain state consensus. Undecidable scenarios are characterized by bifurcated (canonical) network chain states.

3 Visual Glossary

Figure 1: A block

Figure 2: A transaction

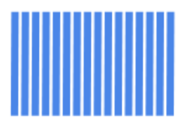


Figure 3: Several transactions

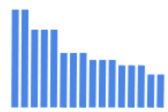


Figure 4: Several transactions representing relative gas consumption

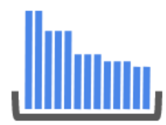


Figure 5: Transactions in a block

Figure 6: A block segment (chain)

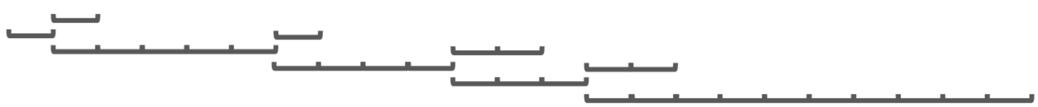


Figure 7: A block segment with forks

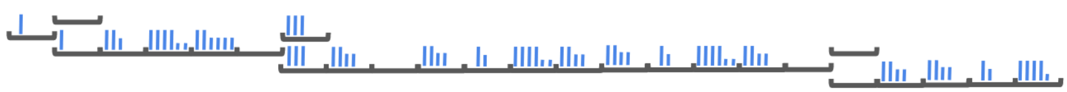


Figure 8: A block segment with forks and sparse transactions

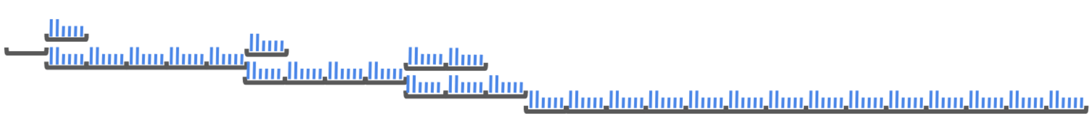


Figure 9: A longer block segment with forks and full of transactions

Figure 10: Block composition of a large reorg (eg. finality attack)

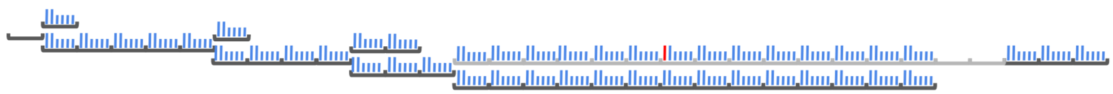


Figure 11: Block composition of a large reorg (eg. finality attack) including transactions. The ‘fraudulent‘ modified or omitted (censored) transaction is red. which the double-spend was made?

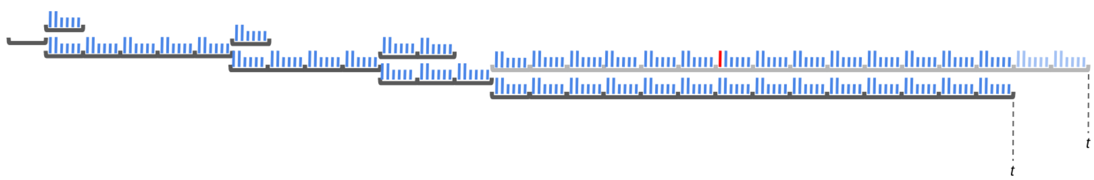


Figure 12: Block composition of a large reorg (eg. finality attack), noting a value t representing some point in time. Since the domain of this visualization is (implicity) block number, we deduce that the gray (top) chain has built more blocks per unit of time than its counterpart. The value of time t in this case can be understood as an arbitrary instant where the segments (via their head blocks) are evaluated for canonical status.

Figure 13: Two illustrations of transaction inclusion under the large reorg scenario and assuming successively aggressive τ rates (composite frequency and value of the proposed Transaction Ancestor Hash field and validation).

Figure 14: A component diagram of block and transaction occurrences under the assumed reorg scenario with transactions using Ancestor Hash.

Figure 15: Visual conceptualization of the incumbent canon-arbitration initial condition using Total Difficulty values.

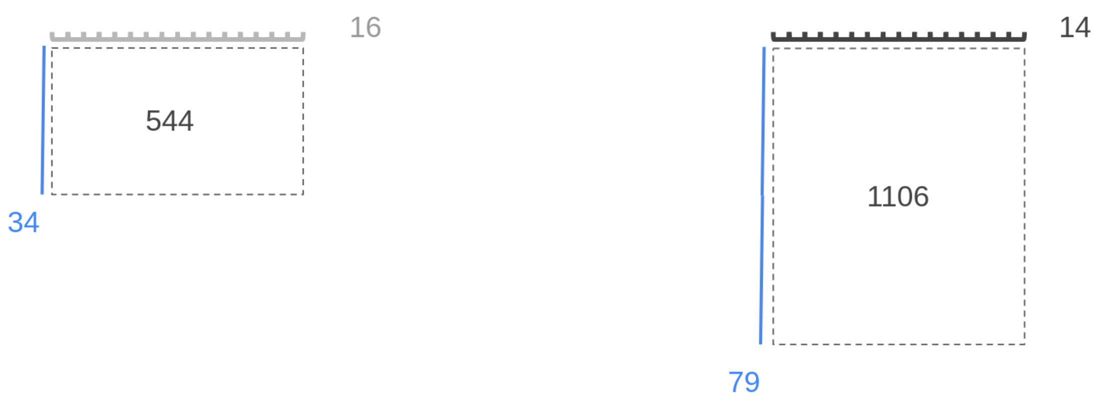


Figure 16: Visual conceptualization of the proposed canon-arbitration initial condition using Difficulty and TABS values.

4 Scenarios

Table 3: Abstracts of Referenced Chain State Scenarios

Scenarios	
Finality Fraud	A premeditated fraud scheme exploiting a victim's assumption of chain state which is later invalidated by the attacker. The exemplary scenario defined and reused throughout this paper assumes a single entity as a victim and a single censored transaction. In most contexts, these assumed parameters could be modified to use a plural set of victim entities and/or a plural set of censored transactions while still being considered Finality Fraud.
Finality Terrorism	Generally schematically equivalent to Finality Fraud; the difference is in the attacker's motive and an intentionally large number of victims. The exemplary scenario defined and reused throughout this paper assumes all users of a blockchain as victims and the censorship of all transactions. These parameters are extremes. In most contexts, these assumed parameters could be modified while still being considered Finality Terrorism.

5 Specifications

In this section we describe the elements of the *TABS* modification to the *Modified-GHOST* algorithm.

5.1 Transactions with Ancestor Hash

Transactions are afforded an additional field T_{a} (*Ancestor Hash*) whose value either describes a block hash or is left undefined.[3]

If a transaction defines T_{a}, its validity is conditioned on the existence of a block with this hash in the canonical chain prior to the block including the transaction.

Transactions of this type are considered segment-specific. The degree to which they are specific depends on which block is referenced by the hash is provided. Transactions referencing relatively older blocks are less specific than those referencing newer ones. The generalization of this concept is defined as τ (see Glossary).

As proposed, transactions are not required to provide this value. However, discretionary use is not necessary. The field's annotation *could* be demanded by the protocol.

5.2 TABS Validation

Total Active Balance Synthesis (TABS) is a positive scalar value, an aggregate of all transaction sender accounts (and the miner) for a blockchain of some length. Generally considered, it provides a representative measure of active capital on some chain state. It is denominated in Wei.[4] Its value is used as a parameter in the proposed canon-arbitration condition.

Validation of the TABS value of a block of header H is asserted as $\mathrm{TABS}(H)$,

where:

$$\mathrm{TABS}(H) \equiv \begin{cases} K_0 & \text{if} \quad H_{\mathrm{i}} = 0 \,||\, P(H)_{H_k} \text{is undefined} \\ \max(K_0, P(H)_{\mathrm{H_k}} + x \times y) & \text{otherwise} \end{cases} \tag{1}$$

where:

$$K_0 \equiv 128 \times 10^{18} \tag{2}$$

[3]Original and complete specification, rationale, and motivation for this feature is documented as a standalone proposal at https://github.com/ethereumclassic/ecips/tree/master/_specs/ecip-1105.md

[4] 1000000000000000000 (1e18) Wei equal 1 Ether.

$$r \equiv 4096 \tag{3}$$

$$x \equiv \left\lfloor \frac{P(H)_{\mathrm{H_k}}}{r} \right\rfloor \tag{4}$$

$$y \equiv \begin{cases} -1 & \text{if}\,\mathrm{TAB(B)} < \mathrm{P(H)}_{H_k} \\ 0 & \text{if}\,\mathrm{TAB(B)} \equiv \mathrm{P(H)}_{H_k} \\ 1 & \text{if}\,\mathrm{TAB(B)} > \mathrm{P(H)}_{H_k} \end{cases} \tag{5}$$

The TAB(B) value of a block is defined as the sum of the miner balance and all of the sender balances for transactions with an assertive T_a ancestor hash field:

$$TAB(B) \equiv L(B, B_c) + \sum_{T \in B_{\mathrm{txes}}} L(B, T_{\mathrm{sender}}) \tag{6}$$

where:

$$L(B, s) \equiv P(B)_{\sigma[s]_{\mathrm{b}}} \quad \text{The account state balance at parent block } P(B) \text{ of account } s \tag{7}$$

K_0 expresses an arbitrary minimum value for $TABS$, while x expresses an adjustment ratio of the $TABS$ value between two parent-child related blocks. The constant r is defined relative to the minimum incumbent Ethereum difficulty adjustment factor ($\left\lfloor \frac{P(H)_{\mathrm{H_d}}}{2048} \right\rfloor$) with r being marginally less in order to maintain the precedence of the difficulty value in the resulting Consensus Score product. This ensures that difficulty remains the primary driver of chain growth, keeping miner incentives invariant.

5.3 Canon Scoring

Canon-arbitration algorithms are assumed to compare two blocks and to indicate decisively which one of the two ought to receive canonical status. This section formally defines a value Canon Score *CS*, which is used as the *initial condition* for a canon-arbitration algorithm.

In both the pre-existing and proposed algorithms, this condition may result in a tie. If the condition is decisive (a winner is determined), subsequent conditions are not evaluated. If indecisive (the result is a tie), subsequent conditions are evaluated, each potentially decisively, in program order.

For the sake of context, we write the assumed existing initial condition for canon-arbitration in our notation below. The **pre-existing Canon Score value** is defined as:

$$\mathrm{CS} \equiv H_d \tag{8}$$

And derived from that a Total Canon Score (TCS) for any segment as:

$$\mathrm{TCS} \equiv \sum_{H_{\mathrm{i=0}}}^{\mathrm{i<=n}} H_d \tag{9}$$

Respective to this assumed pre-existing condition, **we propose its modification** as,

in the atomic case:

$$\mathrm{CS} \equiv H_d \times H_k \tag{10}$$

and in the segment case:

$$\mathrm{TCS} \equiv \sum_{H_{\mathrm{i=0}}}^{\mathrm{i<=n}} H_d \times H_k \tag{11}$$

In both the pre-existing algorithm and the proposed algorithm, a candidate block having a greater TCS value than the other should be selected for the exclusive canonical state.

6 Models

We model block emissions at the network level using a Poisson Distribution (where independent mining exemplifies a Poisson Process).[5]

6.1 Miner Expected Revenue

We naively define miner expected revenue over time t (in seconds) as:

$$\frac{Miner_{h/s}}{Network_{h/s}} \times P(E,t) \times P(C) \times (Block_{\text{reward}} + Block_{\text{tx}_{\text{f}}\text{ees}}) \tag{12}$$

(Note that the ratio $\frac{Miner_{h/s}}{Network_{h/s}}$ is intuitive but naive, and can be further developed.)

6.2 Probability of network block production

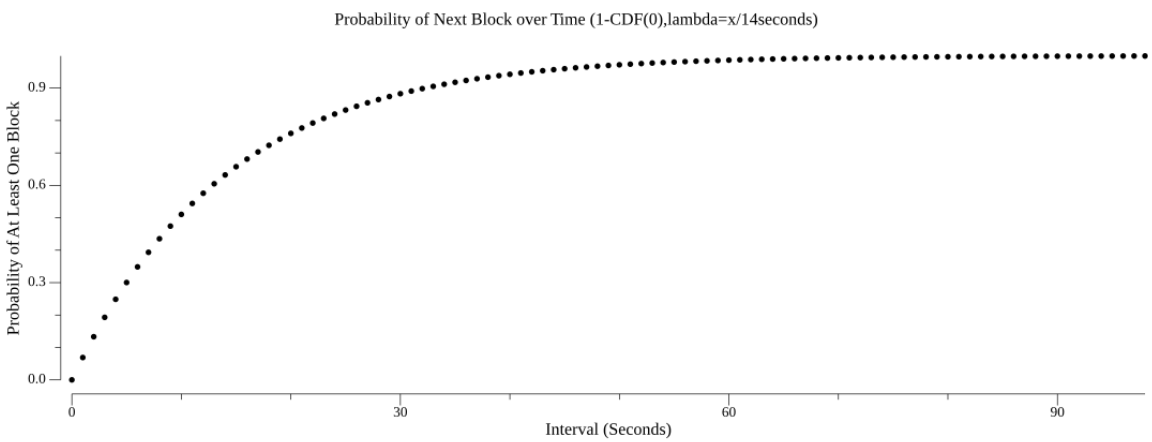


Figure 17: A Poisson Cumulative Distribution Function (CDF) is used to model the probability of a block occurrence (y-axis) in some interval (x-axis). This plot is produced by the included Go test `TestPoissonCDFNextBlock`.

[5] - Nakamoto. 'Bitcoin: A Peer-to-Peer Electronic Cash System', 2007.
- Ren. 'Analysis of Nakamoto Consensus', 2019.
- Gazi, Kiayias, Russel. 'Tight Consistency Bounds for Bitcoin', 2020.
- Su, Liu, Narula. 'The Power of Random Symmetry-Breaking in Nakamoto Consensus', 2021.

We assume that the occurrence of block events at the network level can be modeled as a Poisson Process.[6] From this, we can define the probability of any block being discovered in an interval of t seconds given an event rate r (such that $r * t = \lambda$) as a derivation of the Poisson CDF. [7]

$$P(E,t) \equiv 1 - e^{-\lambda} \times \sum_{i=0}^{|k|} \frac{\lambda^i}{i!} \tag{13}$$

6.3 Probability of a block's canonical state

The probability (at the network level) of a block being ultimately accepted as canonical by the network:

$$P(C) \equiv 1 - \epsilon \tag{14}$$

In this simple definition, ϵ represents the frequency in which a miner authors a valid block which is not ultimately accepted into the public canonical chain.

We take $\epsilon < 1$ for granted as common sense; canonical blocks exist. We can also assume that $\epsilon > 0$, since non-canonical blocks exist and are observable as uncles (also known as "ommers") on chain.

A practical value of ϵ can be approximated by an empirical measurement of a network's uncle rate. Given that observable orphan data on chain may not be complete, we can derive a reasonable expectation for ϵ using a theoretical model, as follows.

Assume that for some time interval (eg. 1 second) a (one) block has been created. We model the probability, then, for the case that *another* block will *also* be created in that same interval.

$$\epsilon \equiv \frac{\frac{1}{14}^{k=1} e^{-\frac{1}{14}}}{k = 1!} = 0.066504484... \tag{15}$$

The value 0.0665... represents the probability that for this arbitrary interval, at the network level, another (one) block is also found.

[6] One must note that while it may be granted that the model fits *recorded* network block interval data, it may not necessarily represent the *actual* production intervals. Neither must it represent accurately the production interval of blocks by any subset of miners.

[7] The plot in Figure 17 shows this probability for intervals of 0 through 99 seconds assuming an average block interval rate of 14 seconds.

To derive the probability of any one miner winning this arbitration, the network-level probability of orphanhood should be divided by the number of competing miners (assumed here to be 2).

$$P(C) \equiv 1 - \frac{0.0665}{2} = 0.96675 \tag{16}$$

Generalization of this model to any number of near-simultaneous block-production events is trivial by modifying the Poisson k value, and adjusting the miner share respectively.

6.4 Network orphan probability

The expected network orphan rate is generally the converse of the previous model.

Since the orphan rate for a network can be (albeit somewhat loosely) understood as the occurrence of at least (but usually only) two blocks within some interval (usually the network latency variable, but here 1 second), the expected network orphan rate with these assumptions is around 0.0665.

We can extend this definition by including cases where another one *or more* blocks are found in the given interval.

$$\sum_{k=1}^{\infty} \frac{\frac{1}{14}^k e^{-\frac{1}{14}}}{k!} = 0.068937 \tag{17}$$

Note that as the considered $\text{interval}_{1\ \text{second}}$ changes, this value will change, and that determination of this interval should be made with respect to relevant estimated network latencies.[8]

7 Discussion

The consensus properties of so-called Nakamoto Consensus are well studied.

We seek to show that an implementation of the proposed AHA/TABS protocol for a PoW network either preserves or improves key characteristics of network behavior and usability.

We provide the outcomes of two programmatic simulations as experimental evidence of our claims.

[8] Network latencies between miners are more relevant, for example, than latencies between non-mining nodes.

7.1 Chain-Growth

Because the TABS adjustment rate is smaller than that of Difficulty in all cases, we can show that PoW mining remains the primary driver of chain-growth.

A: Difficulty+, TABS- B: Difficulty0, TABS+ =¿ B

$$(H_d + \frac{H_d}{2048}) \times (H_k - \frac{H_k}{2049}) > (H_d + 0) \times (H_k + \frac{H_k}{2049}) \tag{18}$$

$$(H_d + \frac{H_d}{2048}) \times (H_k - \frac{H_k}{2049}) > (H_d + 0) \times (H_k + \frac{H_k}{2049}) \tag{19}$$

A: Difficulty+, TABS- B: Difficulty-, TABS+ =¿ A

$$\text{TCS} \equiv \sum_{H_{\text{i=0}}}^{\text{i<=n}} H_d \tag{20}$$

(12947922331000968 + 12947922331000968 * (1/2048)) * ()

7.2 Decidability Matrix

TABS is expected to increase network decidability rates.

table content

7.3 Network block emission rates

We can reframe the previous section's theoretically modeled orphan rate as an indicator of what some domain literature considers "symmetry" during "chain-growth."[9] The coincidence of block emission occurrences is an event of chain-growth symmetry. Since only one block may be ultimately selected for canonicalization, we consider the rate of orphan production a measure of network *efficiency*. Orphan blocks represent wasted electricity on behalf of the orphan miner.[10]

[9] Narula, 2021.

[10] Compensating for orphans is the main objective of the GHOST protocol, which specifies increasing difficulty for blocks with uncle citations.

The AHA/TABS protocol is expected to reduce the orphan rate of the network by increasing decidability (asymmetry) during canon-arbitration by virtue of an assumed asymmetry in respective block Total Active Balances.

Block emission rates will not change. Emission rates are a function of difficulty, which is codependent with block intervals. The expected reduction in network waste will *not* ultimately cause faster chain-growth. Instead, it will cause an increase in *finality = canonical* rates for proposed blocks. There will be fewer uncles.

Should uncles be rewarded less? More? The same? #*Modified-GHOST* .

7.4 Miner incentives

Where Total Difficulty alone drives the *Modified-GHOST* and Nakamoto protocols, AHA/TABS introduces another variable in the miner's efforts to produce a canonical block: transaction sender balances. Since this value should be expected to impact their probability of a canon win, we should expect them to prefer transactions with senders of greater balance over those with lesser.

To what degree should miners prefer transactions and their balances over puzzle solutions? Will slower blocks win more than they do now? Will rich miners win more blocks than poor miners?

* Because we assume that miners (entities) hold minority shares of hashrate (no 51+%s), we expect that the most profitable strategy for miners is to mine on top of the last best block they observe. We can thereby expect that puzzle solutions will continue to be the primary driving factor for chain-growth.

Do we expect any additional vulnerabilities to malicious behavior? Can the TABS Consensus Score be exploited to allow slower blocks produced by rich miners to win?

When could a slower block produced by a rich miner win? Block difficulty function is a step function parameterized by block interval. We assume two blocks: - A: t=9 - B: t=10

... For blocks produced within some network latency of each other, but occupying different steps in the difficulty function (thus, different TDs):

1 / 2048 2 / 2048 with uncles = 1 / 1028

minimum difficulty = 131072 current difficulty = ETH: 14615997 0xf378378d0575b8ed5ac48434173a61f29b693a D=12,947,922,331,000,968 ETC: 14961128 0x20cceb7960ce30edea9c5f2b817a861ecf7ebe5b0f7a518a4f6fdd15380 D=00,357,336,764,833,767

d * tabs ¡ d * 1027/1028 * tabs * tabs$_multiple1 < 1027/1028 * tabs_multiple1028/1027 < tabs_multiple$

but as written: $\text{tabs}_multiple = 1/128$

7.5 Transaction availability

Poor transactors should expect lesser transaction availability, richer transactors should expect greater availability. Overall network transaction availability is preserved as an invariant.

8 Program Models

The theoretical models described in the previous section serve to frame our understanding and expectations for programmatic simulations.

We use two computer program simulations to model competitive block emissions.

`go-block-step` models block emissions from the network level. It tests and compares simulated network emission outcomes against the expected Poisson distribution of block time intervals.

`go-miner-sim` models block emissions in an "actor-based" way, simulating block emissions given a set of independent miners following a common protocol.

As neighbors, these models can be considered as 'top-down' (`go-block-step`) and 'bottom-up' (`go-miner-sim`) approaches to network block emissions and consensus modeling.

We compare the data produced by these programs with empirical chain data, testing for representativeness. We want to understand the change the TABS protocol presents on block emissions rate, network consensus rate, miner incentives, and transaction liveness[11].

9 Model Comparison: Fork Rates

We have now two models which can be used to estimate fork rates during simulated chain growth. We compare the information provided by these models with empirical data.

We are concerned with fork rates because a fork represents, by definition, an instance of reduced finality. Only one block per block number can be ultimately considered canonical; the others will become obsolete (impermanent).

[11] *Liveness* is a database-jargon term used in several related and referenced papers. In this context, it is the general chain state's processing of available transactions.

The ambiguity of *which* block shall become canonical is a matter of network, and miner, efficiency. Time and energy spent on impermanent blocks is wasted.

We note further, that the cost of forks are not limited to the two (or miner) beneficiary authors of those blocks. We can consider, as an example, that each of two miners broadcasts their candidates to the network nearly simultaneously. Attributed to network latencies and graph shapes, the network consensus bifurcates and global hashrate is halved (each miner having won-over half of the network). At this point 50% of network hashpower will be ultimately wasted; only one side of the fork will ultimately become canonical. The cost of the ambiguity is borne by all players (though not necessarily fairly distributed).

This assumed state of ambiguity of the candidate blocks is the result of the canonical arbitration algorithm. A canonical arbitration algorithm yielding a lower ambiguity rate will reduce the network's block emission waste and improve expectations of permanence for the overall chain state.

9.1 Simulated Fork Rates vs. Empirical Uncle Rates

We know that uncle rates represent a theoretical minimum measurement of actual orphaned block rates; the orphans *may* be recorded but are *not necessarily*.

Can we use an economic model of the incentives of orphan records to suggest to what degree empirical uncle block rates suggest a complete record? If miners are not incentived to record orphans they do not mine themselves, then the recorded orphans will omit those. On the other hand, if it is profitable for the miner to record orphans regardless of their author, then we should expect the record to be more complete.

The revenues from orphan recording are inuitive; they are clearly defined and scaled to the block reward. They are positive. But what are the costs of recording an uncle?

Total network Wei supply is assumed to be finite for the sake of argument.[12]

Since an orphan record benefits the miner of the orphan *more* than it benefits the recording miner, we are lead to reason that it *may* be that the cost of distributing *any* capital to competitors dilutes the value of a miner's own capital.

We leave this line of reasoning open-ended.

For the sake of our proposed model, it is enough to assume an observable objective measurement (uncle rate), and to handle it as theoretical minimum proxy for actual fork rate.

We turn to empirical data.

[12] This is not a guarantee for Ethereum, though it is for Ethereum Classic.

At the time of writing, Etherscan.io[13] shows $1,207,850$ recorded uncles and a current canonical height of $13,753,436$.

$$1207850/13753436 = 0.0878217 \tag{21}$$

We interpret this as representing an orphan rate of about 8.7%.

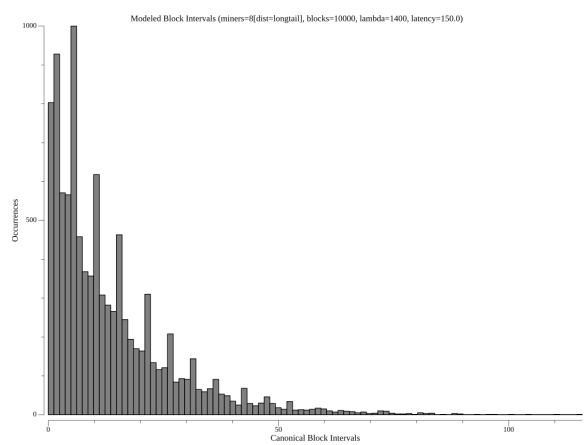
Comparatively, using the `main` program we simulate a network with $\lambda = 14$, $\eta = 1.5$, 8 miners with hashrates distributed as an approximate long-tail, at a tick interval representing 100 milliseconds, over $10,000$ samples, and yielding a fork rate of 8.39%.

Raising latency to $\eta = 2$ can raise the rate to 11.7%. Alternatively, raising the number of miners to 16 seems to raise the rate to around 8.7%.

It is tempting to manipulate modeled latency values in order to meet the shape of empirical block interval data. But we must resist this temptation.

We do not know real network latency values, and have no objective way of measuring them. Further, we expect that even reliably measured subjective values would vary and are regularly subject to change. This point is especially pertinent when latency is considered as both mechanically-derived (tube limits) and arbitrarily decided values (eg. selfish head-start delays).

We note, however, that under this simulation, the generated block interval distribution (Figure right) doesn't seem to match the empirical data. The intervals are too small.



If we re-prioritize the fit to a back-of-napkin same-shape rubric on block intervals, a $\eta = 4.2$ approximates a better fit (Figure 25, below). At this latency rate, the model yields a simulated fork rate of approximately 23.5%. A Poisson distribution sampling for the same tick duration is overlaid in red.

Our intention is not to build a best-fit model for empirical data.

Rather, the intention for the model is to adequately theoretically represent the game which drives the PoW network. With a model that *behaves like* the real world in its key measure-

[13] `https://etherscan.io/uncles`

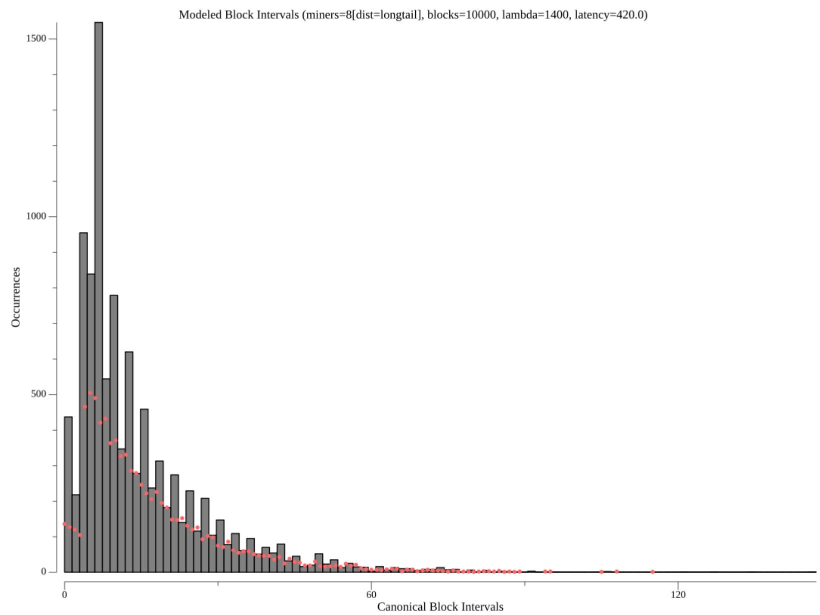


Figure 18: Poisson Distribution interval distribution samples overlaid on simulated intervals.

ments, we can explore the impact of the proposed algorithmic modification to the canonical selection logic.

9.2 Comparing Simulated Fork Rates: Proposed vs. Incumbent Canonical-Selection Algorithms

When an event $k \geq 2$ occurs, network protocol defines a set of rules for determining (deciding/arbitrating) canonical status for exclusively one event (where an 'event' is equivalant to an eligible block).[14]

We compare the incumbent Total Difficulty (TD) condition with the proposed Total Difficulty $\times$ TABS (TDTABS) condition by replacing the arbitration logic used by the model in scenarios (block-space rounds) with 2 or more candidate blocks, focusing on the respective rates of 'decidability', where the condition returns an unambiguous choice for the canonical block.

Under the same conditions of the scenario presented in the previous section, we find the following exemplary values:

[14] In fact, the protocol design for canonical-selection is applicable to *all* scenarios of canonical-status arbitration between *any* two blocks.

```
GOROOT=/home/ia/go1.16.2.linux-amd64 #gosetup
GOPATH=/home/ia/go #gosetup
/home/ia/go1.16.2.linux-amd64/bin/go build -o /tmp/GoLand/___go_build_github_co
/tmp/GoLand/___go_build_github_com_whilei_go_poisson_collision
-------------------------------------------------------------

CONFIG

        Name:              A, ConsensusAlgorithm: TD,
        NetworkLambda:     14, Latency:          1.40,
        TickMultiple:      100, Rounds:           10000,
        NumberOfMiners:    8, HashrateDistType: longtail,

GENERATED MINERS

        Number: 8, Distribution: longtail, Hashrate Checksum OK:
true,
        Hashrates: [0.333 0.222 0.148 0.099 0.066 0.059 0.044 0.02]

INTERVALS

        Mean: 1416.7087,
        Med: 973.0000, Mode: [163.0000],
        Min: 1.0000, Max: 13713.0000,


ELIGIBLE AUTHORS PER BLOCK

        Mean: 1.0793,
        Med: 1.0000, Mode: [1.0000],
        Min: 1.0000, Max: 4.0000,


MINER CANONICAL WINS

        miner=0 hashrate=0.333 winrate=0.344 winrate/hashrate=1.031 (3436)
        miner=1 hashrate=0.222 winrate=0.220 winrate/hashrate=0.990 (2201)
        miner=2 hashrate=0.148 winrate=0.149 winrate/hashrate=1.005 (1489)
        miner=3 hashrate=0.099 winrate=0.099 winrate/hashrate=1.001 (989)
        miner=4 hashrate=0.066 winrate=0.064 winrate/hashrate=0.969 (638)
        miner=5 hashrate=0.059 winrate=0.056 winrate/hashrate=0.948 (555)
        miner=6 hashrate=0.044 winrate=0.043 winrate/hashrate=0.968 (425)
```

```
miner=7 hashrate=0.029 winrate=0.027 winrate/hashrate=0.912 (267)
```

ANALYSIS

```
Ticks: 13056327, Rounds (Blocks): 10000, Ticks/Block: 1305.6327
AuthorSameParentChildTally/Block: 0.207
ArbitrationDecisiveRate: 0.039, ArbitrationDecisiveTally: 386
ArbitrationIndecisiveRate: 0.036, ArbitrationIndecisiveTally: 362


Elapsed: 2.892s
```

9305 POISSON INTERVALS

```
Mean: 1515.6216,
Med: 1095.0000, Mode: [231.0000],
Min: 1.0000, Max: 13049.0000,
```

CONFIG

```
Name:               A, ConsensusAlgorithm: TDTABS,
NetworkLambda:      14, Latency:            1.40,
TickMultiple:       100, Rounds:            10000,
NumberOfMiners:     8, HashrateDistType: longtail,
```

GENERATED MINERS

```
Number: 8, Distribution: longtail, Hashrate Checksum OK:
true,
Hashrates: [0.333 0.222 0.148 0.099 0.066 0.059 0.044 0.02]
```

INTERVALS

```
Mean: 1408.3236,
Med: 973.0000, Mode: [248.0000],
Min: 1.0000, Max: 18234.0000,
```

ELIGIBLE AUTHORS PER BLOCK

```
Mean: 1.0803,
Med: 1.0000, Mode: [1.0000],
Min: 1.0000, Max: 4.0000,
```

MINER CANONICAL WINS

```
miner=0 hashrate=0.333 winrate=0.348 winrate/hashrate=1.044 (3480)
miner=1 hashrate=0.222 winrate=0.225 winrate/hashrate=1.012 (2248)
miner=2 hashrate=0.148 winrate=0.148 winrate/hashrate=0.998 (1479)
miner=3 hashrate=0.099 winrate=0.099 winrate/hashrate=1.005 (993)
miner=4 hashrate=0.066 winrate=0.063 winrate/hashrate=0.960 (632)
miner=5 hashrate=0.059 winrate=0.052 winrate/hashrate=0.882 (516)
miner=6 hashrate=0.044 winrate=0.039 winrate/hashrate=0.882 (387)
miner=7 hashrate=0.029 winrate=0.026 winrate/hashrate=0.906 (265)
```

ANALYSIS

```
Ticks: 12990256, Rounds (Blocks): 10000, Ticks/Block: 1299.0256
AuthorSameParentChildTally/Block: 0.219
ArbitrationDecisiveRate: 0.051, ArbitrationDecisiveTally: 515
ArbitrationIndecisiveRate: 0.025, ArbitrationIndecisiveTally: 250


Elapsed: 2.776s
```

9233 POISSON INTERVALS

```
Mean: 1515.0680,
Med: 1055.0000, Mode: [141.0000],
Min: 2.0000, Max: 15252.0000,



Process finished with the exit code 0
```

The TD model shows a decision rate of 3.9%, and an ambiguity rate of 3.6%, where both rates are measured against total rounds.

The TDTABS model shows a decision rate of 5.1%, and an ambiguity rate of 2.5%, where both rates are measured against total rounds.

10 THE REST OF THIS IS UNEDITED CRUFT

Here be dragons. Ye be warned.

10.1 Probability of canonical acceptance of an eligible block by the network

Intuitively (and ideally), this value is 1. Empirically, this value is less than 1.

We attribute this to network latencies and network graph shapes.[15]

Scenario:

Blocks are produced independently and simultaneously by at least two miners. These blocks have equivalent Total Difficulty values and numbers (for the sake of argument). We assume that the network is bifurcated (forked). For sake of argument, we assume 50% of the network (by hashrate, say) is on one side of the fork, and 50% on the other.

Assuming 'good' data availability is maintained for *the whole network*, we expect that the fork to resolve quickly. Probably one side of the fork will produce a block quicker than the other, yielding a subtree with a greater Total Difficulty value than either of the two forks previously. This 2-block chain segment is expected to be adopted as canonical as soon as it is made available to a node. For 50% of the network, this will result in a minus-1/plus-2 block reorg. For the other 50% who were originally on the eventually-winning segment, no reorg is required and the next block is appended. Consensus is reached again.

The general case for the expectation of this resolution is the probability of simultaneous block production in a series of arbitrary length. If we define the probability of simultaneous block production as P_s, then it follows that the probability of a bifurcation enduring n blocks (where n must be greater than 0) be $(P_s)^n$.

Next, we must examine more closely the criteria for what has been called simultaneity. In fact, we mean 'nearly simultaneously', or 'functionally simultaneous'. We can loosely establish the bounds for this window from assumed (or measured) network latency values. Blocks produced independently within some time interval shorter than the involved network latencies (ie. between the two competing miners) will cause each miner to deviate from the optimal protocol (mine the best block available), because while the next best block available will exist at the network level, it will not be available to *them* because of the mechanical limitations of the network.

We have variables defined for this scenario.

[15] Plural forms are used to emphasize that these values are variable. Both values are difficult to measure objectively with a high degree of confidence.

λ represents network mean block time in seconds.

η represents block message latency between two nodes in seconds.

If we assume a Poisson Point Process distribution for the independent miners, what is the probability expected for 2 independent new-block events to occur within an interval η of each other?

10.2 Impact of Proposed Algorithmic Condition on Network Consensus

Conceptually, TABS relies on a fundamental assumption: capital is not distributed evenly. If we hold transaction availability to miners constant, in order for TABS to be effective, we must assume that miner balances are not equal. If all miner balances are equal and transaction/block inclusion is held constant, TDTABS does not modify the expectations of an incumbent TD-only based algorithm.

Assume a waste rate for the network of 5%. The Ethereum network's current uncle rate is 5%. We consider this a minimum value (there could exist unrecorded/unobserved uncles), but all recorded uncles are observable and valid. We assume that uncles are economically undesirable (compared to full blocks), and as such, that their existence is optimized at the lowest level mechanically possible. The occurence of an uncle signals an instant of network consensus undecidability. An uncle represents a fork. An indecisive event signals a lower state finality expectation than a decisive event. It also signals, for the miners of each twin block, a reduced expected revenue for that block (since the probability of impermanence is greater than for an uncontentious block).

We use Digiconomist data from 20211203 claiming 92.42 TWh as Ethereum's annualized energy consumption.[16]

94.42 TWh $\rightarrow MWh = 94420000 MWh$

We then cite Wikipedia's[17] data for projected LCOE by 2025 (as of 2020) for US simple average cost for advanced nuclear energy as \$81.65 / MWh.

94420000 MWh $\times \$81.65/MWh \equiv \$7,709,393,000$

If TDTABS reduces network block waste from 5% to 2.5%, that suggests an approximate annual global savings of \$192,734,825.

10.3 When Should Miners Mine on the Next Block?

[16] https://digiconomist.net/ethereum-energy-consumption/

[17] https://en.wikipedia.org/wiki/Cost_of_electricity_by_source

11 Rationale

12 Placeholder

12.1 Transaction Ancestor Hash

12.2 Another Thing

13 Theoretical Analysis

This section provides reasoning around the theoretical implications of the proposed algorithm.

13.1 Network Behavior Analysis

Demonstrate that the 'objectivity' characteristic of consensus-facing information is invariant. The additional observable data required by the proposed algorithm is chain data. It is in the same scope ('context') as data relied on for the existing canonical-preference scheme. All nodes can make the same decision given the same information at any point in time.

Need to show that consensus properties for the network are invariant.

Demonstrate that the incentive to mine the NEXT BEST block (ASAP) is preserved as an invariant.

Consider also: - theoretical additional block processing cost (*more* to do) - necessity of observable chain state (well, only balance) for block validation. Compare with requirement of Ethash for block validation. Compare with other validations, eg. Parent Hash, Number.

14 Game Theoretical Analysis

Show the equation of miner predicted (expected) revenue for some (next) block. Next blocks are uncertain, future events. We can show the probability of some miner winning the next block as the plot of the percentage chance of next-block being found over time, divided (scaled) by their relative hashrate. We show that for t=0..t=n that the expected revenue of the miner goes up.

We consider this plot for the scenario where a miner observes a next block.

14.1 Placeholder

15 Economic Analysis

16 Practical Analysis

This section offers analysis and interpretation of empirical and derived data.

17 General Proposition

We want to improve chain state finality characteristics without compromising other security characteristics of PoW network protocol.

We propose the introduction of segment-specific transactions and an accounting on a representative measure of capital saturation for some segment, yielding a "hybrid" canonical-score value.

We intend to extend the incumbent PoW canon-arbitration protocol to include reasoning about segments' relative "capitalization" rates. The introduction of such a variable can be shown to raise the expected cost of an alternative-history-based attack and to have minimal negative side effects.

We get to to keep, and still deeply rely on, an incumbent PoW emission and consensus pattern.

18 Open Questions

18.1 What if emissions revenues were separated from tx processing revenue?

18.2 Can we show that 3%/97% is a sensible simultaneous-production rate using just the Poisson emission model?

NOTE that we don't assume perfect simultanaeity; the Poisson model forbids this. We set a window we define as "competitive." We characterise this by a heterogenous acceptance of blocks at some number at the network-level; blocks produced near-enough-to-simultaneously to produce a network fork (of some scale; a point which could be additionally investigated: to what scale? 50% partisanship? 0.01%?), e.g. 300 milliseconds.

18.3 Can we show formally that it is game-optimized to mine on the next block immediately (no matter a miner's own anticipated TABS score)?

We assume under the standard Pow/Ethash game, that it is optimal to always mine on the greatest block available. However, we know that this assumption has limits; "selfish" miners with sufficient relative hashrate (greater than 25 percent) can/should be expected to optimize toward their own self interest.

Is the strategy around potential witholding of next-blocks maintained as an invariant? Is the strategy around mining the best-available block (vs. continuing work on a potential own fork) maintained as an invariant?

A miner will be able to calculate their anticipated TABS in advance (before the finalized production of their next block, depending/awaiting an Ethash solution). This leads one to wonder:
- If a miner foresees a "+" TABS block of their own (eg. they have receive in private a large transaction to process, or they are just rich),
- after an interval of 3 seconds from its parent, child block from another miner is broadcast having a "-" TABS value. This creates a block with CS=(2049*127)/(2048*128).
- Is a competitive, "rational" miner incentivized to continue mining for some period of time in the expectation of discovering a block with a comparatively greater CS?
- If they discover a block at the 8 second interval (5 seconds later), their block would have CS=(2049*129)/(2048*128).
- BUT, the competing miners will have had a 5 second "head start" on their *next* block (n+1).

I think this can be solved by finding the relative area under the PDF curve (CDF?) of the Poisson distribution for that time interval (the chance of another block appearing). This value should then be multiplied with the expected value of that next block (in CS units).

My intuition is that the CS value of a "whole" block is enough greater than the marginal differences caused by Difficulty and/or TABS adjustments that time spent toward that whole cookie is more valuable than chasing crumbs.

To demonstrate this, we can show that the CDF (?) of the Poisson distr. for lamda=13 gives a 2% likelihood to a next block at the 1-second interval. This number is totally made up. We show that 0.02 * NextBlockCS is greater than the marginal difference possible with the selfish miners competing block. So the expected value (after some very small amount of time) is higher for our potentially-selfish miner to pursue the next block instead of their own.

We could add realistic complexity by noting that our potentially-selfish miner controls some portion of the network hashrate. Their decision to pursue a selfish block/fork instead of mining the next-best block would remove hashrate from that block's potential child, thus increasing the expected interval to that child. For example, loosely: a miner with 25% network hashpower could (waves hands) extend the expected interval for the next block from a lambda = 9 seconds to 9*1.25 seconds = 12 seconds by pursuing their own next block. I have no idea if this math is reasonable. Would this mean they have a "free" 12-9=3 seconds to spend in pursuit of their selfish block? However, we should also note that the miner, pursuing independent solution, will have only 25% of the network's difficulty/hashrate calibration, and so should expect to see a block production rate averaging 25% of the network; this means expecting a slower block production rate (it will probably take them longer than 13 seconds, eg. 13*4 = 52 seconds).

To account for this last point, we can reuse the CDF logic from above; the selfish miner's expectation for block discovery in the next 1 second is small than that of the network.

$$P(\text{hps}, k) \equiv \text{probability of discovering Ethash solution in } k \text{ seconds given hps hashes per second} \tag{22}$$

- *hps* will be lower for the selfish miner than for the rest of the network (in aggregate)

19 Discussion

19.1 Ancestor Hash vs. Chain ID

Background information has been provided in this paper on Chain ID (see Section 1.4) for the purpose of contextualizing the proposed Ancestor Hash feature. There are clear similarities. In this section we show that the proposed Ancestor Hash feature is an generalized implementation of the existing Chain ID feature.

Chain ID is a single, static, arbitrary scalar value delimiting transactions per "chain". The meaning of *chain* in this context is congruent to the border drawn by ticker symbols. At least, this is the scope of which Chain ID *appears* to have been intended for.

Ancestor Hash proposes a protocol which generalizes the idea of delimiting transactions by chain. The single, static value is swapped for an arbitrary, dynamic value; and in turn, the idea of "chain" moves from a segment marked with a single, hardcoded block (genesisor a fork block), toward an idea that much more accurately represents and describes the dynamic and occasionally ambiguous growth of PoW chains.

The job that Chain ID does could also be done by Ancestor Hash. (Though this paper does not propose replacing it.) Where Chain ID values are controlled centrally, by developers (via client release defaults, usually). Ancestor Hash values are determined by transaction authors (read: chain users).

Chain IDs contain (or *can* contain) very little information compared with Ancestor Hash.

Ancestor Hashes do not force transactions authors to assume anything about the chain that they wouldn't otherwise. Or at least, to make no assumptions or compromises beyond their original, and intuitive, models.

The values that can be annotated are arbitrary. An Ancestor Hash specifying block 1.912.489 (TODO GET THE HASH) on ETH would be *more* ambiguous than Chain ID. On the other hand, an annotation citing their latest seen block represents the maximum expression of confidence (or dependence) on a specific chain chronology.

It is a way for transaction authors to declare a meaningful confidence interval (like exchanges do with confirmation delays). The delay enforced by the exchanges is an expression of doubt; the delay expresses by the transactor is an expression of confidence. (Exploring the thought experiment further: What HFC.Hash values would exchanges "like" to see on ETH for token deposits? Short ones; near the head. The nearer the transaction dependency toward the chain head, the more likely the deposit is to become *invalid* in the event of a reorganization. The denser the dependence on the chain head becomes... more gravity, more finality.)

By expressing a dependency on a specific block, transactions implicitly *also* express dependency on global chain state, and the transactions that drive it.

Another thought experiment:

I sell all my Bitcoin because Elon sold his last block. Then, in a twist of fate (is there such a thing as coincidence?), a two block reorg happens; removing Elon's sell, but keeping mine. With HFC.Hash, I could annotate the hash of the block with Elon's transaction in it. With that specific block missing from the reorg'd replacement block, my transaction won't be valid, and I won't have sold my Bitcoin like a sucker.

It can be argued that Ancestor Hashes give more control to the transaction authors in regard: transaction placement and assumptions about state.

We take it for granted that transactors make assumptions about chain state. When you send some ETH, it's not crazy to check your balance first. To check your transaction history; to make sure you're on the right nonce. These are assumptions about chain state.

Ancestor Hash transactions allow you to express these assumptions. To write them into code.

19.2 The Necessity of Ancestor Hash to the Proposed Canonical-Arbitration Algorithm

In the scenario of a malicious chain state-modifying ‘attack‘, an attacker exploits an assumption by a counterparty on chain state finality.

The fraud strategy is based on the eventual censorship of one or more transactions on which a reciprocal exchange depends.

We assume that an attacker should only censor transactions on which the attack depends.

Spurious censorship beyond the interests of the attack results in risky collateral damage (ie. an increased risk of counter-attack).

19.3 Ancestor Hash: Database Availability Impact

Theoretically, the utilization of Ancestor Hash negatively impacts the 'availability' of the chain state database to pending transactions.

In other words, this reduces the chain's transaction 'throughput'.

We argue that the extent to which database availability is restricted is not necessarily undesirable. The marginal availability constraint proposed, we argue, aligns with reasonable assumptions by a transactor about current chain state, and that the reduction in state database availability reflects potentially undesirable/spurious transaction inclusion in alternative and unobserved chain state histories.

19.4 Will miners include as many transactions per block as possible? (Is network transaction throughput optimized/invariant?)

- Assume "full blocks" (more pending transactions than available block space).
- Assume transactions come from senders with positive balances.
- Don't consider: variable balances for transaction senders.

The more transactions in a block, the greater the $TABS$ value. The greater the $TABS$ value, the greater the chance the miner authors a canonical block. The more canonical blocks a miner authors, the more money they make.

QED There exists incentive for miners to include as many transactions as possible for each block they mine. NOTE as an *invariant* with original *Modified-GHOST* protocol.

19.5 Will miners prefer transactions from senders with relatively greater balances?

Yes. But transaction gas fees and costs need to be considered simultaneously (or at least not forgotten). That is, until $B_{iTAB} > B_{i-1k}$; then it will prefer transaction gas profit, an *invariant.*

19.6 Will miners be incentivized to raise the block gas limit indefinitely?

- Assume (or remember) that given sufficient transaction volume for equilibrium fee competition, miners are incentivized to include as many transactions in a given block as possible in order to maximize gas fee revenues by raising transaction volume.
- Assume that the existing network's GasLimit has not risen indefinitely.
- Assume there exists some reasonable reason for this, like block rationing processing time

and energy, network latency optimization, social norms, mechanical ignorance of or insufficient means of execution (don't know to code it, or don't know how to code), unexplained or unfamiliar (to your authors) distributed decision-making rationales etc.

Given these assumptions, we consider only the game theoretical impacts of additional transaction inclusion in regard to $TABS$ and the role of that value in the proposed segment preference algorithm.

Point 1. GasLimit adjustment bounds can be insufficient to achieve the case.

This is because the minimum cost of a transaction is 23000 Wei. The current GasLimit is 10.000.000 Wei. The GasLimit adjustment algorithm permits the change of the incumbent (parent) GasLimit header value by +/- $B_{iGasLimit}/1024$. 10.000.000/1024 = 9765.63. 9765.63 $\langle$ 23000. If we assume an initial upward march of the GasLimit (at B_i), then before arriving at block B_{i+3} we shall encounter a block with GasLimit y ¡ parent.GasLimit - (parent.GasLimit/1024), at which point a further increase to the GasLimit shall not be sufficient to permit the inclusion of another transaction. A raise at this instant does not bring any immediate benefit or future advantage for the block author. In some cases, a raise could benefit the competiton. The game equilibrium becomes stasis.

Point 1/Counter 1. GasUsed values are composites of diverse sets of transactions. It is theoretically possible for GasLimit (at any adjustment value/rate) to be sufficient to build a case where the GasLimit should be expected to be driven indefinitely higher.

For example:

Block.Parent.GasLimit = 10.000.000
Block.Transactions.GasUsed = 9.985.900

Extra pending TX GasCost = 23000

If Block.GasLimit is raised to 10.009.765, the "Extra" transaction will be eligible for inclusion (total GasUsed 10.008.900). Scenarios like this can be built with arbitrary transaction gas costs (eg. EVM use). However, practical fabrication of scenarios like this involve centralized dominance of the pending transaction market (ie. by paying exhorbitant gas prices, by controlling miners, by suppressing entry to the public transaction pool, etc.).

If we grant Point 1/Counter 1, are the remaining unconsidered conditions also sufficient?

(Before we dive into this, we should also remember that an additional, adjacent, protocol change could be introduced capping the valid GasLimit, or capping its max adjustment value. Either or both solutions would mitigate the risk here, and could do so without noticeable effects on the status quo.)

—

It is at this point we need to consider the block production and selection business of $TABS$. How does $TABS$ shape consensus? What's its impact on segment selection?

The TAB value for any given block is – from a block *production* perspective – "incidental." Blocks may (still) be produced with no transactions, and with the block miner having a 0 balance. This is an invariant. Also invariant is that block validity depends on a unique, verifiable Ethash solution; the design and implementation of which remains unchanged. Nakamoto assumes that an optimized game sees honest, minority miners mining immediately on any next best block available to them.

We expect about 97% of blocks produced to be distributed and adopted without any observable contention. TODO CITE my experimental design and evidence. Supplementarily, we expect about 3% of blocks to be contentious (to some degree at the network level). Uncle rates typically hover between 3-6%. Rates in excess of 3% we consider to be driven by what this paper considers externalities:

- selfish mining behavior,
- variable network latencies,
- program processing time and/or priorities (eg. between various client implementations),
- and potentially variable economic conditions (eg. uncle "rewards," which once-upon-a-time made uncles nearly (if not actually) more profitable than canonical blocks themselves on Ethereum).

NOTE that GHOST (and later, *Modified-GHOST*) specifies difficulty-accounting for recorded uncles.

THIS IS SUPER IMPORTANT ¿¿ NEED TO MAKE SURE ITS WRITTEN REEAALLL GOOOD ¿

We reason that in 97% of blocks, the consensus-driving factor will remain the discovery of an Ethash solution and the prompt composition and broadcast of that solution. In these cases, the impact of the $TABS$ value in the TCP product is ineffectual; a "non-uncling" block discovered in the network mode of 8 seconds will see its adoption as canonical regardless of its $TABS$ score; just by virtue of being first to the solve.

SIDENOTE. As such, we expect the impact of the $TABS \times \mathrm{H}_d$ scoring system on overall block production rates to be small.

In the 3% of cases where honest miners generate simultaneous solutions:

- Assume the usual case where the competition is bilateral (not tri-, or quad-, etc.).
- Assume blocks have equivalent objective consensus-algorithm scoring: total difficulty, block number.

- Assume these two miners each convince 50% (by hashrate) of the network to their fork per their network connection graphs.

In these cases, a figurative coin is tossed. Each miner wins 50% of the time. We see that miners see a 98.5% "efficiency" rate against their purely theoretical, independently-considered hashrate. Another way of saying this: the network operates at 98.5% efficiency measured as time and energy spent vs. the "ideal" market value of that work. 1.5% is lost to the confusion of a competitive, decentralized network; the ethereal (and theoretical) waste heat of the network.

$TABS$ will have an impact on the outcomes of these 3% of blocks. Experimentally, $TABS$ values will supercede what otherwise would become a coin toss between 50% and 80% of the time. Why?

- Assume 50% miner hashrate has less, and 50% more, relative to the instantaneous network $TABS$ value (at some public chain HEAD (or current state)).
- Assume 0% frequency of an producing a block with a $TABS$ score equivalent to its parent. (If/when we relax this assumption, we'll see that the undecideability rate returns upwards toward that of the proposed algorithm's predecessor.)

In these cases:

- Miners each have a 50% chance of a "high" $TABS$, and likewise a 50% chance of a "low" score.

What is the chance of H:L OR L:H (AND NOT H:H OR L:L)? 50%! 2 of these 4 possible cases are decisive; 2 are indecisive.

And so, in the "flattest" (least colorful, varied... most general) case, overall miner "efficiency" will increase by 0.5 * 0.5 * 3 = 0.75% for "rich" miners, decreasing by the same amount for "poor" miners. The theoretical network inefficiency (undecidability) rate will decrease by approximately half.

NOTE FOR ALL OF THIS: An uncle rate (and the waste-heat theory) is largely a function of block emission rate. Faster blocks yield more confusion, slower emission rates see less frequent simultaneous emission events by virtue of a wider range of discrete emission times (eg. 1-99 seconds on ETH vs. 480-900 seconds on BTC). (A Poisson model for emission intervals is assumed.)

19.7 Hashes vs. ETH: Unit Cancellation in Total Canon Scoring

This section shows in math how taking the product of the Difficulty and TABS values yields results that cancel the units (hashes and ETH, respectively). This is important because there's no good way interchange hashes and token values.

—

In reality, it is likely that many of the characteristics of this general scenario are less homogenous than assumed. For example:
- Mining hashpower is not evenly distributed.
- Capital (ie. tokens) are not evenly distributed among neither miners nor transaction senders.

—

Dad's comments/questions from Thanksgiving:

Difficulty does not have units. It is just an integer.

Rounding errors? Random walk re: TABS value algo, +/- 1/128 – does this tend to median? Is it a random walk? Why 128?

—