Canonical Transaction Ordering for Bitcoin

Joannes Vermorel (Lokad), Amaury Séchet (Bitcoin ABC), Shammah Chancellor (Bitcoin ABC), Tomas van der Wansem (Bitcrust), *June 12th, 2018*

In this document, Bitcoin alway refers to Bitcoin Cash

Abstract: We propose to replace the topological transaction ordering rule of Bitcoin by the canonical transaction ordering rule, where transactions are expected to be sorted against their transaction identifiers within a block. This change eliminates an entire class of scalability challenges for Bitcoin in order to process very large blocks. This change also delivers two compelling use cases. First, it allows to produce compact proofs of transaction inclusion/exclusion, making chainless apps more capable. Second, it gives a newer degree of control to Bitcoin participants to localize their transaction within blocks.

The consensus rules of Bitcoin inherited from the original Satoshi client requires that the list of transactions listed within a block be topologically sorted from the perspective of outpoints[1] consumptions, i.e. if a transaction B depends on a transaction A to be valid, then the transaction B must appear *after* the transaction A within the block. Thus, the consensus rule enforces a *partial ordering* on the transactions.

In the following, we refer to this rule as the *topological transaction ordering* rule (TTOR). The point of the present paper is to demonstrate the superiority of the *canonical transaction ordering* rule (CTOR).

Unlike many other consensus rules, it's not clear that the TTOR was ever intended as such for Bitcoin. Most likely this rule is the unintended consequence of the early implementation of the Satoshi client itself. Indeed, a naïve implementation of the block validation consists of enumerating the transactions of the block while sequentially updating the UTXO dataset. Thus, transactions are required to be topologically ordered for this implementation to work.

We propose the CTOR as an alternative where transactions, within a block, are ordered by increasing *transaction identifier*[2]. We argue that the CTOR is highly desirable for a series of reasons, notably:

[1] In Bitcoin jargon, an outpoint is the combination of a transaction identifier along with an index. The UTXO set is a key-value store where keys are outpoints.

[2] At this point in Bitcoin, the transaction identifier is still identical to the transaction hash. However, if a change were to happen to Bitcoin where the transaction identifier and the transaction hash were becoming distinct, then the canonical ordering rule should remain attached to the transaction identifier, not the transaction hash.

- Block emission is more efficient
- Block propagation is more efficient
- Software implementations are simplified
- Proofs of transaction inclusion are improved
- Opt-in locality between participants becomes possible
- Potential attack vectors are mitigated

We will review those benefits in greater details in the following. Yet, before delving into those benefits, we need to assess whether the present TTOR may be providing subtle but important economic incentive to the network.

Finally, we will also review the alternative option the any order rule (AOR) to justify why this option is deemed inferior to the CTOR.

Incentives attached to the TTOR

The TTOR implies a partial order[3], which means that to a large extent transactions can be rearranged within a block without violating the consensus.

Considering the dominant use case of Bitcoin which is *electronic cash*, it can be reasonably expected that the average topological depth of transactions within a given block is relatively low, with probably a median depth of 1. From this insight, it can be also expected that *on average* a massive amount of transaction permutations within a block is also valid according to the topological transaction ordering rule.

Let's immediately point out that over the first decade of operations of Bitcoin, there is no clear economic upside that has been identified in taking advantage of this flexible ordering of transactions. No significant participant is known to the authors to take advantage of this "feature" to achieve any purpose of economic value to herself or to the ecosystem at large.

Also, transactions can only be successfully propagated within the peer-to-peer network if broadcasted in their topological order. The inner ordering - or lack of ordering - of the transactions within a block has no positive influence on the transaction propagation, only a negative one as detailed in the following.

In conclusion, Bitcoin would not lose any of its economic properties if the TTOR was removed from the consensus rules.

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

Asymptotic performance analysis

Implementing an *offline* solution to the TTOR is a problem known as the topological sorting problem[4]. This problem has been extensively studied. If we assume a graph is *n* nodes and *m* edges, then the problem can be solved in O(*n*+*m*) with the Kahn [1] algorithm. Using an impractically large number of processors, the problem can theoretically be solved in $O(\log_2(n))$ as pointed out by Cook [2]. In practice, using P processors would only bring a $\sqrt{P}$ speed-up at the expense of a $\sqrt{P}$ loss in overall computing efficiency.

However, from a Bitcoin perspective, it's the *online* solutions to the TTOR which are of interest. Indeed, when considering arbitrarily large blocks for Bitcoin, participants capable of claiming blocks for themselves seek to maintain an *updatable* data structure that represents a valid block template ready to be turned into a valid block when supplemented by a proof-of-work. As blocks are on average 10 min apart, it's important to take advantage of the whole duration in-between two consecutive blocks to process incoming transactions as evenly as possible, making the most of available computing resources.

Implementing an online solution to the TTOR is a problem known as the incremental topological sorting problem. This problem has been even more intensively studied than its offline counterpart. See [3] for a recent compilation, and [4] for what could be considered as state of the art results. State of the art results are built upon prior arts such as [5], [6], [7]. Essentially, while this is still an open research problem, it appears that the best data structure [4] offers $O(m\sqrt{n})$ to insert m edges in a graph assumed to be sparse. It's also not even clear how the performance will fare once distributed over P processors.

In the specific context of Bitcoin, it can be reasonably assumed that most transactions will only consume endpoints that are already buried in a previous block - in this case the transaction can appear anywhere in a block. Even if only a small fraction of the transactions - say 1% - happens to be constructed over 0-conf transactions, this small fraction will be troublesome to manage. State of the art algorithms only offer $O(\sqrt{m})$ as the best performance for incremental topological ordering. For a block of 1TB, assuming 1% of non-trivial transactions, we have at least 40 millions edges to consider, i.e. a constant factor of $\sqrt{40,000,000}$ ≈ 6325 per insertion. This constant factor is not even taking into account the performance loss which will incur as the calculation is distributed over multiple processors. In practice, we are contemplating an overhead of order 10,000 per insertion as the direct consequence of the TTOR.

Like most theoretically hard algorithmic problems, the actual real-world topology of the 0-conf transaction graph of Bitcoin would probably lend itself to powerful heuristics. For example, it is likely that the 0-conf transaction graph could be near-perfectly partitioned into disconnected components which would lend themselves to low-overhead parallelism. However, while

[4] https://en.wikipedia.org/wiki/Topological_sorting

heuristics are tremendously efficient to cope with most situations, they tend to be weak against adversarial behaviors. Again, further heuristics can probably be uncovered to securely handle such adversarial behaviors, but at this point, Bitcoin would be piling sophisticated heuristics on top of other sophisticated heuristics which is an undesirable direction for infrastructure software.

TTOR weakens the economic model of Bitcoin

The computing resources to be paid for any given Bitcoin transaction are expected to be fundamentally *local* to the transaction. Indeed, transactions are paid for through *transaction fees* which mirror this locality. However, as pointed out in the previous section, the TTOR introduces a context-dependent cost where the marginal cost for 0-conf transactions built on top of other 0-conf transactions increases as their number grows over time until a block gets emitted.

In particular, TTOR deters participants from building deep transaction graphs, which are of interest for many economic usages[5] where low latencies are expected and where the ambient trust is high, making 0-conf transactions secure thanks to a favorable context. This behavior goes somewhat against the fundamental economic model of Bitcoin. While TTOR remains a minor hindrance for Bitcoin, it cannot be considered a desirable property.

Despite the successive breakthroughs in algorithmics related to the incremental topological ordering, which happened essentially *after* the publication of the original Satoshi client, this problem remains very difficult. Considering that state-of-the-art results do not benefit from parallelized versions yet, it appears unclear to the authors whether those results will be further improved upon.

CTOR as a superior alternative

The canonical transaction ordering rule (CTOR) consists of ordering the transactions according to their transaction identifiers. An exception to the ordering rule is made for the coinbase transaction that remains at the index zero within the block. We argue that this rule is superior to the TTOR from all the relevant angles. Let's also point out that the CTOR does not impact the generation of block templates.

The offline validation of a block can be implemented as follows:

```
CtorValidate(Block B)
for each tx in B
  for each iep in OutputsOf(tx)
    UTXO.Add(iep)
```

[5] Supply chains are a good example. Bitcoin transactions could be produced as goods are moved through high-speed conveyor belts. Double-spend attacks are pointless considering that participants are well identified, and perform thousands of exchanges between each other every single day.

```
for each tx in B
  if not IsBalanced(tx) return false
  for each oep in InputsOf(tx)
     if not UTXO.Claim(iep) return false
return true
```

The `IsBalanced` function merely checks that the *bitcoins* in the inputs are greater or equal to the *bitcoins* in the outputs; however as the inputs are only known from the UTXO, this check has to be done in the second loop. This offline validation lends itself to a 2-stage embarrassingly parallel version, the first (resp. second) stage being associated with the second (resp. first) *for each* loop. The correctness of the approach is domain-dependant: the transaction graph is acyclic by construction due to the SHA256 hashing that takes place when an edge is introduced, i.e. when a transaction references another.

Then, the online version of CTOR requires to maintain a *sorted set* which delivers $O(\log_2(n))$ for insertions, with well-known implementations that deliver near perfect distribution efficiency in practice as long as the number of processors - or rather shards - is not too high.

However, CTOR yields another property which is equally desirable: blocks can now be treated as *sets* of transactions, rather than being *list*s of transactions. Indeed, as the transaction identifiers can be recomputed from the transactions themselves, under the CTOR a block is fundamentally equivalent to a set of transactions, as there is only a single valid ordering for any set of transactions.

By simplifying the data model toward a *set*, CTOR brings to the blockchain entire classes of algorithms collectively known as the *set reconciliation problem*, see [8] for an introduction. Applying those results to Bitcoin yields efficient block propagation protocols, see [9]. Also, in terms of macro architecture, Bitcoin remains - as a whole - topologically sorted at the blockchain level.

Those results are important because they allow participants to make the most of their bandwidth capacity, propagating as much information as possible ahead of time prior to the emission of a new block.

The CTOR offers the possibility for any participant to zoom into a block to identify whether a transaction is found or not without processing the whole block. This property is of high interest because *chainless*[6] apps gain the possibility to verify flows of transactions without being encumbered by an arbitrarily large blockchain.

[6] See *A taxonomy of the Bitcoin applicative landscape*, Joannes Vermorel, May 2018 (pdf)

Opt-in locality between participants

The CTOR offers a new degree of control to the participants of Bitcoin: by using a process similar to vanity hashing, albeit focused on the transaction identifier rather than the address, a payer can *localize* her transaction within a block[7]. Assuming that some opt-in guidelines are established, close participants could decide to colocalize their respective transactions within specified identifier ranges within blocks.

This property is probably one of the most valuable side-effects of CTOR for Bitcoin at large, as it offers a practical possibility to narrow down the data scope to be processed for specific use cases within Bitcoin by several orders of magnitude. While processing a large blockchain is an irreducible big data challenge, the CTOR offers leeway to the ecosystem to expand their use cases by making them more amenable to “small” data solutions.

At this point of time, malleability remains a concern for this localization process though, as adversaries might maliciously malleate transaction identifiers for the purpose of wreaking havoc on the localization scheme. A solution to this problem is already known as the MalFix[8] proposal. Let’s point out that the CTOR proposal does neither support nor oppose MalFix. We are merely pointing out potential interactions between CTOR and other proposals being made for Bitcoin.

Even if MalFix does not become part of the Bitcoin consensus rules, many prominent actors in the Bitcoin ecosystem have expressed their intent to counter 0-conf double spend attacks. Those very same actors can be expected to counter malleability attacks as well, which are fundamentally similar to 0-conf attacks.

Annex: non-deterministic block validation time as attack vector

The history of distributed computing indicates that non-deterministic performance with a complex system offers many angles to perform attacks ranging from simple denial of service attacks, exploiting a supposedly infrequent edge case leading to poor performance and putting the system at risk of downtime, to complex attacks taking advantage of faults that happen within the system caused by edge cases.

As we have seen, coping with TTOR when considering blocks beyond 10GB will require complex data structures. It is reasonable to regard those future data structures as pending security liabilities, with security problems waiting to happen. In contrast, CTOR requires only semi-trivial algorithms, which are also straightforward to distribute.

---

[7] The idea is similar to the timestamping geolocation as done in *Ansible, practical faster-than-light secure 0-conf transactions for Bitcoin*, Joannes Vermorel, April 2018 (pdf)

[8] See https://github.com/tomasvdw/bips/blob/master/malleability-fix.mediawiki

Annex: any order rule

Some of the problems relative to the TTOR could be addressed by simply removing the rule altogether. This would be the Any Order Rule (AOR). The AOR would eliminate the incremental topological sorting problem from Bitcoin, which would already be a significant upside compared to *status quo*.

However, the AOR does not yield any of the other properties which make the CTOR desirable, such as improving the block propagation, allowing compact transaction inclusion/exclusion proofs and offering an opt-in degree of control on transaction locality to Bitcoin participants.

As the implementation overhead of the AOR is essentially similar to the CTOR, we do not see any reason to favor AOR over CTOR.

References

[1] *Topological sorting of large networks*, Arthur B. Kahn, 1962, Communications of the ACM

[2] *A Taxonomy of Problems with Fast Parallel Algorithms*, Stephen A. Cook,1985, Information and Control

[3] *Practical performance of incremental topological sorting and cycle detection algorithms*, Ragnar Lárus Sigurðsson, 2016, http://publications.lib.chalmers.se/records/fulltext/248308/248308.pdf

[4] *Incremental Topological Sort and Cycle Detection in O(m sqrt(n)) Expected Total Time*, Aaron Berstein, Shiri Chechik, January 2018, https://epubs.siam.org/doi/abs/10.1137/1.9781611975031.2 (pdf)

[5] *A New Approach to Incremental Cycle Detection and Related Problems*, Michael A. Bener, Jeremy T. Fineman, Seth Gilbert, Robert E. Tarjan, February 2016 https://dl.acm.org/citation.cfm?id=2756553

[6] *A new approach to incremental topological ordering, Michael A. Bender, Jeremy T. Fineman, Seth Gilbert*, January 2009, http://www.comp.nus.edu.sg/~gilbert/pubs/SODA09.pdf

[7] *Faster Algorithms for Incremental Topological Ordering*, Bernhard Haeupler, Telikepalli Kavitha Rogers Mathew, Siddhartha Sen, Robert E. Tarjan, 2008, https://pdfs.semanticscholar.org/0b07/68807284e4237b739bf120f7fd58c98934f7.pdf

[8] *Set Reconciliation With Nearly Optimal Communication Complexity*. 2000 Yaron Minsky, Ari Trachtenberg, Member, IEEE, and Richard Zippel (pdf)

[9] *Graphene: A New Protocol for Block Propagation Using Set Reconciliation*, 2017, A. Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, Brian N. Levine, College of Information and Computer Sciences (pdf)