The Preconfirmation Gateway ~ Unlocking Preconfirmations: From User to Preconfer

Much thanks to community for review, specifically George Spasov, Drew Van der Werf, and @g11tech

~tl;dr~

In this post, I examine a few problems in existing Preconfirmation proposals. I introduce a new role: "The Preconfirmation Gateway" to completely abstract preconfirmations from users. I also explore a few examples of how the gateway may be able to facilitate coordination between preconfers.

This post builds on a few select ideas from my article on Transaction Submission on Based Rollups.

First, let's dive into the problem that gateways solve, then we will explore how gateways work and how they solve these problems.

User → Preconfer

On L2s today, transactions move from wallets to centralized sequencers. This flow is facilitated through RPC endpoints. RPC infrastructure is used across many chains and wallets.

With shared sequencing, wallets send transactions to shared sequencer endpoints.

With based preconfirmations this simple flow completely changes. To request preconfirmations, users and wallets will need to do a few complicated things:

Preconfirmation Tip Estimation

Preconfirmations impose limitations on block builders. By restricting a builder's design space, offering preconfirmations for free is negative EV for the proposer (lower block rewards). To compensate a proposer, users need to pay a preconfirmation tip. This tip has some similar properties to an MEV bundle bribe, and it can also be negative.

Estimating preconf tips is incredibly complex because the fair tip depends on the current state of the preconfer. Giving this responsibility to wallets will require complex modifications that would involve estimating the MEV effects of every transaction that a user sends.

Preconfirmation Request Routing

Request routing is the job of deciding which preconfer to send preconfirmation requests to. To do this, wallets will need to check the lookahead to determine who the next preconfer is.

Asking users (and their wallets) to manage a list of preconfers and send Preconf Requests to the correct preconfer may require small modifications to wallets. Importantly, wallets will no longer be able to support a single RPC for all transactions (unless this RPC implements some sort of routing itself.)

Note that this routing complexity may still exist with based sequencing when wallets determine which sequencer to send transactions to.

Preconfer Responsibilities

The "Preconfer" role, as proposed by Justin Drake in Based Preconfirmations has two distinct types of responsibilities:

Preconfirmation Handling

Every preconfer has a core competency: pricing and generating preconfirmation commitments. The specific implementation of a preconfer is a very wide design space.

Preconfirmation Tips

Preconfirmations impose limitations on block builders. By restricting a builder's design space, offering preconfirmations for free is negative EV for the proposer (lower block rewards). To compensate a proposer, users need to pay a preconfirmation tip. This tip has some similar properties to an MEV bundle bribe, and it can also be negative.

Other Responsibilities

Preconfirmation Gossip

Preconfirmation gossiping is simply publicly sharing (or gossiping) a preconfirmation commitment.

There are two reasons presented by Justin Drake why preconfers "should" gossip preconfirmations publicly.

1. Solving the Fair Exchange Problem

The Fair Exchange problem in preconfirmations is that a user paying a preconfirmation tip does not have complete assurance that the preconfer will actually preconfirm their transaction. A user may end up paying for a preconfirmation that they never receive. By gossiping preconfirmation commitments, the preconfer can show users (using a cryptographic signature) that they have preconfirmed other user's transactions. This will avoid the scenario where a preconfer can take tips without providing preconfirmations for more than a few users at once.

This problem is explained in the original Based Preconfirmations post.

There is a fair exchange problem with promise requests and promises. Given a promise request, a preconfer may collect the preconf tip without sharing the promise to the user. A simple mitigation is for users to enforce that promises be made public (e.g. streamed in real time) before making new promise requests. This mitigation solves the fair exchange problem for all but the latest preconf promises.

2. Preconf Transparency

Preconfirmation gossiping also provides transparency to the sequencing process. By gossiping preconfirmations, other entities have a more up-to-date view of the current state of the preconfer. This is important for a few reasons:

- It allows users (or other actors) to price new preconfirmations more accurately
- It avoids spamming the preconfer with multiple preconfirmation requests for the same state
- It provides transparency into the preconfer's behavior regarding preconfirmations (e.g. are they waiting to preconfirm transactions with higher tips, or playing other timing games)

Preconfirmation Delegation

If a preconfer has already preconfirmed a full slot (probably unusual, but possible) they will be unable to preconfirm any more transactions. In this case, the next preconfer in the lookahead may be able to preconfirm the transaction instead. Unfortunately, the next preconfer may not be able to verify that the first preconfer has really preconfirmed a full slot. To avoid being slashed, the next preconfer will need to be sure that the first preconfer has actually preconfirmed a full slot, or else they may be at risk of being slashed in a safety fault.

Delegating preconfirmations has benefits for the user experience (no need to submit a transaction twice) but is also risky for the preconfer who does it.

Chained Preconfirmations

Chained Preconfirmations were suggested by Uri Klarman at the end of Ethereum Sequencing and Preconfirmations Call #1. The idea is that a future preconfer can preconfirm all of the same preconfirmations as previous preconfers. This is a very powerful idea, but it's not clear how it might be implemented. This meta-preconfirmation process could occur ahead of time. Assigning the blame for a safety fault in a chained preconfirmation system is a complex problem.

One very nice property of chained preconfirmations is that users get additional liveness guarantees: if a preconfer goes offline, the next preconfer in the lookahead still provides a preconfirmation!

The Preconf Gateway

The Preconfirmation Gateway is a new role that handles Preconfirmations for the user and manages coordination between preconfers.

The Preconf Gateway does two things: abstracts preconfirmations from users, and coordinates preconfers. The preconf gateway handles all preconfer coordination so that preconfers can focus on providing preconfirmations for users.

Without Gateway:

With Gateway:

The gateway is responsible for the following:

Establishing, Marketing, and Maintaining an RPC endpoint

The original post about Based Preconfirmations introduces the idea that Preconfers will have an endpoint to receive preconf requests from users:

endpoints:

Preconfers can publicly advertise point-to-point API endpoints to receive promise requests and return promises. At the cost of latency, p2p gossip channels can be used instead of point-to-point endpoints.

I propose that Preconf Gateways support a single RPC endpoint that works exactly like a normal RPC (no extra fields for preconfirmations.) This RPC can be integrated into user's wallets who would like to use preconfirmations without any wallet modifications.

Routing Preconf Requests

The gateway will keep track of the active preconfers and their Preconf Request endpoints. The gateway will also keep track of the preconfers that are currently sequencing and responsible for preconfirmations. To do this, the gateway will need to monitor the lookahead and compare it to their list of active preconfers.

Handling Preconf Tips

The other user-facing responsibility of the Preconf Gateway is to handle preconf tips. The gateway will need to estimate the tip for a preconfirmation and add it to the transaction to form a Preconf Request. Depending on the implementation of preconfirmation tips, this may involve a signature to unlock the tip from a smart contract, or an additional transaction to pay the tip. For negative tips, the gateway will be responsible for collecting the tip.

Estimating Preconf Tips

Since Preconf Gateways have a list of active preconfers, they could estimate the tip based on communication with the preconfer. Even this might not be necessary if the gateway has recently interacted with the preconfer while handling other preconfirmations. A Preconf Gateway with a large market share for a specific block might even be able to estimate the proper tip based solely on its view of the market.

Tipping

Similar to a paymaster and regular gas fees, the gateway will sponsor preconf tips for users. Gateways will receive payments from users through other channels, as described below.

Tipping Allowance

A tipping allowance is an on-chain token allowance that gives the preconf gateway permission to transfer a certain amount of a user's tokens.

A tipping allowance will allow users to preapprove a certain amount of tips to be paid for preconfirmations, removing the need for users to think about preconfirmation tips when sending a transaction.

Many transactions (like simple transfers) may not require any preconfirmation tip. For transactions with little to no estimated tip, gateways could subsidize a user's transaction.

Although tipping allowances seem like a pain to set up, there is actually a very easy way to create a preconf gateway tip allowance system if negative tips are implemented.

Tipping Surplus

is the concept of any value that the gateway captures from the preconfirmations that a user requests over their lifetime. Preconf Gateways can keep track of their user's surpluses and use the surplus as a part of the user's tipping allowance, even if the user doesn't do anything. This is a sort of off-chain agreement between the user and the Preconf Gateway that the gateway can optionally support to help their users. Gateways can earn revenue by charging a fee on the surplus that they save the user.

Tipping Surpluses can be created in two ways:

1. The user requests a preconfirmation, but the Preconf Gateway estimates the tip to be negative. The gateway will collect the tip from the preconfer and keep it as a surplus.
2. The user requests a preconfirmation with an inclusion tip (priority fee) that is too high. The gateway realizes that the user has overpaid for their transaction and decides to send a negative preconfirmation tip to the preconfer. The gateway will keep the difference as a surplus.

Case 2 is a little more complex, but it's a very powerful idea as it protects users from overpaying for gas. It also means that users who choose to send a preconfirmation tip through their gas fee will be able to get a refund if they accidentally overpay.

Additionally, users may decide to attempt to pay a tip through their gas fee if they don't have a tipping allowance set up.

Intentionally setting a high priority fee may be a good way to establish a tipping surplus with a preconf gateway.

Gossiping Preconfirmations

Preconf Gateways will also manage the gossiping (or sharing) of preconfirmation commitments. Each preconfer will gossip their preconfirmation commitments to the gateway. The gateway will then create a public endpoint for external parties to monitor the preconfirmations.

Having the Preconf Gateway handle the gossip unlocks a few benefits:

Solving the Fair Exchange Problem (again)

Gossiping preconfirmation commitments does not completely solve the Fair Exchange problem on its own. A preconfer could actually generate fake preconfirmation requests that they preconfirm and then gossip. To an external observer, it would look like the preconfer is providing preconfirmations, but they are actually just preconfirming their own spoofed requests. When a real user (maybe through a gateway) sends preconfirmations to the preconfer, they could pocket the tip and ignore the rest of the request.

[

Spoofed Preconfirmations

640×563 84.8 KB

](https://ethresear.ch/uploads/default/original/2X/a/a8f3c30dbb6bf91351cf49e117e2ab003a24475c.png)

To avoid this, a user needs to be able to verify that the preconfer is actually preconfirming real requests. The gateway can provide this verification by monitoring the preconfer's gossip and verifying that the preconfer is actually preconfirming real requests.

If a preconfer is caught gossiping fake preconfirmations, the gateway can stop sending them preconf requests with tips before users lose funds.

Solving the Fair Exchange Problem (last time I promise)

As previously mentioned, gossiping preconfirmation commitments provides transparency into the preconfer's behavior. This concept of holding preconfers responsible for their behavior can be extended even further. The next fair exchange problem that we can solve is related to timing games that the preconfer can play. By not responding promptly to a user's preconfirmation request, the preconfer could "wait and see" if a higher tip comes in. In this world, preconfirmations essentially become an auction, and we lose the speed benefits of preconfirmations.

To avoid this end state, the gateway can do a few things:

- Monitor the preconfer's gossip and verify that they are preconfirming requests promptly, if not, the gateway can simply stop sending them preconf requests
- Withhold new preconf requests from the preconfer until they provide preconfirmation commitments for the old requests. This will force the preconfer to preconfirm the requests promptly, or else they will lose out on new tips.
- Trust Bonus:

If preconfs trust gateways, the gateway could escrow preconfirmation tips and only release them to the preconfer after they have preconfirmed the transaction.

Transparency + Historical Data

The gateway can also provide historical data about the preconfirmations that the preconfer has committed to (preconfirmed). This is useful for research purposes and may help other actors to understand the performance of specific preconfers.

Delegating Preconfirmations

Preconfirmation Gateways will also be responsible for handling cases where a certain preconfer cannot preconfirm a transaction. The gateway will need to learn that a specific preconfer is no longer able to preconfirm a specific transaction (or all transactions if they have already preconfirmed a full slot.) The gateway will need to find the next preconfer in the lookahead and send the preconf request to them. The gateway may also need to adjust its tip estimation based on the new preconfer.

As mentioned previously, preconf gateways will have a lot of information about the state of the preconfer, so they may be able to begin delegating preconfirmations that they know the current preconfer will not be able to preconfirm.

Chained Preconfirmations

The Preconfirmation Gateway will also handle Chained Preconfirmations. It is still an open question how the slashing around Chained Preconfirmations would work, but it may be possible for the gateway to handle this as well. If

all preconfers trust the gateway (not necessary for the other responsibilities) then the gateway could handle the slashing for the preconfers.

The Preconf Gateway in practice

& FAQ

I propose that the mev-boost relays will serve as preconf gateways initially.

Relays are already trusted by proposers, which solves the quick start problem if we want this.

Why “Gateway”?

Although I have proposed that the gateway will initially be the same as the relays, I’m intentionally not calling it a relay because I don’t want to keep the association strong (mev-boost relay = preconf relay).

What are the differences to a mev-boost relay?

The key difference between a relay and a gateway is the trust involved. A relay is trusted by proposers to include their transactions in a block. Proposers (and preconfers) actually don’t need to trust a gateway at all. In fact, preconfers don’t even need to know that gateways exist. Depending on the specific coordination mechanisms between preconfers that gateways support, they may have similar trust requirements to mev-boost relays.

What is the Preconfirmation Gateway Endgame?

My personal vision is that the top relays start running gateways and users select their favorite (based on how they distribute tip surplus, etc.)

Over time, new specialized entities may emerge and gain market share.

Eventually, users will have many options for gateways, and a competitive market for a gateway’s services will emerge. The competition will drive fees lower and improve the UX for users

So what is actually being sent around?

Some of these diagrams and explanations assume certain implementation details around preconfs. Although there hasn’t been group consensus around all of these, the concept of “a Gateway abstracting away preconfirmations” works with many possible implementations.

In the model presented above, users send signed transactions to Gateways. Gateways send preconfirmation requests (signed transaction + tip + metadata) to preconfers. Users may also send preconfirmation requests to preconfers.

Inclusion vs Execution Preconfs

How does the user specify their preference for an inclusion vs an execution Preconfirmation? I believe that gateways can also determine (from examining a user’s transaction) which type of preconfirmation to request from the preconfer. If a user would like to specify this, there are other ways to (e.g. a registration system or URL parameters) support this.

This same idea can actually be applied to any additional (optional) features that gateways could implement.

References

I used many resources while writing this article:

https://mteam.space/eth-sequencing-preconfs-resources