The Espresso Sequencer

- Layer 2 rollups are delivering on their promise to scale Ethereum and make it useful for a wider range of users and applications, but currently rely on centralized sequencers.
- Espresso Systems is developing the Espresso Sequencer to support rollups in decentralizing, without compromising on scale.
- The Espresso Sequencer is designed to offer rollups a means of achieving credible neutrality, enhanced interoperability, mitigation of negative effects of MEV, and long-term economic incentive alignment with L1 validators.

In November, we shared our work on the Espresso Sequencer: a platform to support rollups in decentralizing.

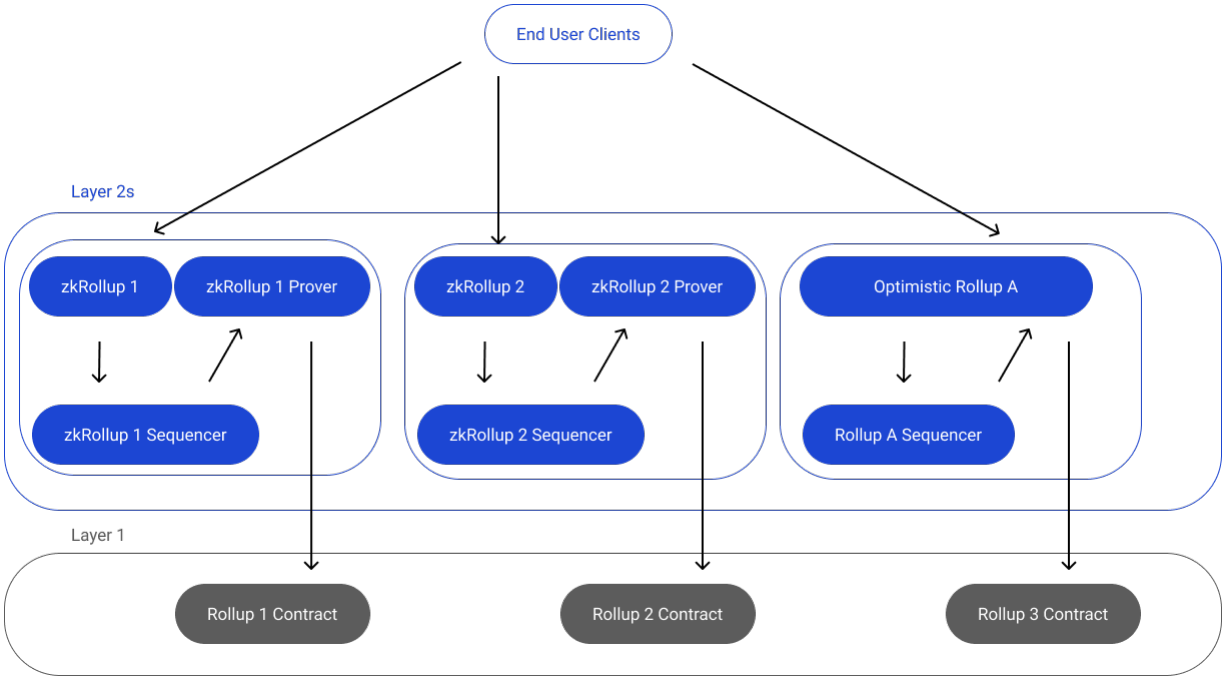
Rollup projects across the ecosystem are solving the problem of scaling Ethereum every day. Where we come in is in helping them decentralize their sequencer component to deliver on credible neutrality, security, reliability, and interoperability without sacrificing performance.

Now, we are sharing more about our goals, design principles, and how the Espresso Sequencer fits into the landscape of L2 components and architectures.

I. Introduction

A. Rollups Overview

Rollups consist of several distinct system components designed to deliver scaling benefits to blockchains. These components include: client software, a rollup-specific virtual machine (VM), one or more mempools, a sequencer, a proving system (for zk-VMs), and a rollup contract on the L1 (e.g., Ethereum).



Users create transactions with the client software of the rollup (for example, a wallet).[^1]

These transactions are submitted to the rollup and then queued in a mempool.

The sequencer component continually fetches transactions from the mempool(s), finalizes the order in which they will be executed as instructions to the VM, and ensures that these instructions are available to clients.[^2] It may also publish periodic checkpoints in this ordering (e.g., in the form of Merkle commitments) to the rollup contract on the L1. The sequencer component may be instantiated with a single server (centralized sequencer) or a decentralized network of nodes operating a consensus protocol. The sequencer can be implemented within the rollup contract, or using the L1 consensus itself (see Justin Drake's recent post, which calls this a "based rollup").

In the case of a zkRollup, a proving system executes these transactions and generates a proof of the resulting VM state transition. Typically, the prover then submits the proof, the new state root, and the block commitment to the rollup contract on the L1. The rollup then checks the proof and block commitment are valid and registers the state transition.

An alternative design, dubbed "sovereign rollup", does not use a rollup contract on the L1 to agree on the results of execution or the definition of the VM. In a “sovereign rollup” full rollup nodes directly read and execute the ordered list of transactions defined by the sequencer and compute a state change along with a proof. State roots and proofs are broadcasted to light clients for verification. A smart contract on the L1 may still be involved in implementing the sequencer, but because “sovereign rollups” do not use a contract on the L1 to define the VM this means that the rollup could have an external governance mechanism that may make changes to the VM, and thus affect the way transactions are interpreted and the results of their execution.

Transactions may affect not only the rollup VM state but also the L1 VM state, as with deposits of cryptocurrency into the rollup or withdrawals from the rollup to the L1. The rollup contract keeps the system honest by verifying proofs of state transition and serves as a bridge to the rest of the Ethereum ecosystem (e.g., by providing external smart contracts with the ability to verify the rollup state). Note, however, that a user who has directly verified the sequencer decision and rollup transition proof on its own does not need to wait on the L1 for confirmation.

This all works in a similar fashion for optimistic rollups, whether they utilize the smart contract to record a proposed state change and to arbitrate fraud proofs, or they are "sovereign", in which case fraud proofs must be gossiped among all light clients.

B. Siloed versus Shared Sequencers

The architecture of L2s that exists today sees each rollup operating within its own silo. The underlying L1 protocol to which rollups deploy contracts is shared infrastructure. Outside of this, each rollup runs its own sequencer, its own prover (for zk-rollups), and its own execution environment. This fragmentation sacrifices many of the benefits of leveraging open, composable blockchain protocols. Users operating across two or more of these rollups must suffer the cost and complexity of cross-sequencer bridges for interoperability (e.g., transferring an NFT on one rollup, programmatically contingent upon a payment in a stablecoin on another rollup). DeFi applications must contend with fragmented liquidity. Market inefficiencies arise across rollups.

A shared sequencer promises to reduce the complexity in achieving the atomicity, composability, and interoperability that users and developers seek from open, permissionless blockchains.

C. Centralized versus Decentralized Sequencers

For the most part, rollups on Ethereum today have centralized sequencers. While a centralized sequencer is highly performant, providing high throughput, low latency, and instant finality to users who trust the service, it undermines many of the benefits of running applications on Ethereum in the first place.

The "based rollup" solution to centralized sequencers lies at the other extreme altogether: a rollup might utilize the Ethereum L1 for sequencing in order to reclaim the reliability and security guarantees offered by the Ethereum validator set. This, however, is also limiting: the rollup system can at most alleviate computational bottlenecks (i.e., if transaction execution is the bottleneck for throughput) and achieve small factor improvements on communication complexity. Its throughput would still be fundamentally limited by the transaction ordering and data availability throughput of the L1. Users would also experience the same transaction confirmation delays as on the L1.[^3]

A middle ground between these two extremes is to replace the centralized sequencer with a decentralized consensus protocol that, at least under best-case conditions, provides higher throughput and faster confirmation of transaction inclusion than Ethereum natively does on its own. This protocol only needs to be responsible for ordering and availability of transactions, not execution. Separating out execution from ordering and availability can have immense benefits for throughput if integrated with an appropriately-designed consensus protocol. This approach enables removing transaction data dissemination from the critical path in consensus, meaning multiple rounds of consensus messages can be exchanged referencing only hashes of blocks whose data content has not yet been received by all nodes but is guaranteed to eventually arrive, or simply be made available. It is also possible to engage the validators of the L1 in operating this consensus protocol to increase the sequencer's decentralization and shared security with the L1 (as made possible by restaking protocols like Eigenlayer).

We can categorize approaches to sequencing as follows:

1. **Based (L1) sequencing**: same liveness guarantees as L1; high degree of decentralization; alleviates at most computational bottlenecks; throughput still limited by data sequencing of L1.
2. **Non-based sequencing**: encompasses many approaches, including today's centralized sequencers; may not entirely depend on the L1 alone for security.
3. **Non-based sequencing with L1 participation**: shared security with the L1 (e.g., via restaking); can achieve strong decentralization and high scalability in optimistic conditions.

D. The Espresso Sequencer: A Decentralized Shared Sequencing Layer

Motivated by the problems and tradeoffs of existing L2 architectures, we are introducing the Espresso Sequencer: a decentralized sequencing network for rollups that is designed to provide secure, high throughput, low latency transaction ordering and availability.

We envision the Espresso Sequencer as a shared public good for both optimistic and zk-rollups. This utility will not only allow rollups to decentralize, but will also introduce to rollups the interoperability advantages of a shared sequencing layer.

We intend for the Espresso Sequencer to engage the participation of Ethereum validators through restaking contracts for shared security with the Ethereum L1.

The following sections will present the advantages of decentralized and shared sequencing for rollups, the design philosophy of the Espresso Sequencer, and how it fits into the broader rollup ecosystem. Future posts will delve deeper into technical details of the Espresso Sequencer design.

II. Espresso Sequencer: Motivations

A. Resilience & Neutrality

Centralized sequencers are a single point of failure for rollup liveness.

These dynamics sacrifice the resilience and the credible neutrality of the system. Some designs feature a centralized sequencer combined with an "escape hatch" mechanism, whereby users can circumvent omission of their transactions by posting transactions directly to the L1. While this is technically a form of decentralizing the sequencer, it still allows for price discrimination, as transacting on the L1 via the escape hatch can be much more expensive, failing to adequately solve the problem. This brings us to the second point.

B. Anti-Monopoly

If centralized sequencers are rational economic actors, they may price gouge users.

For example, centralized sequencers could set prices nearly as high as escape hatch transactions (the only alternative). As rollups accrue usage and host an ecosystem of apps, the switching costs for users will grow and centralized sequencers will gain monopolistic influence over pricing. Centralized sequencers are rationally motivated to maximally extract value (MEV) from users both directly (e.g., via fees) and indirectly (e.g., via frontrunning, sandwich attacks, etc). A system that creates natural counter-balances to monopolization results in better experiences for users than systems relying on the goodwill of any one actor.

C. Supporting Fair Ordering and Private Mempools

Decentralizing the sequencer is essential to lowering the impact of MEV on users.

Centralized sequencers are effectively monopolists and can unilaterally decide to only accept transactions from the public mempool. Rationally, they would only cooperate with a private mempool that on average extracts more value from users than it would on its own.

Techniques such as fair-ordering protocols or private mempools are only able to mitigate negative effects of MEV in the context of decentralized sequencing layers. Fair-ordering protocols rely on honest-majority assumptions, which only make sense in a decentralized setting. Meanwhile, the promise of private mempools (e.g., proposals like MEV-Share) to reduce the impact of MEV depends upon the existence of competition among nodes proposing blocks across different time slots, giving users greater leverage and preventing monopoly pricing. See section IV.B.2 for further discussion.

D. Decentralization without Fragmentation

Rollup interoperability (cross-rollup bridging and cross-rollup atomic transaction execution).

A sequencing layer shared across multiple rollups promises to make cross-chain messaging and bridging cheaper, faster, and safer. Elimination of the need to build a light client for another chain's sequencer is a free benefit with no cost, creating potential upfront savings. Ongoing savings are also made possible for cross-rollup bridging by removing the need for a given rollup to stay independently up-to-date with other rollups' consensus. Shared sequencers also offer security benefits for bridging: the shared sequencer can guarantee that a transaction is finalized in one rollup if and only if (or even at the same time) it is finalized in the other.

Additionally, shared sequencers enhance users' abilities to express atomic dependencies between transactions across different rollups. Conventionally, Alice would sign and publish her rollup-A transaction t independently of Bob's rollup-B transaction t'. In this case, Alice's transaction might get sequenced long before Bob's, leaving Bob with a long-duration option to abort (e.g., a trade). This optionality imbalance is mitigated with a shared sequencer, where Alice and Bob can instead submit the two transactions together as a signed bundle (i.e., the sequencer must treat these two transactions as one). Combined with proposer-builder separation, this enables many new applications, such as cross-rollup DEX arbitrage, and effectively combines liquidity between rollups sharing the sequencer.

Shared sequencing increases revenue for rollups.

Improved bridging and interoperability offered by shared sequencing can increase the value that rollups offer to users, translating to higher revenue for rollup ecosystems. There are transactions that might have no value independently on each rollup, but have high value if executed atomically across all rollups. In such cases, users might be more willing to pay higher fees for a system that guarantees this atomicity than the sum of fees (possibly zero) it would pay to have these transactions processed by independent rollup-sequencer systems.

This increased revenue can be expected to flow back to the stakeholders of the individual rollup ecosystems. (The stakeholders of a rollup ecosystem refer to the set of parties that have the power to decide upon a sequencing layer for the rollup, whether by appointment or some form of decentralized governance).

E. Preventing Paths Toward Centralization

Mitigating "superlinearity" of cross-domain staking and centralization.

There is a concern that cross-domain MEV leads to greater centralization of stake. Validators who participate in two independent PoS consensus protocols for independent domains (i.e., blockchains, rollups, etc.) can take advantage of lucrative cross-domain MEV opportunities when they are simultaneously elected to propose blocks in each. If they control a p fraction of stake in two different domains then the probability they are elected to both simultaneously in a given slot is p^2, and thus their return on staking is superlinear, which disproportionately favors staking larger amounts, encouraging greater centralization. A shared consensus protocol for sequencing across all rollups on Ethereum sidesteps this problem for rollups.

F. Economic Incentive Alignment with L1 Validators

In a centralized sequencer, nearly all of the rollup value (e.g., fees, MEV) is likely to be captured by the sequencer. If none (or relatively little) of the value generated by a rollup is captured by the L1 validators, then there is a concern that this will destabilize the security of the rollup. This is for the simple reason that the L1 validators can be bribed to fork the rollup smart contract state, and in doing so profit more than they would managing the rollup contract honestly. Decentralizing the sequencer and empowering L1 validators to participate in operation of the sequencer mitigates this concern.

III. Espresso Sequencer: Design Principles and Requirements

A. Complementing Ethereum

The ideal non-native sequencing layer should complement the trade-offs inherent to Ethereum's consensus, which prioritizes liveness under pessimistic conditions rather than responsiveness under optimistic conditions. A non-native sequencing layer shouldn't mirror the same properties as Ethereum's Gasper protocol as this would provide little advantage over using the L1 directly for sequencing.

Ethereum's Gasper protocol achieves dynamic availability, the hallmark achievement of longest-chain consensus protocols, which is the ability of a protocol to remain live under sporadic participation, even if most nodes at any given time are offline. Fast finality, or optimistic responsiveness, is the ability of the protocol to confirm transactions as fast as the network will allow. Confirmation can be nearly instantaneous under optimistic network conditions. This stands in contrast to protocols in which the confirmation delay is tuned to worst-case network conditions, or where transactions are only probabilistically final. Consensus protocols must choose between optimistic responsiveness and dynamic availability—these two properties are incompatible. Most practical BFT protocols to-date — including Tendermint and Casper—achieve neither property.

B. Supporting Scale

Rollups have been invented, and found strong traction, in response to user demands for faster throughput and lower fees. Therefore, any proposal to decentralize rollup sequencers that materially sacrifices on scalability is unacceptable as it negates the value proposition of using a rollup in the first place. In terms of performance, the Espresso Sequencer aims to improve upon proposals for decentralizing rollups that rely on the L1 consensus system for transaction sequencing and data availability.

At the core of the Espresso Sequencer is a consensus protocol,HotShot, that prioritizes high throughput and fast finality over dynamic availability. This protocol builds on HotStuff combined with the latest developments in Pacemakers, verifiable information dispersal (VID), random beacons from verifiable delay functions, and ideas from DAG-transport layers. A critical requirement of HotShot is to achieve strong performance characteristics without compromising on the size of the validator set. At a minimum, it should scale to include the participation of all Ethereum validators (currently estimated at around 500k unique staking addresses and 14k unique physical nodes).

C. Strong Security

Given that a foundational goal of the Espresso Sequencer is to remove the need for trust in a single, centralized sequencer, our design choices revolve around creating strong security guarantees.

1. Security in the Presence of Bribery

A common scaling method for BFT protocols is to randomly sample smaller committees to participate in voting using a combination of random beacons and verifiable random functions (VRFs). The probability that a node will be included in the committee for a given epoch is proportional to the node's ownership of stake. This is done in protocols like Algorand and Polkadot. However, committee sampling lowers the security of the protocol if the malicious party is able to bribe committee members to change their votes. The cost of corrupting a small committee is much lower than the cost of compromising a significant fraction (e.g., 1/3) of all the nodes/stake in the system. To mitigate this, committee members can be elected to broadcast a single message and VRFs can hide their identities before election, preventing the adversary from knowing who to bribe (player replaceability, or YOSO). We do not consider this a sufficient defense against bribery because the adversary can simply advertise the bribe, and even set up an Ethereum smart contract that will reward committee members for signing conflicting votes. We avoid relying on bribery-vulnerable methods like committee sampling and achieve efficiency by other means.[^4]

Achieving optimistic responsiveness and optimistic linear communication requires a leader-based consensus protocol.[^5] The protocol advances through a series of views, each of which determines a unique leader responsible for broadcasting messages and collecting votes. Bribery attacks can thus be used to indefinitely stall liveness, even when leader selection is unpredictable. The adversary simply rewards leaders for proposing empty blocks. Eventually the adversary may exhaust its budget, but this could take a long time. To shorten the duration of such attacks we can employ an asynchronous fallback protocol that uses all-to-all broadcast. The mere existence of the recovery path also lowers the utility of the attack, as the adversary would be spending a large budget only to slow (and not stop) consensus. This serves to dissuade adversaries from attacking liveness in the first place, lowering the likelihood that this slow path will ever be triggered.

We reflect this principle in our approach to data availability (DA) as well. The standard requirement to disseminate all transaction data among every node participating in consensus before a decision can be reached presents a fundamental bottleneck in the throughput of sequencing (i.e., consensus) protocols. Some designs therefore make a performance/security tradeoff by outsourcing data availability to a smaller set of trusted nodes. This set of nodes, called a DA committee, may also be randomly sampled from the larger staking set and rotated every epoch or even every block. The committees can continue to propagate data so that all nodes eventually catch up, but off the critical path of consensus progress. This increases consensus throughput by leveraging the fact that data propagates at higher throughput over a longer latency window, but can quickly become the consensus bottleneck if required to reach every node inside the short latency window in which a decision must be reached. An alternative solution is VID, which verifiably distributes erasure code shares of the data among all nodes participating in consensus such that any subset of nodes owning a sufficiently high fraction of the stake would be able to cooperatively recover the data. VID drastically reduces the broadcast communication complexity required to make a block of data available (asymptotically removing dependence on the number of nodes), but greatly increases the complexity of data recovery.

We do not consider committee-based DA solutions an acceptable performance/security tradeoff. The small committee initially responsible for availability of data at the moment a block is finalized can be bribed at relatively low cost. On the other hand, relying on VID solely for DA is far from pragmatic. Our approach thus composes these two methods to achieve security and pragmatism. Optimistically, high bandwidth nodes will make data available to all other nodes, and the availability of each individual block is also backed up by small randomly (unpredictably) elected committees. (By using a careful sampling method with flexible threshold parameters, the committees can optimistically be very small.) But given the risk of DDoS and bribery attacks on small committees, we use VID to provide a reliable (yet slower) backup path that guarantees availability as long as a sufficiently high fraction of all nodes (weighted by stake) is not compromised.

2. Engaging the Ethereum Validator Set for Security

Engaging the entire Ethereum validator set itself to run the Espresso Sequencer protocol is compelling for a number of reasons:

1. The sequencer is reliant on the same set of nodes as Ethereum for security (consistency/liveness). Only the protocol changes, and with it the properties achieved (e.g., finality vs availability). In fact, Ethereum already effectively runs two protocols, Casper FFG and LMD Ghost, which work in concert to maintain a dynamically available chain with a finalized prefix. Engaging the Ethereum validators in running the Espresso Sequencer thus modularly adds to the components of Ethereum itself in order to better support the rollup ecosystem without changing the security model. Otherwise, users would need to trust a new set of validators for both the liveness of the rollup ecosystem and for fast confirmation/finality.

2. If none (or relatively little) of the value generated by a rollup is shared with the L1 validators, then this risks destabilizing the security of the rollup. For example, the L1 validators can be bribed to fork the rollup smart contract state for greater profit than they would earn managing the rollup contract honestly.

D. Compatibility with Proposer-Builder Separation

The Espresso Sequencer takes on the role of sequencing transactions and proposing blocks. By design, the Espresso Sequencer is compatible with the paradigm known as proposer-builder separation (PBS) in which users submit transactions to a private mempool, after which a trusted third party (TTP) assembles a bundle of transactions and obtains a blind commitment from a sequencer node (i.e. miner/validator) to propose the block. The TTP is a theoretical abstraction that might be replaced using cryptography or trusted hardware, although in practice today it is generally run by an actual trusted party (e.g., Flashbots). The proposer (in this case the sequencer node) only knows the total utility it will get from proposing the block without knowing the contents.

The Espresso Sequencer is designed to be complementary to such TTP-operated private mempool services that can act as builders, with the sequencer nodes as the proposers.

E. Summary of Requirements

1. **Optimistic responsiveness**: With optimistic responsiveness, the system can leverage highly scalable "Web2" architectures (e.g., high bandwidth nodes operating NGINX instances) to optimistically achieve extremely high throughput, while in the worst case still sustaining liveness over a high-resilience gossip-based fallback path. This achieves the best of both worlds: Web2 performance with Web3 security. This rules out protocols like Tendermint.

2. **Optimistic linear communication**: With the goal of scaling to tens of thousands of nodes, keeping communication complexity linear on the optimistic path is critical. This not only rules out PBFT and DAG-based consensus protocols like Bullshark or Tusk. These protocols have best-case quadratic communication complexity and do not scale to such large participation sets.

3. **Bribery resistance**: The system must avoid dependence on bribery-vulnerable methods like committee sampling for security-critical components.

4. **Asynchronous recovery**: To shorten the duration of bribery attacks (or other attacks on liveness) we can employ an asynchronous fallback protocol that uses all-to-all broadcast.

5. **Ethereum restaking**: Both for robust security of the sequencer itself and to align incentives with L1 validators, we engage the Ethereum validator set through Ethereum restaking.

6. **Compatibility with PBS**: In order to support MEV-mitigation approaches, the system must be compatible with proposer-builder separation. In this case, the sequencer nodes play the role of proposers.

IV. Building the Espresso Sequencer

A. Espresso Sequencer Components

In order to achieve these goals in accordance with our principles, we have architected and are implementing a system of several components.

The Espresso Sequencer consists of:

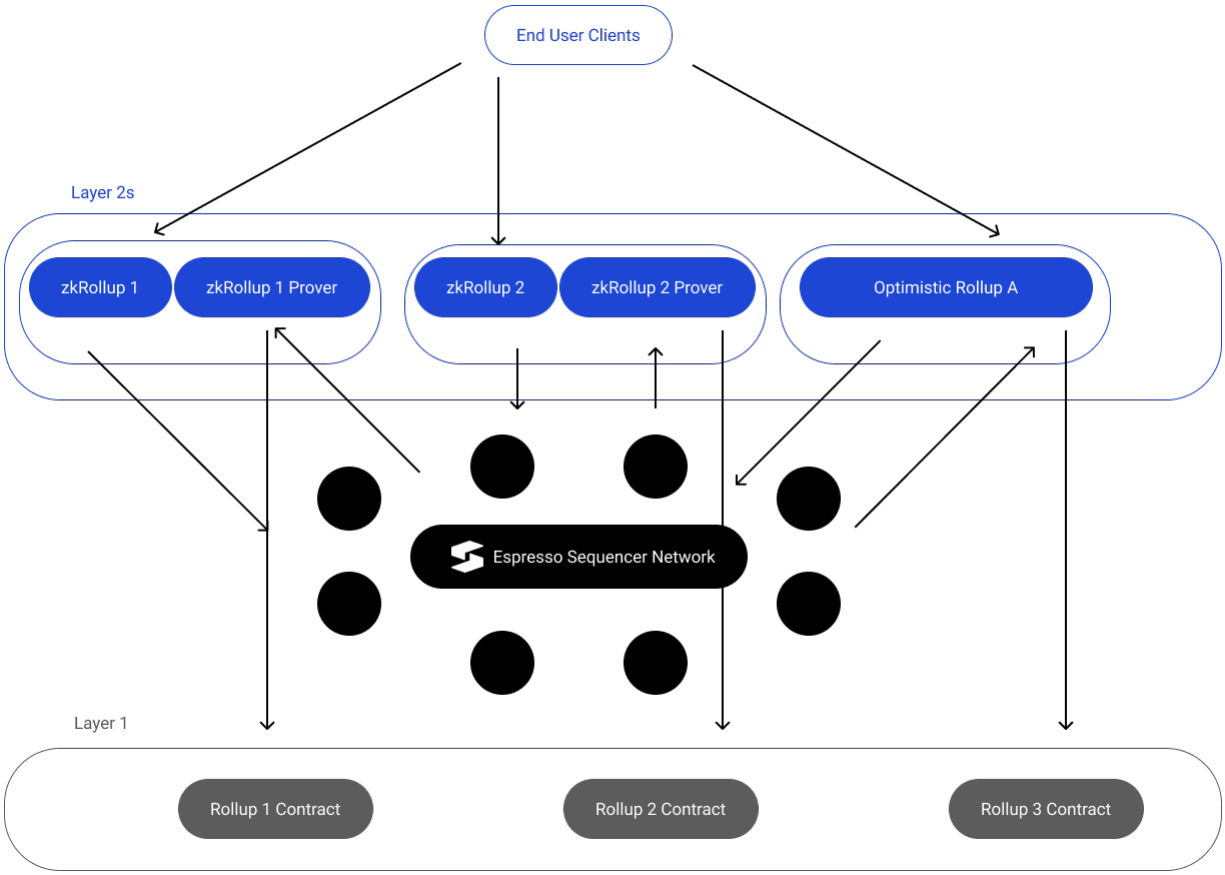
- **HotShot**: A consensus protocol that decentralizes participation in the sequencer network, offers high throughput and fast finality in optimistic conditions (responsiveness), while maintaining safety and liveness guarantees under pessimistic assumptions. Our consensus protocol is called HotShot and is based on the HotStuff protocol. While the original HotStuff protocol was described for only a fixed set of validators, HotShot is open and permissionless. Differences include a new Pacemaker (view-synchronization method), a VDF-based random beacon for leader rotation, a dynamic stake table, and distributed signature aggregation. The design of each of these HotShot components are constrained by the goals of maintaining strong adaptive/bribery-resistant security (e.g., ruling out committee-based approaches), while still scaling to support a massively larger operator set than what the HotStuff protocol was originally intended to run on.

- **Espresso DA**: A data availability solution that is highly scalable and robust. This integrates elements of VID with DAG-based reliable mempools. We optimistically enable fast data retrievability via small committees, but also include a reliable (yet slower) backup path that guarantees eventual retrievability in the worst-case scenario. The Espresso DA produces certified-available blocks that HotShot block proposals can reference instead of the original block data.

- **Rollup REST API**: An API that L2 rollups use to integrate with the Espresso Sequencer.

- **Sequencer contract**: A smart contract that verifies HotShot consensus (i.e., as a light client) and records checkpoints (i.e., cryptographically binding commitments to points in the growing log of ordered transactions). It also manages the stake table of HotShot.

- A networking layer that enables communication of transactions and consensus messages between nodes participating in HotShot and the Espresso DA.



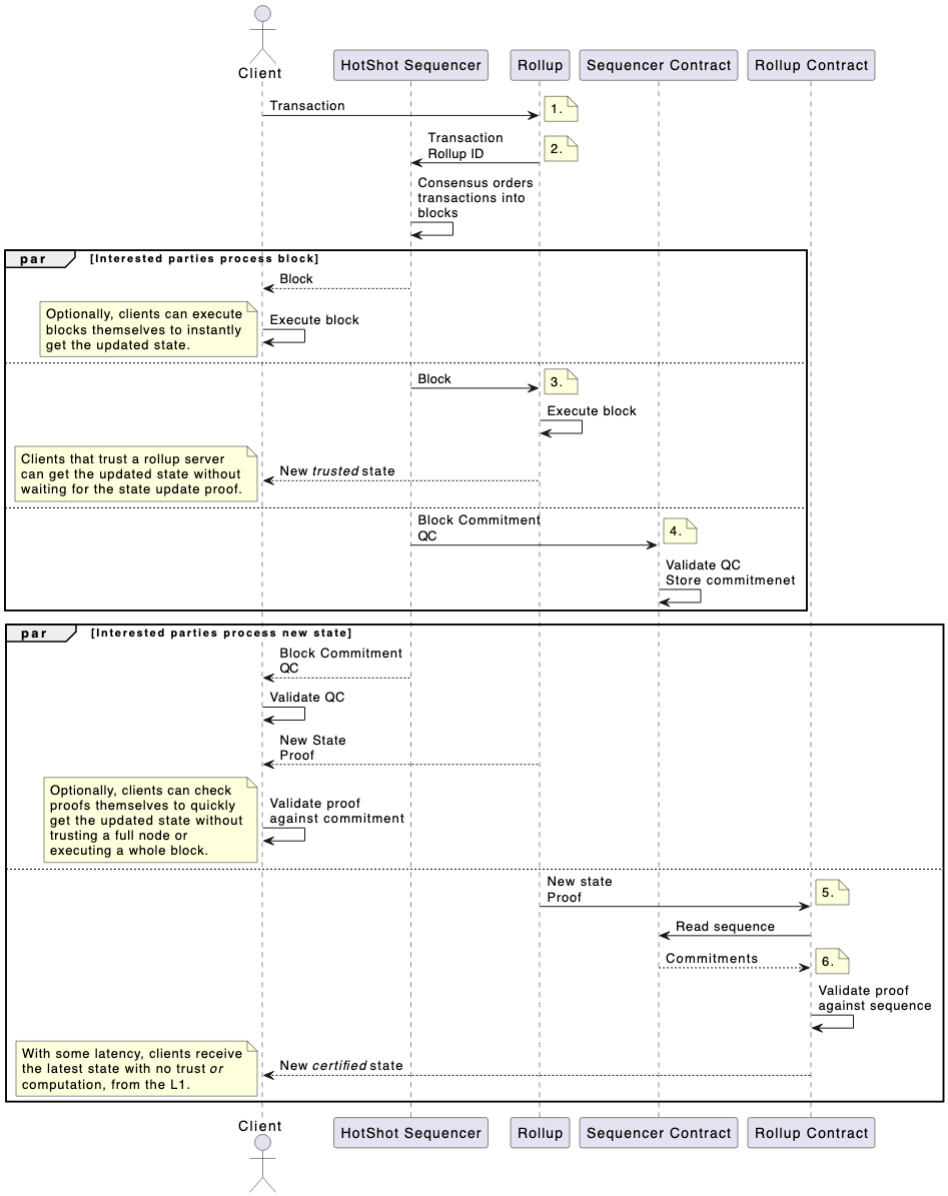
B. Interactions With The Ecosystem

1. Interactions With L2s

The Espresso Sequencer is designed to support arbitrary L2 rollups as a shared, decentralized sequencing layer.

For a detailed description of the Espresso Sequencer architecture and interfaces, see our documentationhere.

The diagram below shows in more detail the flow of a single transaction from a client through the system.



The start of the transaction flow is the same as the way rollups operate today: users create and submit transactions to the rollup. These transactions are gossiped through a network of sequencer nodes and remain in mempools. The leader is then denoted via the HotShot proof-of-stake mechanism, proposes the block, and propagates this back through the rollups' executors and provers before it is finalized in the contract on the layer 1.

The leader sends transactions to the data availability committee and receives a DA certificate in return. It also sends a commitment to the block to the layer 1 sequencer contract, along with a quorum certificate that the contract uses to authenticate the block. This allows the layer 1 rollup contracts to compare the rollup state update proof against a block commitment which is certified as being the output of consensus.

Note that clients have several options for checking the finality of their transactions and querying the updated state. They can leverage HotShot's low latency to achieve fast finality by trusting a rollup server that has executed the transaction before a proof is generated, or trustlessly by executing the transaction themselves. They can also wait slightly longer for a proof to be generated and check that proof. This requires less computation than executing the block, and is still trustless. Finally, if a client does not want to do any computation on their own and does not want to trust a rollup server, they can wait until a state update is certified by the L1 to fetch the updated state with no trust or computation. This shows how the system leverages different properties of HotShot consensus and the layer 1 consensus to provide an ergonomic, flexible user experience.

2. Interactions With Block-Builders and Private Mempools

a. Decentralized Block-Building vs Sequencers

Decentralized block-building designs like SUAVE aim to decentralize the TTP. These designs carry great promise to mitigate the impact of MEV on users, but as we argue here, only on top of a platform for block-proposing and sequencing that is already decentralized. Therefore, we view the Espresso Sequencer (and the existence of decentralized sequencer platforms more broadly) as an essential and complementary component required to combat the negative externalities of MEV on the blockchain ecosystem.

b. Centralized Sequencers Exacerbate MEV

PBS is ineffective at mitigating MEV if the sequencing layer, which controls block proposals and ultimately decides on block inclusion and order, is centralized. A sequencer controlled by a single node is effectively a monopolist. Based on its knowledge of users' willingness to pay, a monopolist would set prices (including all forms of MEV) to maximize profit. Without the ability to price discriminate, this may amount to denying service to users unwilling to pay the price even absent congestion. As a part of this strategy, the monopolist should ignore blocks/bundles produced by the TTP unless they demonstrably offer it higher rewards on average than it could extract on its own (by forcing users into the public mempool). In other words, a monopolist will only cooperate with a private mempool that on average extracts more value from users than the monopolist would on its own. Thus, a private mempool running an MEV auction can only worsen the impact of MEV on users of a centralized sequencer.

On the other hand, a decentralized sequencing layer can support PBS to mitigate MEV.

c. Decentralized Sequencers Mitigate MEV

Decentralized consensus protocols that rotate block proposers (leaders) randomly and unpredictably among participating nodes can protect users frommonopoly pricing. For simple fees (e.g., when a user offers a fee payment that will be collected by the block proposer if and only if it includes the transaction in its proposed block) this is easy to see. Suppose that based on user willingness to pay, the profit-maximizing reserve price is $10 per transaction (i.e., a monopolist would maximize profit by setting a minimum fee of $10, denying service to users unwilling to pay this price). A single leader (monopolist) would thus reject all transactions in the mempool offering less than a $10 fee. However, in a decentralized protocol this is an unstable strategy for all participating nodes to agree on, particularly if the randomly elected block proposer is rewarded most (if not all) of the fees from its block.

In this scenario, half of the nodes could form a coalition A that starts including transactions for $9 fees while the remaining nodes B continue to exclude them. Knowing this, patient users willing to wait on average two blocks instead of one will lower their offers to $9. Suppose half of the users do so. Due to randomness and unpredictability, on average sets A and B will process the same number of transactions from anxious users paying $10, but only A will process transactions from patient users paying $9. Thus, while the combined earnings of A and B would be lower due to deviation from the monopolist strategy (i.e., $9.5/tx versus $10/tx), the set A is now earning more than before on average (i.e., $7/tx versus $5/tx) while B takes the loss (i.e., $2.5/tx versus $5/tx).

Note the importance of encouraging price competition by distributing some portion of fees to the block proposer. In contrast, allocating block fees equally among all consensus participants (e.g., by burning the fees in a proof-of-stake system) encourages monopolistic behavior because there is no individual benefit to a party deviating from the monopolist strategy.

When it comes to more complex forms of MEV, however, decentralization alone does not save users from monopoly pricing. A block proposer in any slot is incentivized to extract as much value as it can from all transactions available in the mempool via frontrunning and other order manipulation strategies. (The proposer may not know how to do so optimally on its own, but can auction off this opportunity to MEV searchers, effectively achieving the same outcome.) In other words, the profit-maximizing monopolist strategy changes and all block proposers are incentivized to follow it. Unlike the case of reserve prices for simple fee extraction, in the case of MEV, a coalition gains no advantage by deviating from the monopolist strategy. Two ways to address this are:

1. **Bidding negative fees**. If the user submitting a transaction were able to perfectly predict its marginal contribution to the proposer's total MEV then it could bid a negative fee, effectively requiring the proposer to pay the user for including the transaction in a block. In this case, the analysis of proposer behavior is identical to the case of simple fees, and intertemporal competition should protect users from monopoly pricing.
2. **TTP-operated private mempool (PBS)**. The monopolist strategy in this case is to reject all blocks coming from private mempools offering lower value on average per block than the monopolist could extract on its own. However, similar to the case of reserve prices, this is no longer stable in a decentralized system. If all proposers agree to reject bundles coming from any TTP, a coalition will gain an advantage by deviating, cooperating with a TTP offering slightly lower value than the default public mempool. The analysis is identical to the case of simple fees.

The first method is largely theoretical, as users are not able to perfectly calculate (let alone predict) their marginal contribution to MEV.

There are many ways to instantiate the TTP-operated mempool, including with an actual trusted third party (e.g., what Flashbots functions as today), with a distributed set of nodes and an honest-majority assumption (e.g., fair ordering protocols, threshold encryption, multi-party computation), and with trusted hardware (e.g., SGX). An orthogonal consideration is whether the TTP simply orders transactions in a neutral way (e.g., FIFO) or internally runs an MEV auction to approximate the MEV-optimal order, and then determines a split allocation of the auction revenue between the proposer and users who contributed transactions. For example, if the revenue is R, it can determine (via a truthful mechanism) each user's utility loss or gain between the FIFO order and the MEV-optimal order, allocate $x \leq R$ back to these users proportional to their loss, and offer $y = R - x$ to the proposer. The advantage of doing so is the ability of the TTP to offer any $y \leq R$ to the proposer, whereas the TTP doing a FIFO ordering may not be able to offer a sufficiently high reward to destabilize a monopolistic proposer coalition. On the other hand, with free entry and users' ability to assign negative fees to transactions in the MEV auction, the revenue could approach zero in equilibrium. Furthermore, for any proposer coalition accepting bundles from TTPs offering $y \leq R$, there is a sub-coalition that may gain from accepting bundles offering $y' < y$.

3. Interactions with Ethereum Validators via Restaking

As discussed in Section III. D, there are several advantages to engaging the Ethereum validator set to run the Espresso Sequencer protocol. This amounts to modularly adding to the components of Ethereum itself in order to better support the rollup ecosystem but without changing the security model. This also evades potential paths towards centralization by sharing the value generated by rollups with L1 nodes.

ETH restaking is an emerging concept that promises to allow ETH that is already staked for Gasper to be restaked for additional participation in the Espresso Sequencer. A node that restakes ETH it has already locked for Gasper is subjecting this ETH to additional slashing conditions from the Espresso Sequencer contract, and will thus stand to lose this ETH if it violates these conditions (even if it does nothing wrong within Gasper). Likewise, it can still lose the ETH if it violates the slashing conditions of Gasper, and its influence (voting power and frequency of proposing) in each protocol would be adjusted accordingly. Eigenlayer is an implementation of ETH restaking contracts currently under development.

ETH restaking has many potential advantages: minimizing operational complexity in onboarding Ethereum validators; creating aligned incentives to achieve high levels of engagement from validators; and accounting for Ethereum's dynamically changing validator set. Restaking is an emerging topic with many possible models and complex implications and considerations for each. In particular, we are researching approaches to mitigate issues arising from exclusively relying on ETH restakers to run the system.

These issues include:

- High cost of capital to participate (requires staking at least 32 ETH)
- If initially the new system is generating significantly less value for restakers than Ethereum itself, then, due to non-linear utility of money, the marginal utility increase that comes from restaking ETH may be very small.
- Similarly, the marginal utility loss to restakers if the new system loses liveness may be very small, presuming that only accountable actions (e.g., safety violations) are slashed. (Slashing nodes for being offline, missing proposals or voting rounds etc, is inadvisable as it can be exploited through DDoS attacks).

This is an active area of inquiry for us, and we are glad to be engaging with researchers and across the industry on this matter.

V. Summary and Roadmap

At Espresso Systems, we have been inspired by the rapid research and development efforts of all the teams working on rollups as scaling solutions for blockchain developers and their end users.

We are developing the Espresso Sequencer to support rollups in achieving decentralization, without sacrificing on their performance benefits. The design of the Espresso Sequencer considers a range of goals:

- Resilience and neutrality of the infrastructure
- Anti-monopoly dynamics to encourage price competition for users' fees
- Support for fair ordering and private mempools
- Cross-rollup interoperability and composability
- Mitigation of paths toward centralization over time
- Economic alignment of L1 validators with rollup systems

To deliver decentralization without giving up scale, the Espresso Sequencer leverages a purpose-built, optimistically responsive consensus protocol based on HotStuff we call HotShot. The Espresso Data Availability solution (Espresso DA) is highly scalable and robust, optimistically enabling fast data retrievability via small committees, but also including a reliable (yet slower) backup path that guarantees eventual retrievability.

In November, we released the first testnet for the Espresso Sequencer:Americano. This release represented an internal-only testnet and our initial set of benchmarks for the optimistic network path. Though this release did not yet integrate the more advanced techniques on our roadmap to scale throughput, it already achieves a high baseline of performance in "good" network conditions. In our benchmarks, we measured throughput in network configurations of 10, 100, and 1,000 nodes with fixed stake distributions. At 100 nodes, we achieved a throughput of 2598 kb/s which maps to 10,000–20,000 ERC-20 transfers per second.

Americano leverages a centralized network architecture to exploit the benefits of optimistic responsiveness. As a reminder, on account of this property, a centralized architecture helps increase throughput in optimistic conditions, but does not impact safety! With our next testnet, Doppio, we plan to further optimize the optimistic path (e.g., with load-balancing, parallelism, and even higher-bandwidth centralized servers) while integrating this with a robust gossip-based fallback path, which helps to retain liveness in pessimistic conditions. We will also integrate a data-availability mechanism to move data (i.e., full block) dissemination off the critical path of consensus.

The Doppio testnet will also take the first steps toward supporting live rollups. If you are working on a zk-VM or optimistic rollup and are interested in leveraging the Espresso Sequencer to decentralize your layer-2 scaling solution, please reach out to us here.

Roadmap

You can learn more about the Espresso Sequencer by visiting our documentationhere, which we will be continuing to update in the upcoming weeks.

There is a lot left for us to research and build and we welcome conversation, feedback, and collaboration from the community. You'll find us discussing all this and more in the replies of Twitter threads and in the comments of forums like ETHResearch. Please join us in our Discord and on Twitter to chat with us about the proposals we have laid out here and the systems we have begun to build.

Thank you to Sreeram Kannan, Shea Ketsdever, Soubhik Deb, Quintus Kilbourn, Alejo Salles, Tarun Chitra, Jon Charbonneau, Dahlia Malkhi, and Joseph Bonneau for the conversations that led to many of the ideas and plans in this post and for the valuable feedback they have provided.

[^1]: There is one public mempool, but possibly multiple private mempools (e.g., for MEV auctions and/or preventions). This is discussed in more detail later on.

[^2]: Data availability can also be separated out as an independent functionality, but it is natural to consider both availability and ordering as one service offered by the sequencer.

[^3]: Ethereum is availability-favoring, and thus does not offer instant finality on transactions.

[^4]: We do employ committee sampling to improve the optimistic performance of our DA layer, but do not rely on it in the worst-case for security. We use erasure-code based VID to disseminate the data over all nodes.

[^5]: The worst-case communication complexity of any adaptively secure BFT consensus protocol is quadratic. Some protocols (as ours) may have an optimistic path that achieves linear communication complexity and a backup path with quadratic complexity. Other protocols pay this cost up front, which may have advantages such as reducing worst-case latency.