Authors:

George Spasov (LimeChain), Daniel Ivanov (LimeChain)

Thanks for early feedback:

Justin Drake (Ethereum Foundation), Brecht Devos (Taiko), Péter Garamvölgyi (Scroll), Simon Brown (Linea), Cooper Kunz (Aztec), Santiago Palladino (Aztec), Nick Dodson (Fuel)

This research is only possible thanks to the research grant support from Ethereum Foundation

.

TLDR

Vanilla Based Sequencing is a design for decentralised sequencing mechanism for rollups led by the L1 proposers. Its main goals are:

- Offer equal if not better UX compared to centralised sequencing in terms of preconfirmation time, guarantees and cost of transaction
- Enable the participating actors, including the rollup protocol itself, to generate revenue
- Enable composability with other rollups without the requirement to trust a single centralised sequencing layer
- Base its liveness guarantees on the Ethereum L1.
- Remove the need for trust towards a single party and ensure a sizable and diverse set of sequencers, so that within an acceptable timeframe a censored transaction will reach an honest sequencer that will be willing to sequence this transaction.
- Addresses cold-start problem of original based sequencing through naturally higher incentives for early participants.

In the design, L2 sequencing is performed by L1 proposers who have opted-in to participate as L2 sequencers and be punished for misbehaviour. The default L2 sequencer is the current slot L1 proposer. If the current L1 proposer has not opted-in to be a part of the rollup protocol, another opted-in L1 proposer is chosen at random to replace them.

The L2 sequencers have monopoly power over the rollup sequences during this L1 slot and can provide services (i.e. preconfirmations) to the users with quality on par with centralised sequencing.

Goals of Decentralised Sequencing

The majority of the current Ethereum rollup landscape consists of rollups with centralised and/or permissioned sequencing. These centralised sequencing layers both introduce trust assumptions and improve user experience.

The goal of a decentralised sequencing protocol would be to address the negatives, while maintaining or improving the positives that centralised sequencing offers.

Secure without UX Sacrifices

Current rollups with centralised sequencers offer superior UX compared to both L1 and rollups with decentralised sequencers.

Firstly, due to their constant monopoly, they offer a quick preconfirmation of inclusion and execution. This is now the standard that the users should come to expect for their transactions.

Secondly, the rollups with currently centralised sequencers fulfil the rollup-centric future promise offering orders of magnitude cheaper transactions compared to L1.

A goal for decentralised sequencing design would be to enable a rollup protocol to offer equal if not better UX compared to centralised sequencing in terms of preconfirmation time, guarantees and cost of transaction.

Economically Sustainable

Current rollups with centralised sequencer generates revenue from the transactions fees in their rollups.

A goal for a decentralised sequencing design would be to enable all the participating actors, including the rollup protocol itself, to be profitable.

Composability over Walled Gardens

Rollups with centralised sequencing cannot offer synchronous composability with other rollups and are forced to remain fragmented unless they opt into a single centralised shared sequencer. This can lead to multiple rollups having a single trusted centralised sequencer and further exacerbating the downsides of centralised sequencing.

A goal for a decentralised sequencing design would be to enable composability with other rollups without the requirement to trust a single centralised sequencing layer.

Inheriting Ethereum Liveness over External Liveness Guarantees

Rollups with centralised sequencing rely on a single centralised sequencer to be live in order for the system to be operational.

A goal for decentralised sequencing design would be to derive its liveness guarantees from the Ethereum L1 liveness guarantees.

Protocol over Trust based Censorship Resistance

Rollups with centralised sequencing requires the users to trust them that they will not censor them. This makes the centralised sequencer a single point of failure and subjects the rollup, among other things to geopolitical risks.

A goal for a decentralised sequencing design would be to remove the need of trust towards a single party and ensure a sizable and diverse set of sequencers, so that within an acceptable timeframe censored transactions will reach an honest sequencer that will be willing to sequence this transaction.

Types of Decentralised Sequencing

The major decision of a decentralised sequencing design is the selection of the next L2 sequencer(s). Two groups of design ideas are currently being explored by the rollup teams - “Free for all” and “Leader election”.

Free for all designs see the role of the sequencer as completely open at any time. In “Free for all” sequencing any user can act as sequencer and submit a sequencing transaction in L1. The order of inclusion in the L1 block of the (possibly multiple) sequencing transaction is decided by the L1 proposer and the L1 block building pipeline.

The leader election design sees a single sequencer be elected to have monopoly over the sequencing rights for some timeframe (usually denoted in L1 blocks). Two subgroups of leader election designs ideas are being explored by the various rollup researchers - “External Sequencing” and “Based Sequencing”.

External sequencing sees the leader election to be performed through an external to L1 consensus algorithm. The rollup has its own set of participants that have opted to be part of the consensus algorithm (with its crypto-economical incentives - i.e. staking) for selection of the L2 sequencer.

Based sequencing sees the leader role (L2 sequencer) be assigned to the current L1 proposer. In order to be part of the rollup protocol selection process the L1 proposers need to opt-in for additional slashing conditions for their L1 stake.

Vanilla Based Sequencing

This document outlines a design iteration over the based sequencing concept that aims to fulfil all the outlined goals of decentralised sequencing. The main difference between the original based sequencing concept and the vanilla based sequencing concept is the protocol behaviour when the current L1 slot proposer has not opted-in to be an L2 sequencer.

An important design consideration aimed by the design is to address the “cold start” problem - achieve the desired goals of decentralised sequencing even early when the participation of L1 proposers is expected to be low.

The following sections will review the three crucial design suggestions of “vanilla based sequencing” design.

L2 Sequencer Selection

Vanilla based sequencing L2 sequencer selection starts with the same general idea as original based sequencing - that the L2 sequencer is the L1 proposer. For a L1 proposer to become eligible to be a certain rollup L2 sequencer, the L1 proposers would need to opt into slashing conditions - punishment for misbehaviour as a sequencer in the rollup.

The design recognises that only a subset of the L1 proposers would opt-in to be L2 sequencer for the rollup. This nuance requires the mechanism to define the sequencer selection in the two possible scenarios:

1. Primary selection
2. When the current L1 proposer has opted-in to be a L2 sequencer for the rollup (depicted in green)
3. Fallback selection
4. When the current L1 proposer has not

opted-in to be a L2 sequencer for the rollup (depicted in blue)

[

Preconfirmations-Vanilla Based Sequencing.drawio

851×720 55.5 KB

](https://ethresear.ch/uploads/default/original/3X/6/9/69c5d356bf06fae2d6c3594caada9f40161b067a.png)

Primary Selection

In the primary selection

case the current L1 proposer has opted in to be a L2 sequencer (depicted in green above). The L1 proposer is automatically assigned as L2 sequencer for the duration of this slot by the rollup protocol. During this period, this proposer is able to provide the best security and timeliness guarantees - that the sequencing transaction(s) will get included in this block and exactly in this block (if no reorgs happen). As L2 sequencer the L1 proposer is able to get additional revenue from the transaction fees and extracted value from the rollup transactions.

Fallback Selection

One drawback of the original based sequencing concept is the appearance of “sequencing gaps” - an L1 slot whose proposer has not opted-in to be L2 sequencer. These gaps lead to an unpredictably long sequencing time - the next opted-in L1 proposer might be beyond the current L1 lookahead. Such gaps result in rollup service degradation.

In the fallback selection

case the current L1 proposer has not

opted in to be a L2 sequencer (depicted in blue above). In order to fight sequencing gaps, an L1 proposer is drawn at random from the other opted-in L1 proposers and is assigned to be the L2 sequencer. As L2 sequencer the L1 proposer is able to get additional revenue from the transaction fees and extracted value from the rollup transactions.

Unlike the primary selection case, the L2 sequencer no longer has monopoly over the L1 slot too. This necessitates that the L2 sequencer employs strategies to maximise the chances of inclusion of their sequencing transaction in the assigned slot and exactly in the assigned slot. Such strategies might include, but are not limited to, sending the sequencing transaction to the public L1 transactions mempool with an increased base tip.

Rollup Slot & Sequencing Frequency

In the diagram above we’ve equated one rollup slot - the time that a single L2 sequencer has monopoly rights of sequencing - to one L1 slot.

Depending on the usage and mechanics of the rollups, some rollups might want to temporarily or permanently lower the sequencing frequency by adjusting the rollup size to multiple L1 slots.

In a rollup slot spanning multiple L1 slots, the sequencer selection criteria stay the same but are applied only considering the last L1 slot of the rollup slot.

Requirement for Punishment Mechanism in the protocol

Similarly to any type of sequencing, vanilla based sequencing requires the sequencers to be punished for misbehaviour. Such punishment mechanism can vary and is a design decision of the rollup protocol. Several popular options are:

- Staking
- requiring the l2 sequencers to post a stake that is slashed on misbehaviour. Flavours of this can be restaking - putting forward your ETH validator stake or delegated staking - allowing other users to post stake on the sequencer behalf.
- Bonding
- requiring the l2 sequencers post a stake every time they sequence and getting it back upon finality is reached and no

misbehaviour is detected.

In the context of vanilla based sequencing, the only important requirement is for such a punishment mechanism to exist in order to disincentivise the sequencers to misbehave.

Selection Mechanics

Most of the selection mechanics are performed off-chain and are verified on-chain. This verification is part of the onchain sequencing process and is part of the logic of either the smart contract that deals with sequencing for this rollup or the finality mechanism.

First option is to have the selection performed in the L1 rollup smart contracts. The contracts need to verify the eligibility of the sequencer to be the current L2 sequencer. Due to the lack of access of L1 execution layer to the current

L1 proposer, this check needs to be performed as a separate transaction in subsequent L1 block, or an optimistic challenge mechanism needs to be employed.

Second option is for the rollup to include verification of the correct selection of the sequencer within their finality mechanism - validity or fraud proof.

In both cases, the selection can be known in advance based on the current L1 lookahead, and can be efficiently verified offchain.

The mechanisms of opting-in and selection verification are the subject of an additional research document.

Transaction List Building Delegation

Being a sequencer for one or multiple rollups increases the sophistication requirements of the opted in L1 proposers. An important factor for the quality of service for any vanilla based sequencing rollup is the high L1 proposers participation rate. Increased L1 proposer sophistication requirements and high L1 proposers participation are at a proverbial “tug of war”.

To combat this war a transaction list building delegation mechanism is proposed on top of vanilla based sequencing rollup. The vanilla based sequencing design can be fully functional without this delegation mechanism

, but will require increased sophistication of the L1 proposers.

In a MEV-boost manner, the opted-in L1 proposers are offered the ability to delegate their transaction list building to a secondary block-building pipeline.

The list building pipeline is a subject of an additional research document.

Preconfirmations

Current centralised sequencer rollups offer a superior UX compared to L1 and decentralised sequencing designs. Such a UX is now becoming a minimum standard expected by the users. One major component of this UX is the ability to quickly pre-confirm the inclusion and/or execution of a transaction to its sender. It is a naturally important requirement for the vanilla based sequencing to strive to reach and surpass the expected UX.

Two types of preconfirmations are expected of the system.

First type of preconfirmation is transaction inclusion

preconfirmation. This preconfirmation guarantees the inclusion of a transaction in the subsequent rollup slot. These are useful for use cases like simple transfers.

Second type of preconfirmation is the stronger execution state

preconfirmation. It allows specifying the desired values of parts of the state of the rollup pre or post execution of the transaction. These are useful for more complex use cases like DEX trades and/or arbitrage.

Both preconfirmation types require the sequencer to commit to the inclusion of a certain user transaction. The main difference comes in the ordering of the transactions. Within the context of the sequence, the inclusion of preconfirmed transaction can be located anywhere in the sequence. The state preconfirmation transaction requires a specific ordering of the transactions list up to this transaction.

Inclusion preconfirmations require simple checks of transaction validity - account balance, nonce, etc.

Execution state preconfirmation requires more sophistication to commit and price. Transactions prior to the target one can change the pre-execution state and make the desired post-state invalid, thus rendering the whole preconfirmation invalid. In practice this means that the sequencer must maintain and commit to an ordered list of transaction at the top of the list.

To increase the UX usability and enable wallets to hide away the complexity of retries in case of rejection or preconfirmation reneging, a deadline

field is suggested. Such a field enables wallets to retry without the user re-signing the transaction.

Both types of preconfirmations post a certain constraint on the transaction list that the sequencer can sequence. Such constraints limit to various degrees the value that the sequencer can extract from the sequence. Therefore both preconfirmations require additional payment by the user to the proposer in exchange for their guarantee.

The mechanics of preconfirmations and their pricing are the subject of a separate research document.

Rollup's Revenue

Transaction fees and MEV are the two major value sources of rollup. All of them are captured at sequencing time (assuming the sequenced transactions can be finalised).

To ensure that the protocol is generating revenue and is not forced into altruism, a portion of the sequencing revenue is suggested to be captured by the rollup. The specific proportions and mechanics are design decisions of the rollup protocols themselves.

[

sequencing-design-space-Value Flow.drawio

941×391 36.9 KB

](https://ethresear.ch/uploads/default/original/3X/0/6/06e63d56eeaa90af8ccb8d83fd2d7a7e5996f606.png)

A simple example mechanism can see the rollup embedding a commission fee Z%

over the L2 sequencer balance increase. Such commission fee is aligned with the success of the protocol, as more transactions indicate high quality service and lead to increased revenue for both the rollup and the sequencers.

Universal Synchronous Composability

The rollup design naturally lends itself to become part of a wider universal synchronous composability (USC) mechanism.

Assuming multiple rollups using the vanilla based sequencing design, USC can be achieved for the L1 slots whose proposer has opted-in to be L2 sequencer to two or more rollups. In these slots the L1 proposer becomes a shared L2 sequencer. This shared L2 sequencer can offer additional cross-rollup services like atomic messaging and super transactions.

Sequencer Violations

Regardless of the selection type, the sequencers have several ways to violate the protocol. In order to disincentivise the violations and misbehaviour the rollups are required to embed punishment mechanisms (discussed in the Sequencer Selection section). In case of violations, the sequencers are expected to be punished.

Below you can find a short list of violations and faults applicable specifically to vanilla based sequencing that the actors are expected to be punished for. This is list is by no means exhaustive and rollups should adjust it to their specific design.

Sequencer Liveness and Timeliness Faults

This violation is characterised by the L2 sequencer missing to get L1 sequencing transaction included within their rollup slot. This fault can be objectively proven by the L1 smart contract.

It is important to note that the cause in primary

sequencing can only be attributed to the L1 proposer or its delegation pipeline. Within the context of fallback

sequencing the liveness fault can be caused by the L2 sequencer's inability to guarantee the inclusion of the sequencing transaction on time due to a lack of monopoly over the L1 slot.

This difference might change the severity of the punishment of the L2 sequencer. Furthermore, this is a risk that the fallback sequencers must account for and mitigate as much as possible - for example through increased L1 base tip on the sequencing transaction.

Preconfirmation Reneging

This violation is characterised by the L2 sequencer reneging on their preconfirmation commitment. The specific way it can be proven to the L1 smart contracts is a design decision of the rollup but using a signed commitment is strongly suggested.

Requirements to be Vanilla Based Rollup

The following are two lists to help the reader to differentiate what is strongly required for a rollup to be considered using vanilla based sequencing, and what is a design decision to be made by the rollup. Both lists are likely to change over time and are subject to community consensus.

Minimal Viable Vanilla Based Sequencing Requirements

- The sequencer selection consists of the primary selection of the current L1 proposer - if opted in - and fallback selection among other opted-in L1 proposers - if not.
- Ability to provide inclusion preconfirmations
- Allows for revenue generation of the participating actors

Rollup Design Decisions

- Sequencing Frequency
- Punishment mechanism for sequencer violations
- Support for state preconfirmations
- Mechanism for discovering violations
- Source of revenue for the rollup protocol
- Deposit, finality and withdrawal mechanisms

Goals Achievement Analysis

Secure without UX Sacrifices

Vanilla based sequencing increases the security of the rollup protocol through decentralisation of the sequencers group. By involving the L1 proposers as L2 sequencers the design provides the highest possible timeliness guarantees.

A major focus of vanilla based sequencing and the ability to support UX on par if not better than centralised sequencing. This is achieved through preconfirmations and enablement of composability with other rollups.

Economically Sustainable

No actor in the vanilla based sequencing design is asked to be altruistic. All actors, including the rollup protocol itself, generate revenue for the services they are providing.

Composability over Walled Gardens

Vanilla based sequencing is a neutral design concept that is easily extendable to synchronous composability. Due to the reuse of L1 proposer as L2 sequencer, the L1 proposer can enable composability between rollups.

Inheriting Ethereum Liveness over External Liveness Guarantees

The liveness guarantees of any rollup comes from the liveness of its sequencers as a whole. Due to the reuse of L1 proposer as L2 sequencer, the vanilla based sequencing concept inherits the Ethereum liveness guarantees.

Protocol over Trust based Censorship Resistance

Rollups with centralised sequencing requires the users to trust them that they will not censor them. This makes the centralised sequencer a single point of failure and subjects the rollup, among other things to geopolitical risks.

Unlike centralised sequencing, vanilla based sequencing rollup censorship resistance increases with the increase of the set of L1 proposers opting in to be L2 sequencers. Due to their equivalence, the trust assumptions towards the L2 sequencers

are similar to the ones placed on L1 proposers themselves. The diversity of the L1 proposers group and the clear economic incentives to opt-in makes no single sequencer a long-term single point of failure and lowers geopolitical and technological risks.

Further Research Resources

- Preconfirmations - mechanics, pricing
- L2 Sequencer Opt-in Mechanics, Sequencer Discovery, L2 Sequencer Communication
- L1 PBS pipeline required modifications

[

](https://hackmd.io/JeLKP_pVQMe-_J-kpb3MkQ)