Micro-Rollups: Modular Micro-Services for Web3

Kautuk Kundan
kautuk@stackrlabs.xyz

April 4, 2024

Abstract

Micro-rollups represent a transformative approach in blockchain application development, aiming to bridge the gap between the traditional web2 developer experience and the new web3 paradigm. This paper introduces the concept of micro-rollups as a novel framework designed to enhance scalability, flexibility, and developer accessibility in building decentralized applications. By drawing parallels with the micro-services architecture prevalent in web2, micro-rollups enable developers to construct decentralized applications as a collection of independent, modular components. This approach not only simplifies the development process by allowing the use of familiar programming languages and tools but also significantly improves application performance and user experience.

Furthermore, micro-rollups address key challenges in the current blockchain ecosystem, such as state management, transaction ordering, and interoperability, by providing a customizable execution environment that supports efficient state transitions and asynchronous communication between decentralized applications. This paper delves into the architecture, features, and potential use cases of micro-rollups, illustrating their role in fostering innovation and accelerating the adoption of web3 technologies.

1 Introduction

1.1 Problems with traditional rollup designs

Ethereum stands as a cornerstone of blockchain activity, driving innovation and development within the space. However, as it grapples with the blockchain trilemma, scalability challenges have become increasingly apparent. To address these issues and enhance Ethereum's capacity, Layer 2 (L2) solutions have emerged as pivotal innovations. These solutions aim to alleviate the strain on Ethereum's network by handling transactions off the main chain, thereby reducing congestion and fees while maintaining security and decentralization.

Within the landscape of L2 solutions, both general-purpose and application-specific rollups have played crucial roles. General-purpose rollups offer a broad platform for various applications, extending Ethereum's functionality without compromising its foundational security. On the other hand, application-specific rollups provide tailored environments that cater to the unique requirements of individual applications. However both constructions have their own shortcomings.

1.2 General Purpose rollups

While general-purpose rollups are advantageous for a lot of scenarios, they encounter limitations when it comes to highly specialized or performance-intensive applications. The broad scope of their design can lead to unnecessary bloat and inefficiencies for use cases that demand a lean and highly optimized execution environment.

Additionally, the challenge of interoperability remains significant, as ensuring seamless communication and functionality across different rollups and the wider blockchain ecosystem can be complex. These issues underscore the importance of developing more nuanced solutions that can cater to the specific needs of diverse applications within the blockchain space.

1.3 Application Specific rollups

App-specific rollup environments emerged as a response to the limitations of one-size-fits-all general-purpose rollups, aiming to tailor the blockchain experience to the unique needs of individual applications. By modifying general-purpose virtual machines (VMs) to suit specific use cases, these app-specific rollups sought to optimize performance, reduce costs, and alleviate congestion. However, this approach, while well-intentioned, falls short of being the optimal solution for several reasons.

1. **Complexity of VM Customization**

   The process of customizing general-purpose VMs to create app-specific environments is inherently complex and resource-intensive. It requires significant effort to strip down and reconfigure a VM to fit a particular application's requirements, which can be a barrier to entry for many developers. This complexity not only slows down the development process but also limits the flexibility to rapidly adapt to changing needs or incorporate new functionalities.

2. **Fragmentation and Interoperability Challenges**

   App-specific rollups, by their very nature, create silos within the ecosystem. While they may offer optimized performance for specific applications, they do so at the cost of interoperability. Each app-specific rollup becomes an isolated environment, making it challenging for applications to interact with each other and with the broader ecosystem. This fragmentation can hinder the composability that is a hallmark of the decentralized web, where applications are envisioned to seamlessly integrate and build upon each other's functionalities.

3. **Innovation Limitations**

   The reliance on modified general-purpose languages does not fully escape the constraints and inefficiencies inherent to those systems. While customization can lead to improvements in certain areas, it often carries over the limitations and overhead of the underlying technology. As a result, app-specific rollups may still face issues related to scalability, flexibility, and developer accessibility, which they were meant to overcome.

In summary, while app-specific rollup environments represent a step towards addressing the one-size-fits-all dilemma of general-purpose rollups, they are not without their own set of challenges. The complexity of customization, ecosystem fragmentation, and inherited limitations of modified general-purpose languages suggest that a more fundamental rethinking of rollup technology is necessary to unlock the full potential of scalable, interoperable, and developer-friendly blockchain applications.

2 Micro-Rollups

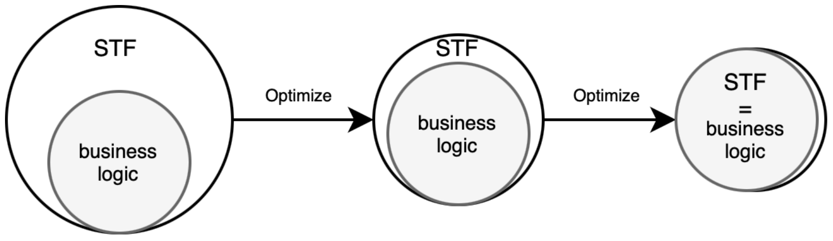

Figure 1: App-specific Rollup to micro-rollup

Micro-rollups mark a significant advancement in blockchain scalability through a specialized framework designed to create execution environments tailored for **single-app rollups**. By providing each application with its dedicated rollup, this approach effectively removes the battle for shared state, thus alleviating congestion and minimizing transaction fees.

The core advantages of micro-rollups lie in their support for modular development and their ability to scale. Echoing the functionality of micro-services, micro-rollups autonomously manage their resources and perform their designated tasks with high efficiency. Despite their independence, these rollups are engineered for seamless communication and interoperability with each other, allowing for the efficient sharing of data and functionalities without the complexities often seen in cross-chain interactions.

1. **Custom-Designed for Specific Applications**

   By design, micro-rollups are tailored from the ground up for a specific application, eliminating the need for the cumbersome process of stripping down and reconfiguring a general-purpose VM. This bespoke approach significantly reduces the barrier to entry for developers, streamlining the development process and enhancing the ability to swiftly adapt to evolving requirements or integrate new features

2. **Inherent Interoperability**

   Designed with interoperability at their core, micro-rollups avoid creating isolated silos within the ecosystem. They integrate standardized communication protocols and interfaces right from the start, enabling smooth interactions between different applications. This built-in interoperability mechanism facilitates seamless interaction among disparate applications in the ecosystem

3. **Unlimited Innovation in system design**

   By moving away from the constraints of general-purpose VMs, single-app rollups offer a more optimized and efficient execution environment tailored to the application's specific needs. This focus on bespoke execution environments not only enhances performance and scalability but also opens the door to a wider range of programming languages and development paradigms. Developers are no longer confined to the idiosyncrasies of blockchain-specific languages, broadening the pool of potential contributors and accelerating the pace of innovation.

4. **Elimination of State Growth Concerns**

   Micro-rollups address the issue of state growth, a significant challenge in traditional blockchain systems. By isolating the state to specific applications and optimizing data storage and management, micro-rollups prevent unchecked state expansion, ensuring sustainable scalability and performance over time.

5. **Enhanced modularity**

   The architecture of micro-rollups inherently supports modularity, allowing developers to plug in or swap out components as needed. This flexibility empowers developers to use the best tools available for their specific use case, whether that involves integrating with existing systems, adopting new technologies, or customizing functionalities to better serve their application's unique requirements. This modularity not only accelerates development cycles but also ensures that micro-rollups can evolve alongside technological advancements and changing market demands.

2.1 Micro-rollups are not entirely new

Micro rollups are not a completely new concept; single-app rollups have existed for a considerable time, serving as highly optimized solutions tailored to specific use cases. Examples of early implementations of this concept include Loopring, dYdX, and Fuel-v1, each designed to optimize performance for their unique requirements. Another notable instance is the Hubble Optimistic Rollup, a project that predates the current buzz around app-specific rollups and was instrumental in supporting Worldcoin's token infrastructure initially.

The evolving demands of the blockchain ecosystem, characterized by a push for greater scalability, interoperability, and user-centric optimizations, underscore the need to revisit and rejuvenate the single-app rollup approach

3 Stackr's Micro-Rollup Framework

Stackr provides an opinionated framework for building Micro-rollups (MRUs) in general purpose languages. It takes rollup development to its core elements: operating a state machine off-chain, logging incoming transactions, processing these transactions to update the state, and ultimately recording the state roots and transaction proofs on the parent blockchain. The architecture of MRUs is composed of two distinct layers -

1. Execution Layer
2. Verification Layer

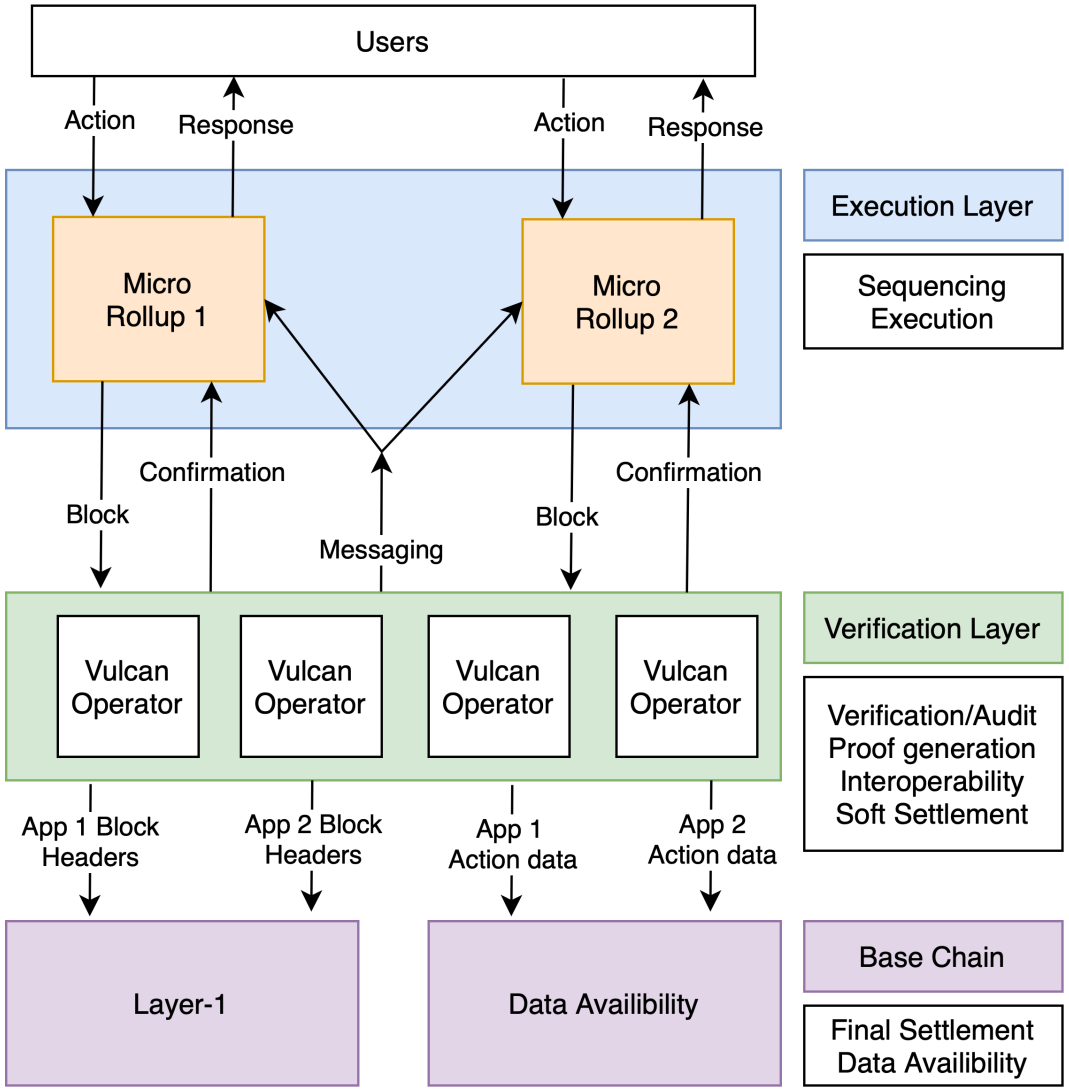

Figure 2: High level architecture of Stackr MRU framework

Hence, MRUs essentially contain state machines that executes certain logic off-chain and then delegate the verification of the execution to another layer, which Stackr calls "Vulcan" that verifies the state updates and pushes the proof of computation on-chain.

3.1 Features

1. **Customization Flexibility**

   Any component of the toolkit can be modified by the developer without much effort. The library also contains abstract implementation of core functionalities which can be overridden.

   The developers can also create custom transaction schemas which we call "actions" that is akin to an API for the micro-rollups.

2. **Modular by design**

   External dependencies like sequencing, data availibility, settlement etc are customizable by the developers and they can choose to use anything they wish in the form of hot swappable modules. Stackr does not wish to lock the developers into a particular ecosystem and provides the ability to mix and match different tech stacks.

3. **Proof Mechanism Choice**

   Developers can choose the kind of proof mechanism they want to deploy with their rollups. Initially Stackr will support pessimistic verification by default using WASM runtime.

4. **Sequencing Choice**

   Each micro-rollup comes with an inbuilt sequencer module. The SDK will support shared sequencing as well through first and third party plugins.

5. **Self Hosting with Censorship resistance**

   The execution layer of the framework can be self-hosted with censorship resistance mechanisms. This allows the execution layer to be fast and each application to be isolated.

6. **Permissionless**

   Deployment of application and joining the verification network will be permissionless and the network economics will govern the functioning of the system.

4 Execution Layer

The execution layer hosts the main application logic. Stackr offers a comprehensive software development kit (SDK) for constructing these applications. This environment delivers a developer experience akin to traditional server-side or backend application development. Additionally, Stackr's roadmap includes plans to offer a mechanism for hosting these applications on behalf of developers in the future.

4.1 Execution Model

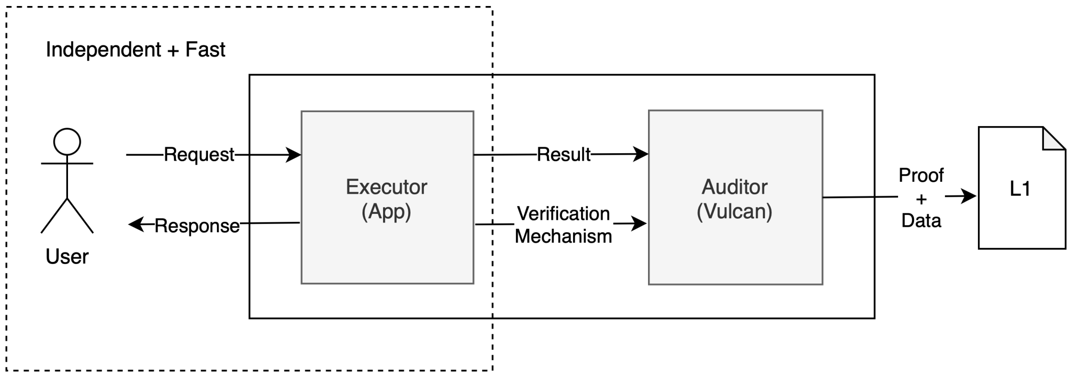

Figure 3: end-to-end data flow of MRU

MRUs deliver exceptionally high throughput, mirroring the instant interaction users expect from web2 applications. Upon receiving a user's request, the application promptly attempts to execute the state changes. If successful, it instantly provides feedback to the user. Subsequently, in according to MRU configurations, the application dispatches the block containing the state update and the corresponding actions to the Vulcan layer. This layer is responsible for verifying the accuracy of the transactions before ultimately recording the data on the blockchain.

4.2 Chain Abstraction

Micro-rollups offer chain abstraction by decoupling the application development process from the underlying blockchain infrastructure, allowing developers to focus on building their applications without needing to navigate the complexities of the blockchain itself. This abstraction layer masks the intricacies of the blockchain, such as smart contract deployment and interaction, transaction handling, and state management, presenting a simplified interface for development. As a result, developers can leverage the benefits of the parent chain while working within a standalone, more familiar and accessible development environment. This is inspired by the CAKE framework

4.3 Custom state machines

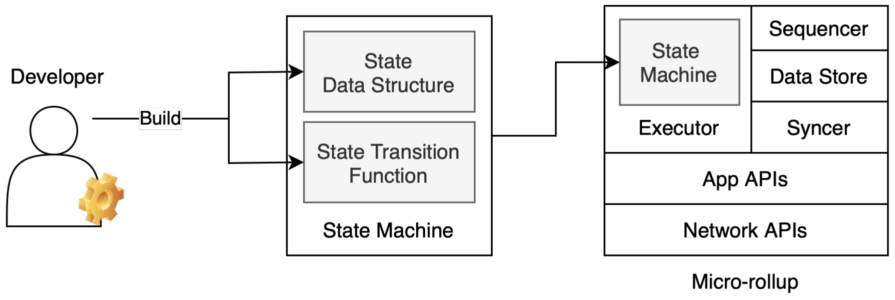

Figure 4: Developer pipeline

Using Stackr's SDK, developers can build their own state machines by defining their app's state model, the state transition rules that can mutate the state and custom transaction formats (known as actions) which app users can use to invoke the mutations. The SDK then wraps the state machine inside an executor and attaches other Rollup APIs to it. Once the Micro-rollup is setup it can be used anywhere in any applications.

Additional helper utilities like Message Pool, Sequencer, Executor etc are also included in the library and can be used along with the micro-rollup.

4.4 Genesis State

The developer defines a genesis state for the state machine which is used to initialize the rollup. This genesis state is also sent over to the verification layer during the deployment process. Any subsequent state transitions are applied from this state.

4.5 Action

Stackr introduces an open and flexible framework for crafting action schemas and, consequently, defining user actions. The choice of the term 'actions' over 'transactions' highlights the adaptability of these actions, distinguishing them from traditional transactions which often follow a set structure of attributes. Through Stackr, 'actions' are fully customizable, accommodating any parameters required by a particular application's state machine, underscoring the framework's flexibility to meet diverse application needs.

4.6 Rollup

The state machine is given to the MRU that undertakes the following actions -

1. Initialize the state machine with a genesis state.
2. Accept incoming user actions in a pool.
3. Order the actions.
4. Create a block from the actions.

5. Execute the block using the state machine.
6. Update the rollup state.
7. Send the updated state root and the actions block to Vulcan for verification.

It is important to note that the rollup and the state machine are independent of each other.

4.7 Sequencer

Stackr offers applications the capability to autonomously manage their action pools and orchestrate the sequencing of these actions. Once acknowledged by the rollup operator, actions are placed into an inbuilt actions pool, where they remain until they are selected for ordering. Stackr includes a default FIFO (First-In-First-Out) builder as a part of its standard library but developers will also get the option to opt into third party shared sequencer via plugins.

4.8 Plugins

Stackr exposes a Plugins API that allows developers to build shareable utilities on top of the current framework. These utilities extends the developer toolkit and can help bootstrap a community driven plugins marketplace.

By default, Stackr provides a Playground plugin that provides a visual interface to interact with and debug their MRUs during the development stage. It shows a high level overview on the state updates and the lifecycle of the blocks produced by MRUs.

4.9 Action Lifecycle

4.9.1 Execution Status

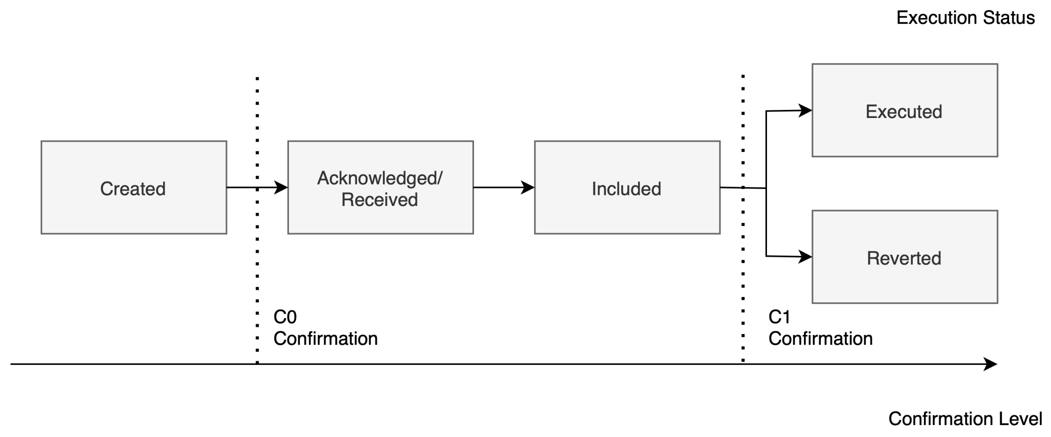

Figure 5: Execution Lifecycle of an Action

The action goes through multiple steps in it's lifecycle as follows -

1. **Created** - The action reaches the MRU.
2. **Acknowledged/Received** - The MRU validates the action with it's schema and generates a proof of receipt and adds it to the mempool.
3. **Included** - The MRU sequencers picks the action, orders it and puts it inside a block.
4. **Executed** - (Inside a block) If the action conforms to the state transition requirements, it causes a mutation in the state. Otherwise the action moves to a 'Reverted' state

After all the actions within the block have been executed, whether they lead to a state mutation or are moved to a 'Reverted' state due to non-conformity with the state transition requirements, the block reaches its final stage of the lifecycle and is considered finalized on the execution layer.

This finalization process marks the completion of the action's journey through the MRU, ensuring that each action has been appropriately processed and its outcome recorded, thereby maintaining the integrity and consistency of the state within the MRU.

4.9.2 Confirmation Status

The framework provides four distinct levels of confirmation or finality for each user-initiated action, each successive level increments in time to completion but concurrently enhances the security guarantee. This nuanced stratification affords developers of micro-rollup applications the opportunity to craft user experiences around throughput-security trade-offs.

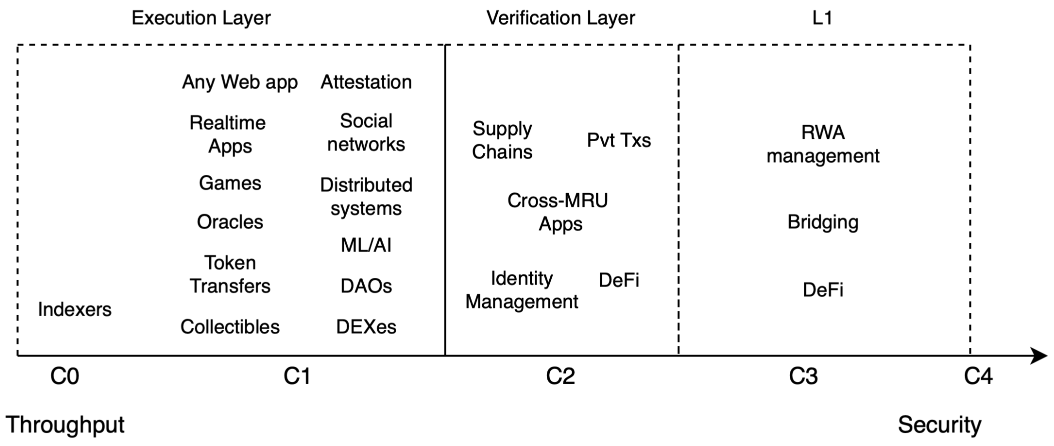

Figure 6: Different user experiences for variety of use-cases

C0 : Acknowledged (**instant**) : The rollup operator generates and returns a signature on the user action. This signature serves as a formal acknowledgment denoting the successful reception of the user action and serves as a rudimentary mechanism for censorship resistance, particularly important considering the central hosting of micro-rollups.

C1 : Executed (**milliseconds**) : The application undertakes the execution of the user action, thereby affecting a change in the application's internal state. This confirmation guarantees that the executed action is incorporated into a block.

C2 : Verified (**seconds to minutes**) : The third level of confirmation involves the transmission of the action batch to the verification layer, which, pessimistically re-executes the state transition for correctness, thus cementing the assurance that the application has properly executed the state transitions.

C3 : Finalized (**minutes to hours**) : The verification layer posts a commitment to this data, either as raw state data or ZKPs, on-chain and provides a guarantee that the batch has successfully traversed to the L1 layer

Stackr splits C3 even further between DA and L1

- C3a: DA Finality (seconds to a minute): This confirmation level involves the posting of the action batch data as a blob to a data availability layer. This ensures that the data can be trustlessly retrieved to derive the rollup's state.
- C3b: L1 Finality (minutes to hours): The verification layer posts a commitment to the action batch data on-chain to verify and guarantee that the batch has been appended to the rollup's canonical chain and is impervious to subsequent tampering or alterations.

C4 (optional) : Settled (hours to days) : For an optimistic use-case the transaction can also implement an optional C4 confirmation level. This level indicates the end of a dispute period, certifying that the data is now immune to further challenge. At this stage, the state within the micro-rollup is irrevocably settled on the L1 chain.

4.10 Deployment

After the state machine is written by the developer. The SDK extracts, compiles, and packages the state transition logic into a WASM binary, which is then uploaded to the verification layer along with the genesis state for future utilization. This process bears a resemblance to the generation and on-chain storage of smart contract bytecode. This design choice also ensures that the execution layer is language-agnostic and is ready to be implemented in any mainstream language that compiles or converts to WASM, like Rust, Go, Python etc.

This uploaded binary along with the genesis state is used by the verification layer to verify the correctness of the blocks generated by the application.

4.10.1 WASM from arbitrary TypeScript

A standout feature of the Stackr SDK is its ability to compile WebAssembly (WASM) from arbitrary TypeScript (TS) code. This feature ensures that regardless of the original development language, applications can be executed in a consistent and secure environment. Furthermore, WASM facilitates deterministic execution, a critical requirement for blockchain applications to ensure predictable and verifiable outcomes.

This compiled WASM is transported to the verification layer where the Vulcan committee uses it for checking the correctness of the state transitions.

5 Verification Layer

The Stackr verification layer, affectionately referred to as 'Vulcan', draws inspiration from the extraterrestrial humanoid species portrayed in the Star Trek universe and known for their rationality. Its primary role revolves around the verification of the correctness of an application's computations. Vulcan operates as a decentralized and permissionless network, composed of autonomous nodes dedicated to establishing a confirmation layer tailored specifically for micro-rollups.

With this framework in place, applications can concentrate their efforts on execution while entrusting the proof mechanisms to this layer. Vulcan seamlessly operates as an intermediary, bridging the gap between applications and the parent chain.

5.1 Verification Model

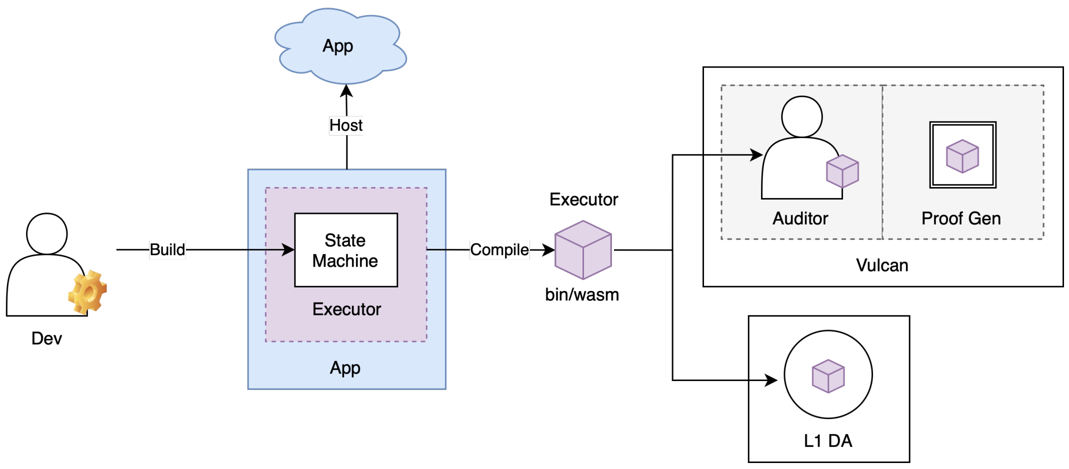

Figure 7: App deployment pipeline

5.1.1 Pessimistic re-execution

Vulcan employs a pessimistic re-execution model to verify the incoming blocks from micro-rollups. The blocks from the execution layer is sent to Vulcan periodically. Vulcan uses the application's WASM binary and applies the actions inside the block one-by-one on the genesis to generate the new state. It then matches the generated state roots with the state root inside the block. If it succeeds the generated state becomes the current state so the next block will be applied on this state. Once the block passes the verification step, it takes the block and submits it on the desired L1 chain.

5.1.2 zkWASM

Stackr is exploring the potential of a zkWASM-based system, enabling Vulcan to execute applications' WASM binaries within a zkVM. This will improve the verification mechanism by creating a proof of re-execution, which is then recorded on the parent chain. This not only enhances the trust model of Vulcan but also optimizes the volume of data recorded on-chain, streamlining the overall process of verification.

5.2 Pre-Confirmation

Since the Vulcan layer is responsible for checking the correctness of actions, it can also provide a confirmation before the data hits the parent chain. As mentioned previously, Stackr's Micro-rollups provide multiple levels of confirmations, which can be used for building different user experiences. Vulcan behaves like a fast finality settlement layer, providing soft confirmations, orders of magnitude faster than the underlying parent chain and DA layer.

5.3 Settlement Mechanism

Vulcan after verification, splits the block into 2 parts. It stores the block headers on the parent chain and the action and state data on a DA Layer. It can also acts as a trusted bridge between DA and parent chain by taking data commitment proofs from DA after settlement and storing that as metadata alongside block headers on parent chain. Stackr will also include other trustless DA to L1 bridging solutions provided by third party providers.

5.4 Interoperabilty

The Vulcan layer also assumes the role of a message passing layer since it has a view of every micro rollup, in lieu of its settlement mechanism. Messages between micro rollups are treated as native actions which allows them to follow the exisitng flow of execution on their respective micro-rollups but undergo joint verfication on the Vulcan layer.

Stackr is exploring a novel design for leveraging the verification layer to perform general message passing across the micro-rollups in the execution layer further strenghthening the micro-services analogy.

5.5 Vulcan Decentralization

Initially the Vulcan layer will be bootstrapped with a single node, called Spock, run by Stackr. However going forward the Vulcan layer will expand and will allow more node runners to participate in the network to secure the applications in the execution layer. The node operators will be incentivized via network rewards.

5.5.1 Bootstrapping the network

Stackr is exploring constructions to bootstrap the validator set using restaking mechanisms that can help Stackr derive security from other larger ecosystem.

5.5.2 Vulcan sub-comittees

Once the network is fully decentralized, Stackr will also explore partitioning the network into sub-committees. This will allow MRUs to select subcommittees that best match the throughput-security trade-off of their use-case.

For instance, a committee composed of restaked operators running Vulcan nodes can attest to the verification of a batch of actions and ensure a requisite level of crypto-economic security while also achieving quick C2 confirmation times. This arrangement is particularly suitable for low latency MRUs with defined security requirements. Similarly, another committee might be formed with nodes which re-execute and validate batches of actions within a Trusted Execution Environment (TEE), offering MRUs with alternative security guarantees and fast finality as well.

The diversity of Vulcan's subcommittees will enable micro-rollups to effectively tailor their security models, choosing from different verification approaches to achieve the best balance between security and throughput according to their specific requirements.

6 Network Economics

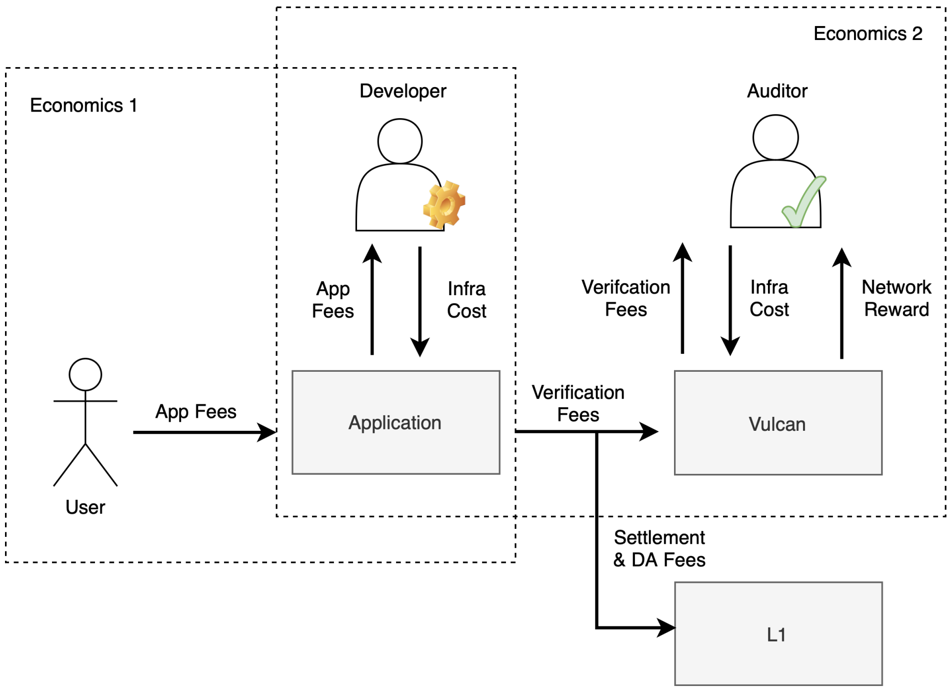

Figure 8: Such a structure of network gives rise to a hybrid network economy

6.1 Hybrid Model

The unique hybrid nature of Stackr introduces an innovative network economics structure, blending elements of both Web2 and Web3 economic models. In this setup, the application and end users form one group, while the application and auditors constitute another. These economic models can operate independently. The application may charge users using a Web2-style API or subscription pricing model, while auditors can levy verification fees on the application, depending on the volume of data sent for verification via a *metering system* or some other format.

6.2 Sufficient Decentralization

At the heart of sufficient decentralisation is the recognition that complete decentralisation, while ideal in theory, may not be practical or necessary for all aspects of an application. Instead, developers are encouraged to identify and decentralise critical components of their applications, thereby achieving a hybrid model that combines the best of both worlds. This approach not only enhances efficiency and user experience but also ensures the integrity and security of the application's core functions.

6.3 Security and Trust Assumptions

Stackr's framework is inspired by the pragmatic approach of web2 applications, recognizing that a balance between centralized efficiency and decentralized security can offer the best of both worlds for many applications. This nuanced approach allows for a "halfway-house" level of security, where critical components are decentralized for integrity and trust, while others may remain centralized for efficiency and user experience. Here's how Stackr navigates the security landscape:

1. **Sovereign Application Control:** Applications within Stackr maintain centralized control over their operations, including transaction ordering. This ensures efficiency and responsiveness but is balanced with mechanisms like censorship-resistant acknowledgments to enhance trust. This centralization is complemented by the potential for a multi-nodal setup, where applications can operate across multiple nodes to enhance fault tolerance and data redundancy.

2. **Shared Sequencing and Verification:** While applications can independently manage transaction ordering, Stackr encourages the use of shared sequencers or a consortium of verifiers for enhanced security. This setup promotes transparency and decentralization in the verification process without sacrificing the application's autonomy.

3. **Verification Layer Integrity:** The Vulcan layer, composed of a committee of auditors, verifies the correctness of batches from applications. Operating under an honest-majority assumption, this layer ensures that data integrity and application behavior adhere to predefined rules. The committee's role is crucial for maintaining system integrity, with mechanisms in place to mitigate potential liveliness failures.

4. **Control Over Settlement:** Applications retain control over the settlement process, with capabilities to invalidate state roots in adversarial scenarios using a delayed settlement mechanism. This control layer adds an additional security measure, ensuring that applications can respond to threats without compromising the overall system integrity.

5. **Exploring Enhanced Security Architectures:** Stackr is actively exploring architectures that exceed the honest-majority assumption, aiming to provide even stronger security guarantees to the verification layer. This ongoing experimentation is part of Stackr's commitment to pushing the boundaries of what's possible in decentralized application security.

6. **Decentralized Verification Expansion:** As Stackr evolves, the Vulcan layer will transition from a singular node to a fully decentralized network of validators. This expansion not only distributes the verification process across multiple parties but also introduces a layer of redundancy and resilience against single points of failure, significantly enhancing the security posture of the network.

7. **Restaking Mechanisms for Bootstrapping Security:** Stackr is exploring innovative restaking mechanisms that allow it to leverage the economic security of larger, more established blockchains. This approach enables Stackr to bootstrap its security model by aligning with the security guarantees of these networks, providing an additional layer of trust and reliability for its operations.

8. **Rollup Forced Exit Protocols:** In the event of discrepancies or malicious activity, Stackr includes mechanisms for a forced exit from rollups. This safety feature ensures that users can retrieve their assets without relying on the rollup's normal operation, providing a crucial layer of user protection and trust in the system's integrity.

9. **Sub-committee Verification for Tailored Security:** Recognizing the diverse security needs of different applications, Stackr plans to implement sub-committees within the Vulcan layer. These sub-committees allow for specialized verification processes that can be customized to match the throughput and security requirements of individual micro-rollups, offering a more nuanced and flexible approach to application security.

7 Conclusion

Micro-rollups will be paving the way for innovative concepts that blur the lines between centralized and decentralized systems. They are more evolutionary than revolutionary and a natural next step in making developer experience refined and easy to use. Web3 is still a niche, even after years of development it has still not reached the level of intuitiveness which web2 has reached and there is a lot of room of improvement. Micro-rollups can help reduce that friction.

1. **Novel App Concepts**

   Stackr grant apps control over their message pools, thus sparking intriguing MEV use cases. With applications freed from VM limitations, new computational mechanisms can also flourish, allowing a concentrated effort on execution and finding more interesting distributed computing use-cases.

2. **Novel Network Economics**

   Stackr framework is built as a hybrid system that marries web2 and web3 economic models prioritizing app developers while also incentivizing the security council.

3. **Open Innovation of Execution environments**

   Since Stackr framework takes care of running state machines and VMs are essentially a subset of state machines, a lot more possibility of experimentation opens up. Micro-rollups can themselves be used to bootstrap other execution environments.

4. **Bridging the Gap Between web2 and web3**

   Developing a Micro-rollup and developing regular backend systems are practically the same experience with Stackr Framework. A lot more things could be migrated on-chain with this setup.

5. **Next wave of a million web3 developers**

   With more and more developers adopting to build web3 native applications with familiar and easy to use toolkit, a lot more applications will start popping up. Smart contracts will become legacy and will be abstracted away.

6. **Towards Mass Adoption of web3**

   More developers mean more applications and with a lot many applications consumers will have a much easier time adopting the new technology. This will surely accelerate the pace of mass adoption of web3 technology.

8 Future Work and Research required

As Stackr continues to evolve and refine its micro-rollup framework, several areas of future work and research have been identified to further enhance the capabilities and reach of the ecosystem. These areas represent both the immediate next steps and the long-term vision for Stackr, aiming to address current limitations and unlock new possibilities for decentralized applications.

1. Inter micro-rollup communication.
2. Liquidity Hub Micro-Rollup for shared liquidity across the Stackr ecosystem.
3. zkVM to generate proof of execution on Vulcan.
4. Progressively decentralising Vulcan and economic security.
5. Multi-nodal micro-rollup systems.
6. Reducing Trust Assumptions Across the Network.