The Based Stack by Spire Labs

interoperable · censorship resistant · fast · lightweight

Spire is the first **based rollup framework** & network on Ethereum, enabling a wide range of app developers to create their own appchains on Ethereum. Spire's sequencing design enables appchains to interact with Ethereum's liquidity and protocols natively (think: cross-chain swaps). Spire appchains support a wide range of execution environments, scalability upgrades, and decentralized preconfirmations for lightning-fast UX.

The Based Stack is a culmination of *years* of research from many sources. Especially inspirational (and helpful as background) were the following:

- Based rollups by Justin Drake
- Based preconfirmations by Justin Drake
- The Ethereum sequencing & preconfs calls
- *and many, many more*

Personal Comments

While working with Ethereum researchers we have been consistently amazed at the gracious and kind culture that permeates every interaction that we've had so far. Other teams (perhaps even future competitors) have gone out of their way to just be helpful. While research writeups and discussions have been incredibly helpful, we have found that the based ecosystem is more than willing to help in other ways as well: intros, advice, and more. As Spire Labs continues to define our plans for the next few years, we would like to continue to be supportive of good-faith teams and the Ethereum ecosystem. This level of support is, as of today, only present in the Ethereum ecosystem - and we firmly believe it is Ethereum's most valuable "feature."

To everyone who helped Spire Labs get here today: we express a heartfelt gratitude for all of your contributions and truly wish you the best. We owe a lot of favors.

Table of Contents

Problem Statement - fragmentation at its finest

“Appchains lose composability with each other and Ethereum L1.”
We’ve talked to many OG DeFi apps on Ethereum, and we keep hearing the same sentiment: “We want to launch an appchain but can’t sacrifice composability with Ethereum.”
There are many types of fragmentation that exist on Ethereum today (liquidity, mindshare, users, etc.). Our goal is to reunite every one of these into a single seamless experience.

The Based Stack makes appchains practical by introducing a variety of features that, when combined in a seamless developer experience (abbr. DX), are the obvious choice for app developers.

Design Ideals

Our design ideals are based on many conversations with apps and users. **We especially prioritize ideals that are pragmatic for apps and users instead of trying to be “Ethereum Aligned.”** With that being said, we have found that apps that exist today on Ethereum do not want to sacrifice the “features” and ideals that the Ethereum L1 offers - **especially censorship resistance, permissionlessness, liveness, and decentralization.**

Censorship Resistant

While Ethereum itself is having a bit of a censorship crisis, there is light in the dark forest and significant research into this ideal. For some apps, censorship resistance (CR) can be crucial, but for others, it is simply a “nice to have.” **We believe that robust CR is important for any distributed consensus system, especially blockchain applications.**

Efficient

At Spire, we are strong believers in the “Appchain Thesis” but recognize that **an effective appchain framework must support applications with low volume and few users.** Adoption of appchain frameworks like the Cosmos SDK has demonstrated that any bootstrapping required to launch an appchain is a huge barrier to entry. At Spire, we want to support the smallest viable application without massive tradeoffs.

Open

Open source, open innovation, and transparency. These are all characteristics that are fundamental to our company philosophy and key to all of our workflows.

Seamless

Perhaps the most important on this list, we believe **a seamless experience is key to adoption and success.** In practice, this looks like a consistently polished UX and DX. Read on to learn how this ideal permeates every aspect of our design.

Based Rollups - from the ground up

At Spire, we have spent a long time thinking about the "**based thesis**." In this section, we'll build from first principles to argue for the adoption of based sequencing.

Based sequencing is an answer to the question: **who should sequence my chain?**
Many options exist, but at the end of the day, a sequence of transactions must be agreed upon.

- **Centralized Sequencers:** A single designated entity has a monopoly over sequencing. Anyone can force-include a transaction by posting it directly to L1, albeit after a delay.
- **Decentralized Sequencer Election:** What is colloquially referred to as "decentralized sequencing" is often just a mechanism for choosing, or *electing,* a centralized sequencer for a (usually) short period of time. This sequencer can be constrained in multiple different ways. *This is the design that Ethereum uses.*
- **Shared Sequencer Election:** Colloquially referred to as "shared sequencing," this mechanism involves coordinating with other sequencing domains (chains) so that the same centralized sequencer is elected for each domain. Again, this sequencer can be constrained in multiple ways.

Centralized sequencing is nice. It enables a single node to scale an entire (sufficiently) decentralized rollup. Projects like Eclipse and MegaETH are taking advantage of a superpower that centralized sequencing can offer. With that being said, **centralized sequencing is fundamentally at odds with a decentralized (fragmented) ecosystem of applications.**

The "solution," shared sequencing, is being pioneered by the excellent teams at Espresso, NodeKit, and more. **Existing designs support some impressive features** on paper, like atomic inclusion on low-latency block times, BFT preconfirmations, and fast finality. When paired with proof aggregation and shared deposits (à la AggLayer) these options seem very attractive. One potential hurdle to widespread adoption is simply **social coordination.**

Based Sequencer Election (based sequencing) is a subset of Shared Sequencer Election
Based sequencing, in practice, involves a specific subset of shared sequencer elections where one of the domains being sequenced *is* Ethereum. Because Ethereum's sequencer (proposer) election is done with a stake-weighted pseudo-random lottery facilitated by the Ethereum network, we can't change the sequencer election mechanism to another mechanism without a major Ethereum upgrade. Therefore, **based sequencing designs attempt to reuse Ethereum's sequencer election mechanism as their own sequencer election mechanism.**

Now that we understand that it is possible to use Ethereum as a sequencer, we return to the original question: **who should sequence my chain?** Even if Ethereum is an option, why is it any more appealing than other sequencer election mechanisms?

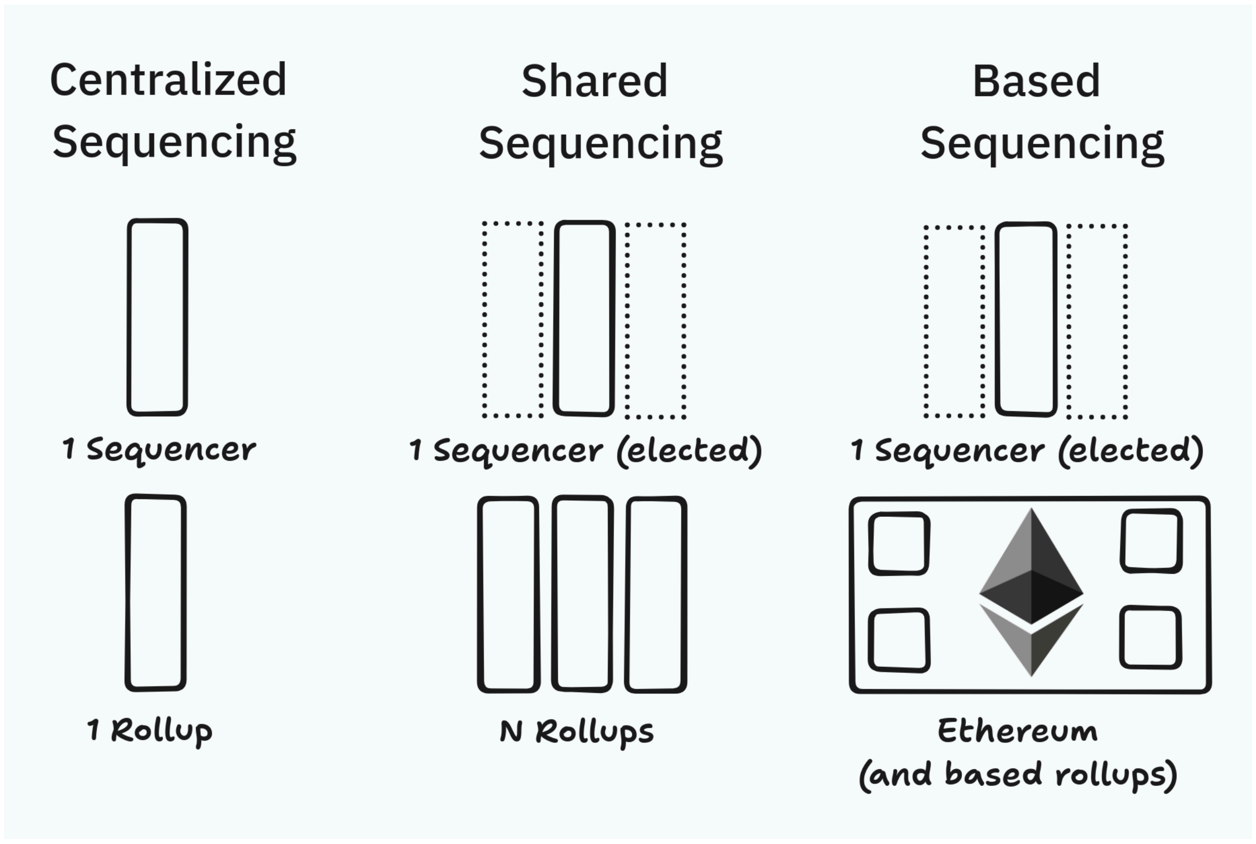
First of all: **Ethereum's sequencing is not perfect today.** MEV-Boost has always come with tradeoffs. The "diverse" block-building market has consolidated into a small group of competitors who receive private orderflow and extract MEV themselves with very few constraints. MEV-Boost is not meaningfully decentralized. Additionally Ethereum block times are constrained by the large validator set, sitting at approximately 12 seconds. With this being said, the Ethereum block building market is still relatively competitive and efficient.

MEV-Boost today

- Validators don't need to participate in "the MEV game."
- Client agnostic, widely used, and is valuable for informing future PBS research.
- Centralized, yet competitive block building market.
- Censorship resistance is not enforced *yet*
- Slow (12s per block) but MEV-Boost is not the throughput limiter here.

But a sequencer is not only defined by *how* it sequences - but also by *what* it sequences. The "Ethereum sequencer" is most valuable in this regard (*especially* for Ethereum-native applications) because **Ethereum is the most valuable blockspace.** Ethereum's dominant TVL, institutional/developer mindshare, and vibrant research community all make it especially inviting to compose with. Additionally, the "Ethereum" brand is a schelling point for rollups: Ethereum's rollup-centric roadmap is almost custom fit to this vision. While we don't believe based rollups are "built to support Ethereum" any more than a traditional rollup, **the coordination that a neutral and Lindy layer like Ethereum offers can bring value to many applications and existing L2s.**

Finally, the noted issues with MEV-Boost (PBS) are all expected to be resolved by constraining block builders (see Inclusion Lists) and other Ethereum upgrades (check out commit-boost too!). At Spire **we fully expect that Ethereum becomes an even better sequencer over time.**



The Based Stack

“The Based Stack makes appchains practical by introducing a variety of features that, when combined in a seamless developer experience (abbr. DX), are the obvious choice for app developers.”

Spire’s **core product offering** is the “Based Stack" by Spire Labs. The Based Stack is **a rollup framework** (analogous to the OP Stack, Polygon CDK, and zkSync’s ZK chains) that includes native support for based sequencing and other features. The Based Stack is specially designed to support “**based appchains**” with single-app rollups instead of general-purpose rollups. In this section, we introduce a few aspects of the Based Stack:

Based Sequencing

As the name implies, the Based Stack includes a **based sequencer election mechanism** developed by Spire.

Based sequencer election is used to determine the "based sequencer" of an appchain. The elected based sequencer has the exclusive ability to sequence and propose appchain blocks, and therefore to give preconfirmations and other proposer commitments. **The sequencer**

election mechanism used by appchains may differ, appchains may choose to optimize for MEV capture, composability, decentralization, etc. Some sequencer election mechanisms are more valuable when other appchains are using the same one, building a **network effect.**

Before we start, some quick terminology:

- **Based Sequencer:** The entity that has permission to propose an appchain block on L1. Is the winner of the based sequencer election mechanism.
- **Total Anarchy Based Sequencing:** The design that Taiko uses. A block proposal on L1 is made completely permissionless. While the L1 proposer is elected ahead of time, a specific based sequencer (L2 block proposer) is *not* elected ahead of time.
- **Vanilla Based Sequencing:** Originally introduced by Limechain, Vanilla Based Sequencing describes a simple sequencer election mechanism: elect the current L1 proposers if they have opted-in and if not, run a random lottery to choose another L1 proposer that has. Note that opting-in may require some amount of preconfirmation collateral.

The sequencer election design that the Based Stack supports is maximally configurable to support the requirements that different applications have. Here are a few tradeoff spaces we have identified:

1. **Sequencer Revenue (MEV) Retention vs Sequencing with L1:** To retain sequencer revenue with an auction, there must be more than one participant in that auction. There is only one active Ethereum proposer at a time, so tradeoffs must be made.
2. **Preconfirmation Availability vs Sequencing with L1:** A similar tradeoff to the above: to get sequencing with the L1, the based sequencer must be the active Ethereum proposer. If this proposer has not opted in to provide preconfs, preconfs will not be available during this proposer's slot. Using a sequencer-includer design, we can elect based sequencers that are present in the Ethereum lookahead that have preconfirmation collateral and avoid this issue.
3. **Gas Efficiency vs Interop with L1:** Whenever an L2 block is proposed, L1 gas costs must be paid. As Ethereum gas is expensive, we want to minimize the amount of gas used in block proposals and also minimize the quantity of blocks proposed. In practice, this might mean setting the block time of an appchain to X*12 seconds where X>1. These longer block times can be abstracted from users with preconf designs like multi-round MEV-boost.

Construction

We will divide this section into two primary elements of the Based Stack's sequencing construction: based sequencer election and sequence constraints. The based sequencer election component outputs an elected entity with permission to propose an appchain block, while the sequence constraints component imposes constraints on the contents of the proposed block.

based sequencer election

1. An L1 election contract periodically runs a dutch auction to distribute tickets (implemented as NFTs). This market is permissionless and anyone can participate. The number of tickets distributed must equal the period of the auction, so if the auction begins every 10 blocks, 10 tickets must be distributed to the auction winner. Some genesis tickets may be minted. Additional details about the auction process (such as complex bids for multiple appchains) will be explored further in future content.
2. As soon as the Ethereum lookahead for the current epoch is available, based sequencers can be deterministically calculated for the entire epoch through the following (configurable) rules:
    a. First, proposers in the lookahead are determined for eligibility if they control at least one ticket and follow any and all additional configurable constraints that the appchain chooses (such as some minimum preconfirmation collateral posted). Proposers that meet these criteria (referred to as *eligible proposers*) will always be elected as the based sequencer unless no proposers meet this criteria.
    b. Once the subset of eligible proposers has been identified, sequencer election takes place starting at the end of the epoch, with the 32nd slot. The following algorithm is used to determine the elected sequencer:
    c. If the proposer being tested is an eligible proposer, they are elected as the based sequencer. One ticket that they hold is burned.
    d. If the proposer being tested is not an eligible proposer, the last eligible proposer tested (so the next eligible proposer in the lookahead) is elected if they still have a nonzero ticket balance. One ticket that this elected sequencer holds is burned. If no eligible proposer has been tested yet, a random eligible proposer is elected. If the last tested eligible proposer does not have a nonzero ticket balance (has 0 tickets), a random eligible proposer is elected. If no eligible proposers exist, a random ticket holder is elected. If no ticket holders exist, block proposal is permissionless. These fallback rules can be configured.

sequence constraints

We introduce two special sequence constraints:

1. **Forward Based Sequencing:** An additional set of actors, called “forward sequencers” are chosen. Forward sequencers have the special ability to submit “forward sets” to the L1 rollup contract. The forward sequencers can be chosen in multiple ways: a whitelist, token-gated, for a fee, or completely permissionless. An appchain can choose how this election works and may even choose to elect no forward sequencers. Forward sets are comparable to Inclusion Lists, and they constrain the next based sequencer - all transactions in the forward set must be included in the block that the based sequencer proposes. This design has strong similarities to various proposed MCP gadgets such as FOCIL.

2. **CR Committee:** An additional set of actors, called "CR Committee Members" are also given the ability to influence block production: every block proposed must include signed data from at least *K* CR Committee Members, where *K* is a certain number of Committee Members. The message that is signed must be (a hash of) a list of transaction hashes, where each transaction hash present must also be included in the block. This design is somewhat analogous to the "Censorship-Resistance Committee" introduced by Quintus, although implementation details differ. Note that while the mechanism to elect Committee Members is easily configurable by the appchain, we believe whitelisting makes the most sense (as at least *K* CR Committee Members must be live to produce a block). As with forward based sequencing, an appchain can choose to elect 0 CR Committee Members (and set *K=0*) to opt out of this feature.

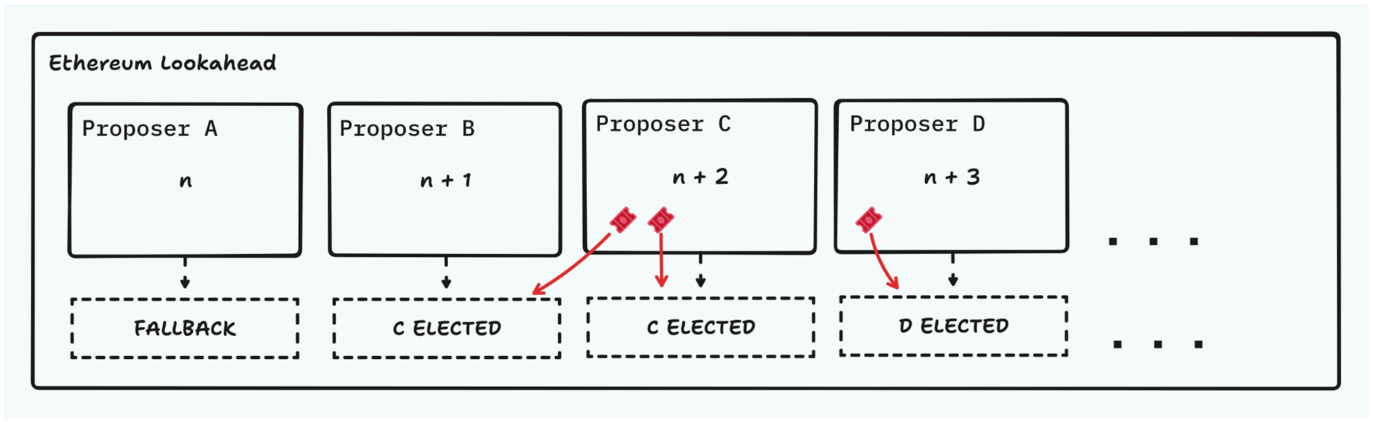
Explanation

Ticket distribution is done with a descending dutch auction for two main reasons: it is a relatively **well-understood simple auction mechanism and it is friendly to combinatorial auction designs.** Also important is that manipulation from *weak censorship* on Ethereum has minimal effect.

The sequencer election mechanism described is designed to prioritize ticket holders with multiple tickets and sufficient preconfirmation collateral - but **these can both be parametrized.** We recognize that some applications may need to prioritize different sequencer election properties (maybe liveness over censorship resistance) and believe this design can be configured to this end.

The final two components listed above are included to support and inherit the censorship resistance of Ethereum. A CR Committee, if implemented correctly, may even lead to an appchain with *better censorship resistance than Ethereum itself.*

These design decisions are also made to support lightweight applications: appchains that may not have many users or frequent transactions. We continue to research how certain tradeoffs can be made to enable **the true tail end of applications.**



Customizable Composability

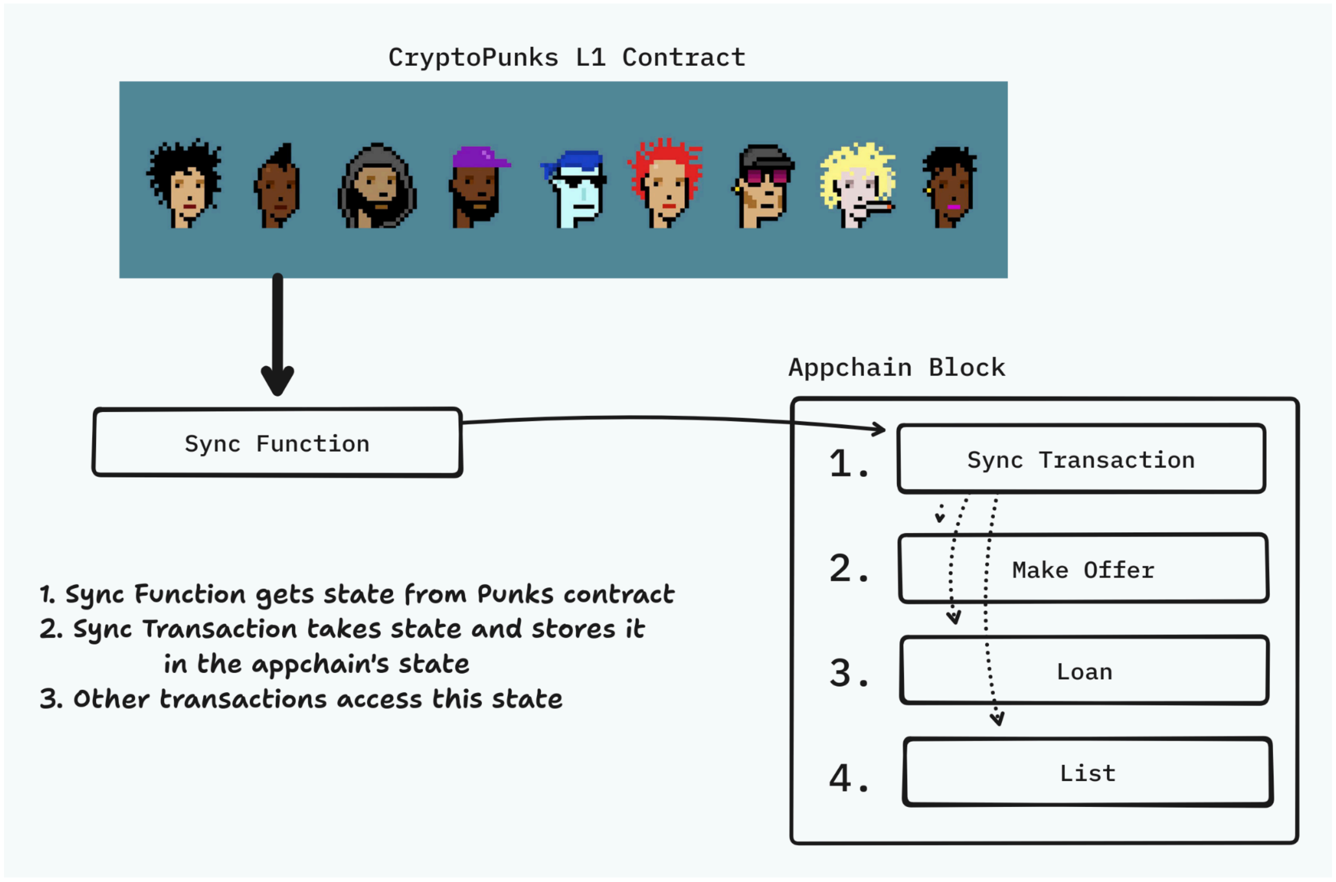
Customizable composability is an important concept at Spire Labs, and the idea is pretty simple. We use the fact that based rollups created with the Based Stack are appchains to **customize the rollup contract in ways that are impossible on general-purpose rollups.**

We introduce the concept of a “sync function” which is an arbitrary function that can be executed on the L1 to produce arbitrary data as output. Then, we enforce (at runtime) that the inputs to the first transaction (the “sync transaction”) in any appchain block proposal is the output of executing the sync function.

This is interesting because it allows applications to specify read access to Ethereum every time an L2 block is executed. **The L2’s execution environment can always have effectively real-time read access to Ethereum.** This includes any state: **L1 oracles,** Ethereum history, and more. Note that this does not require any modifications to the VM used.

For example, imagine an NFT marketplace that would like to allow its users to trade CryptoPunks. Of course, CryptoPunks only exist on Ethereum L1. Implementing this application as an appchain might involve using customizable composability to read the entire state of the CryptoPunks contract into a contract on the appchain. Then, **every transaction in the appchain block will have access to the up-to-date state of the CryptoPunks contract.**

Functionally, this is a kind of new way to be composed with L1 and is an example of **the unique innovation that can be done when implementing appchains vs general purpose rollups.**


CryptoPunks L1 Contract
Appchain Block
Sync Function
1.
Sync Transaction
2.
Make Offer
3.
Loan
4.
List
1. Sync Function gets state from Punks contract
2. Sync Transaction takes state and stores it in the appchain's state
3. Other transactions access this state

Preconfirmations

At Spire, we believe that preconfirmations (abbr. preconfs) are a helpful technology for improving user experience on based rollups. Spire's integration of preconfs is agnostic to preconfirmation protocols. **We are partnering and working with some of the major preconfirmation protocols in an attempt to unify them permissionlessly and encourage the market to decide which is the best option.** Below, you'll read more about The Preconfirmation Registry and The Preconfirmation Gateway, which are two ways we're more involved in preconf research and implementation.

The Based Stack does not include specific preconfirmation support (i.e. support for slashing, collateral, or delegation.) The important exception is Spire's sequencer election, which **intentionally only chooses proposers that have some predetermined minimum amount of preconfirmation collateral.** This is important because it indicates that users will be able to access preconfs more often, even if not all Ethereum validators are opted-in to preconfs.

We also attempt to make the Based Stack helpful and friendly to preconfs in other ways (like making L2 blocks easy to access from the L1 execution environment for things like preconf penalties).

Why preconfirmations? Aside from extremely fast "soft confirmations" for users, some designs for preconfs (execution preconfs to be specific) also enable **toxic MEV protection,** an important part of improving user experience when interacting with applications.

Spire is also actively contributing to preconf research as we believe that getting this right is important to the effective functioning of based rollups in the near future. Ultimately, we believe that faster block times on Ethereum make based rollups (aka "fast based rollups") even more powerful. To this end, we fully support Ethereum research being done to reduce the block time from 12 seconds down to 2-4 seconds. We also understand that such a change requires significant changes to Ethereum's consensus and therefore would not like to be dependent on such an upgrade soon.

Execution Environment / Virtual Machine

One of the important primitives in the Based Stack is the separation of sequencing and execution.
Sequencing: coming to consensus on the inputs to a State Transition Function (STF)
Execution: coming to consensus on the output of an STF.

At Spire, we group concepts like validity proofs (aka ZK proofs) and fraud proofs, but also virtual machine (VM), which is what we will focus on in this section. We believe that **developers should be able to choose the virtual machine that works best for their application.** The Based Stack is designed to eventually support multiple VMs. We are especially interested in **combining the throughput and scalability of alternate virtual machines like the MoveVM and the SVM to scale Ethereum applications** with innovative features like localized fee markets, parallelization, and highly optimized node software.

At launch, the Based Stack will include support for the EVM with OP Stack fraud proofs as we believe that the EVM is, in practice, the most useful for many types of applications. **EVM support is especially important for applications that have already deployed on Ethereum and therefore, have written their code bases for the EVM.**

We are also interested in understanding how **next-generation virtual machines and execution environments** like Cartesi, Lumio, RISE, and Fluent can play a role in the Based Stack. These are all paths and partnerships that we are exploring or starting to explore.

Sequencing Revenue (MEV) Retention

Today, applications leak millions of dollars to their sequencers.

Spire introduces a sequencer election model (described in more detail above) that prioritizes sequencers that are willing to pay to sequence a block, thereby returning some value to the application.

We expect that the total amount of sequencing value returned to the application through this mechanism will be on the order of the expected value of sequencing that appchain. **In practice, this could be millions of dollars for some types of applications** and may end up being very little for others. Because of this disparity, MEV retention is something we want to be flexible with and allow applications to parameterize. Notably, **smart contract level solutions (like Atlas) are also effective ways to retain MEV and can be used in conjunction with sequencer election models.**

Priority fees and other gas fees that users pay to be included in a block are detectable within the execution environment. This means that implementation here can be done in the execution environment, not on the sequencing layer. These types of fees can be trivially parametrized to be more flexible for different apps. We are continuing to do research into the numbers that make the most sense for most applications.

Side note: In our conversations with applications, we have noted that this is one of the least important features of the Based Stack, especially when it is at odds with composability with L1. ***We are working on complete data analysis for applications that exist on L1 to discover exactly how much revenue they are leaking to the Ethereum sequencing mechanism*** *that they could be retaining with something like Spire's sequencer election mechanism.*

Traditional Shared Sequencing

We recognize that significant effort has been made on traditional shared sequencers (shared sequencers that are not "based"). One valuable feature of the Based Stack is that appchains can still reap the benefits of shared sequencing with other L2s. As mentioned above, we implement a design called ***Forward Based Sequencing***, which gives shared sequencers (acting as forward sequencers) the ability to influence block production without a distinct monopoly. **This enables shared sequencers to provide composability between Spire based appchains and other non-based rollups,** especially when paired with execution preconfirmations. Execution preconfirmations allow forward sequencers to get (collateral-backed) execution guarantees that can be used to approximate atomic execution. We call this "optimistic atomic execution."

One important thing to note is that this design could theoretically be used in any rollup with an L1 escape hatch with a low wait period. In the Based Stack, **we take this primitive to the next level by introducing forward sequencers as "first-class citizens"** in block production by focusing on gas efficiency, practical technical integrations with shared sequencers, and docs for integration.

Open Source and Credibly Neutral

The Based Stack is a **public good and will be developed as an open-source project.** We plan to be transparent throughout the entire implementation process and to promote open source contributions from the community. This is an important part of our brand alignment, but also the way we believe that blockchain projects *should* be developed.

Credible neutrality, in this context, means that the Based Stack does not depend on a token to function, rely on a specific service provider, or even extract revenue from the sequencing process.

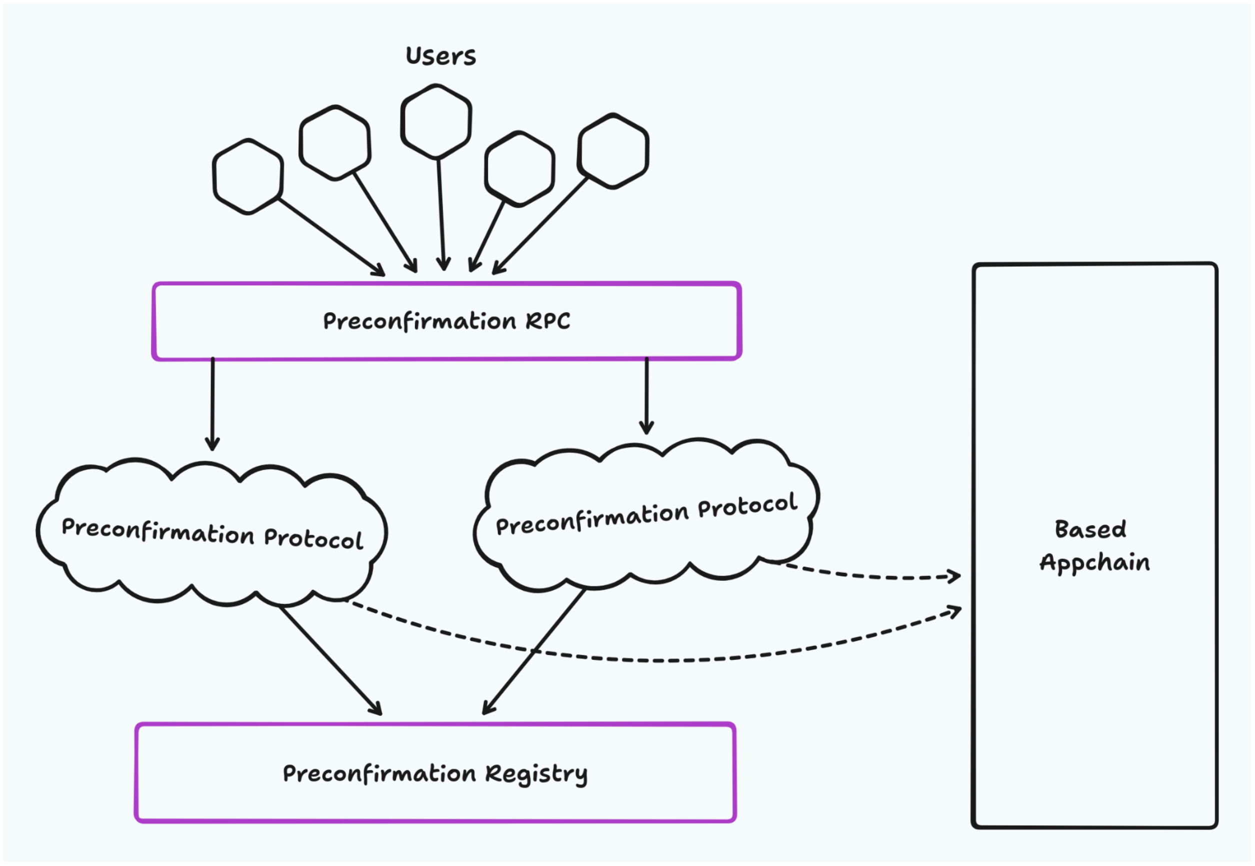
Future Upgrades

After the Based Stack launches and is ready for production use, the Spire Labs team will continue improving the rollup framework. Some specific things we are thinking about:

- **ZK Proof Aggregation / Shared Deposits:** Enabling cheaper bridging between participating appchains. We are interested in collaborating with other teams on this.
- **Blob Aggregation:** Building tools to enable efficient sharing of blob space on Ethereum to minimize DA costs for lightweight applications.
- **Diverse Execution Environment / VM Support:** Integrating and developing important features to support alternative VMs and next-gen execution environments natively.
- **Realtime Proving:** Enabling synchronous composability with Ethereum. We are looking into many designs related to this, including fast ZK proving and also TEE proving.

Auxiliary Products

While Spire focuses on the implementation and design of the Based Stack, **we believe that both of the following products are key components of the preconfirmation ecosystem.** Both of these products also introduce **persistent revenue streams** for Spire Labs. *Canonical links to learn more about these projects have been included.*



The Preconfirmation Registry

The Preconfirmation Registry is a **simple, credibly neutral, open source, and public good smart contract** deployed on Ethereum that facilitates preconfirmation collateral, slashing, registration, and delegation. The Registry is permissionless and agnostic to preconfirmation protocol, enabling the Registry's role as **a key schelling point** for present and future preconfirmation protocols.

Spire intends to work with validators (Node Operators and solo stakers) to help when interacting with the Preconfirmation Registry. In some cases, **Spire may charge a consulting fee for these services**, although it is important to recognize that Spire's assistance is not required to use the Registry.

The Preconfirmation RPC (fka Gateway)

(Initially published on ethresear.ch as "The Preconfirmation Gateway", the Preconf RPC has since been rebranded to reflect updated terminology standards in modern research.)

The Preconf RPC is an important piece of the complete Spire appchain vision as it enables the abstraction of preconfirmation complexity from users. **Users that opt-in to the Preconf RPC can**

access the benefits of fast preconfirmations without switching wallets, as the Preconf RPC implements (and extends) the standard Ethereum JSON-RPC API.

The Preconf RPC is a highly sophisticated server, run by Spire Labs, that converts signed transaction requests into complete preconfirmation requests - and routes them to the correct preconfer. One of the primary responsibilities of the Preconf RPC is to price preconfirmations on behalf of users, an extremely sophisticated task that requires a broad market understanding.

The Preconf RPC run by Spire **will be open source,** and we fully expect competition to emerge (RPCs are completely permissionless) that drives costs for users towards zero.

Launch - going to market

Seamless DX

At Spire, we believe it is **crucial for any rollup stack adoption to have a seamless developer experience.** In practice, this means good documentation, good code examples, and real-time support. One thing Spire is doing differently than other teams is building out **a dedicated migration team** that is focused on helping apps move to Spire appchains from L1 and other sources.

One example of this DX is the introduction of **in-place appchain deployment**. For mutable L1 contracts with high amounts of TVL, the Spire migration team will assist app developers in deploying a Spire appchain as an upgrade of these contracts. This enables a seamless onboarding experience for developers, but also **a seamless experience for users.**

Because the Based Stack is open source, we expect Rollup as a Service (RaaS) providers to integrate the Based Stack as an option. Additionally, **we are working with a few smaller, innovative RaaS providers to create a unified experience for appchain developers.** More details will be announced in the coming weeks.

Integrations

The full appchain experience includes many features that Spire will not provide on its own. We will be announcing partnerships and integrations with **preconfirmation protocols, bridges, blockchain explorers, indexers, wallets, portfolio trackers, AA toolkits, and more** in the coming months. We go deeper than just a surface level integration and focus on **native integrations** at the deepest level of the stack. We believe that it is especially important to stay credibly neutral throughout these many technical integrations while also being very particular about prioritizing the best app experience for our users.

Target Applications

We believe that many different types of applications will eventually use Spire appchains. There are a few special classes of apps that we believe are **particularly suited to the positive features of the Based Stack.** In practice, these are the applications that we will target in BD and marketing efforts and primarily support through our migration team.
The targeted classes include:

- **Existing L1 Applications**
- **Existing Appchains**
- **New Promising Lightweight Applications**

We focus on applications with PMF and existing user bases or extremely promising new applications from teams with solid track records. This is to encourage the launch of **"killer applications,"** which have historically been the **primary drivers of value** for new ecosystems (Friend Tech on Base, Pump Fun on Solana, Uniswap on Ethereum). We avoid a focus on clones or "copy-pasta" apps in any form, as these apps and teams generally exhibit mercenary behavior.

Launch Strategy

At launch, we will partner with a few applications to launch **flagship appchains** that demonstrate the features of the Based Stack in production. Spire Labs may also launch a few in-house appchains to demonstrate features of the Based Stack at a more granular level. We strongly believe that **using our own product** will lead to faster feedback cycles and a more polished final product.

While we have no present plans to incentivize appchain deployment with token grants, we are actively thinking about how we can support Based Stack appchains with marketing and BD.

Value Accrual for Spire

Value Accrual Philosophy

The primary driver of all of our decisions around value accrual is maintaining the credible neutrality of the Based Stack. From a business perspective, we believe that competition will drive infra fees down, and don't want to participate in a race to zero by trying to sell a commodity for a premium towards zero.

However, we do want to prioritize value accrual and revenue generation where the **network effects that Spire has coordinated are valuable to appchains.** Those are places where we're willing to generate revenue, but we aim to be as credibly neutral as possible on every other layer of the stack to make appchains even more appealing.

Wen Token?

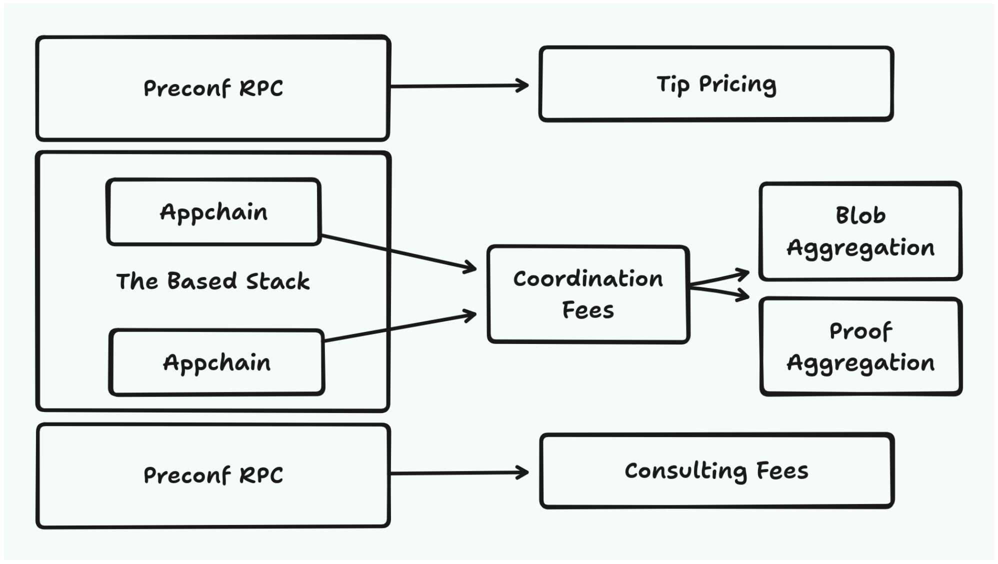
As mentioned before, the Based Stack **does not depend on a "Spire token"** and is fundamentally a tokenless protocol. We recognize that there are many reasons to launch a token, but we don't want to force that upon our users. One big reason is that we are building application-specific chains and we believe that **apps will have tokens that they would like value to accrue to.**

So: wen token? **Not at mainnet.** We may launch a token if the time is right and we have clear utility for said token. We commit to taking the protocol guild pledge in the event that we do launch a token.

Revenue Streams

There are a few different revenue streams for Spire. We like to categorize these into 3 areas:

1. **Coordination Fees:** This includes **post-mainnet protocol upgrades** like ZK proof aggregation, blob aggregation, and more. This also includes small fees on **additional auction efficiency** from the coordination of combinatorial auctions.
2. **Preconfirmation Tips:** Because the Preconf RPC has the ability to **price preconfs** on behalf of users, many fee generation models can be implemented.
3. **Preconfirmation Collateral:** Registering to become a preconfer is an important task that many Node Operators and validators on Ethereum have shown interest in completing. Spire Labs, as experts in this area, may charge **consulting fees** for completing such tasks on behalf of validators.



Concluding Thoughts

This litepaper is only the beginning. We will be announcing partnerships with a variety of infra partners in the coming months.

If you are interested in learning more about technical details we didn't cover above, we encourage you to reach out: hello@spire.dev

For anyone curious about following Spire's progress, we will be using the @Spire_Labs Twitter account for communications.