Hey everyone! I am copying this post from myproposal on the Starknet Community forum which discusses an idea on how the Polkadot tech stack can be reused in Ethereum. Essentially, a lot of players in the Ethereum ecosystem are re-inventing solutions for problems like shared security, interoperability etc. which Polkadot has already solved over the last few years. The proposal is to discuss the idea of modifying the existing Polkadot SDK to make it work for zk rollups. I believe this would be a win-win for both ecosystems. Hence I am posting it here again to get your feedback and suggestions on the feasibility of such an idea.

A glossary for some of the terms mentioned

- Madara : A Substrate based framework for building zk rollups
- Pragma, Kakarot, Mangata Finance, Dojo: App chains (parachains) which might use Madara

Introduction

Today, we are already seeing projects starting to experiment with Madara for their app chains. Pragma, Kakarot, Mangata Finance, and Dojo are just some examples. As long as we believe in the multi-chain future and the power of zk scaling, we will only see many more of these projects in the future. However, the increasing number of app chains also raises questions around

1. Decentralisation
2. Composability
3. Development experience

In this post, I will try to explain the problems that arise due to having a lot of app chains and also pose a possible solution to the problem that I consider elegant and optimal for Madara and Starknet. If you are already well versed with app chains and shared sequencing, feel free to jump to the “Wait, it’s just Polkadot all over again” section.

What happens at 100 app chains?

Let’s say we are in a future where we now have 100 different app chains settling on Ethereum. Let’s address all the problems this will cause.

Fragmented decentralisation

Every app chain will need to solve for decentralisation on its own. Now the decentralisation of app chains is not as necessary as that of L1s mainly because we rely on L1s for security. However, we still need decentralisation to ensure liveliness, censorship resistance and to avoid monopolistic advantages (high fees for example). However, it’s also important to note that If each app chain goes on to solve for decentralisation its own way, this will very quickly lead to fragmentation of validator sets. Each app chain would have to develop economic incentives to onboard new validators. Also, validators would need to select what clients they are comfortable with running. Not to mention the huge barrier to entry this creates for developers to launch their own app chains (vs deploying a smart contract which is just a transaction).

Composability

Composability essentially means cross-app interaction. On Ethereum or Starknet, this just means calling another contract and everything else is handled by the protocol itself. However, with app chains, this becomes much more difficult. Different app chains have their own blocks and consensus mechanisms. Every time you try to try to interact with another app chain, you need to carefully examine the consensus algorithm and the finality guarantees and accordingly set up a cross-chain bridge (directly to the chain or via the L1). If you want to interact with 10 app chains with different designs, you would do this 10 different times.

[

Screenshot 2023-10-09 at 11.24.36 PM

1798×1102 118 KB

](https://ethresear.ch/uploads/default/original/2X/e/e23c8a6aa296149978bb74441e520a9b8a22b0f9.png)

Development experience

Solving for decentralisation and bridging is not easy. If this is a requirement for every app chain, it will make it very difficult for the usual smart contract developer to ever build his own app chain. Moreover, as every app chain tries to solve these problems in its own ways, we will soon see different standards being followed by different chains making it even more difficult to join the ecosystem.

Shared Sequencers can solve this

Now if you're following the app chain space, you might have heard of the term "shared sequencers". It's the idea of having a common set of validators for all chains that aim to solve the problems mentioned above. This is how it works.

Shared Decentralisation

The very essence of shared sequencers is that you don't need to have a different set of validators for each app chain or L2. Instead, you can have a really efficient and decentralised set of validators that sequence the blocks for all the chains! Imagine blocks that contain transactions from 100 different app chains. You might be thinking this is going to really bloat up the sequencer as you need to be able to handle execution engines for each app chain.

Well, you don't!

Since today almost every sequencer is centralized, the sequencer is thought of as a single application that collects transactions, orders them, executes them and posts the results on the L1. However, these jobs can be separated into multiple modular components. For the sake of this explanation, I will divide them into two.

1. Order engine: This is responsible for sequencing the transactions in a specific order. Once this order has been decided by the order engine, it MUST be followed. This is enforced by committing this order on the L1 and forcing L1 verifiers to check if transactions were executed in the required order.

2. Rollup engine: The rollup engine is basically everything else the rollup does - collecting transactions from users, executing them, creating proofs and updating the state on the L1. Ideally, this can be broken into more components but we would avoid that for this post.

Here, the ordering engine is the shared sequencer and the rollup engine is basically all the rollup logic. So the transaction lifecycle looks like this

[

Pasted image 20231009201722

726×1044 22.7 KB

](https://ethresear.ch/uploads/default/original/2X/b/b6e29634260ca9a78d4871744fd237fc0c7888bf.png)

The shared sequencers basically order transactions across rollups and commit them to the L1. Hence, by decentralising the shared sequencer set, you basically decentralise all the rollups linked to that sequencer set! To get a more detailed idea of the working of shared sequencers, you can also refer to this amazing article by Espresso.

Composability

One of the major issues of composability is understanding when the transaction is finalised on the other app chain and accordingly taking actions on your chain. But with shared sequencers, both the composable rollups share blocks with each other. So if a transaction rolls back on rollup B, the entire block is rolled back, and this causes the transaction on rollup A to revert as well.

Now this surely sounds easier said than done. For this. communication between rollups needs to be efficient and scalable. The shared sequencers need to come up with proper standards on how rollups can communicate, what should cross-chain messages look like, how to deal with rollup upgrades etc. While these are solvable problems, they are complicated and not easy to solve.

Developer experience

While the shared sequencers do abstract the decentralisation aspect and make cross-chain messaging easier, there are still some standards that every chain needs to follow to be compatible with the shared sequencer. For example, all rollup transaction needs to be transformed into a general format that the sequencer understands. Similarly, blocks from the sequencer need to be filtered to fetch the relevant transactions. To solve this, I would assume shared sequencers would come up with rollup frameworks or SDKs that abstract the boilerplate code and expose only the business logic part to app chain developers.

Here's a diagram of how app chains will look with shared sequencers

[

Screenshot 2023-10-09 at 11.41.23 PM

1386×1164 175 KB

](https://ethresear.ch/uploads/default/original/2X/3/3f2c389ae5a1968490c5b976f9052d54cceeff6d.png)

Wait, it's just Polkadot all over again

Polkadot started working on the multi-chain future much before Ethereum. In fact, they have been working on it for more than 5 years now and if you're familiar with Polkadot, you might have noticed that the above design is basically re-inventing a lot of things Polkadot has already done!

The Relay Chain (Shared Decentralisation)

The Relay chain is basically the ordering engine + L1 in the sequence diagram above. The relay chain

1. Orders transactions across all the parachains (rollups)
2. It verifies the transactions executed correctly (instead of zk verification, it actually re-runs the execution code of the rollup to verify the state diffs)

[

Pasted image 20231009204452

2000×1122 203 KB

](https://ethresear.ch/uploads/default/original/2X/b/b51d2d91802dce3eb9745944913541fb65baee62.jpeg)

You might have realised the relay is basically the shared sequencer we discussed above. Except for the fact that the relay chain also needs to do verify the execution (as Polkadot is an L1) whereas we leave that to Ethereum.

XCM and XCMP

We mentioned in the previous section that if every chain built its own methods to interoperate with other chains, we would soon be bloated with different standards and formats across all chains. You'll need to keep track of all these formats for every chain you interact with. Moreover, you will also need to answer questions like what happens if a chain upgrades? However, these problems can be tackled elegantly by introducing standards that all chains must follow.

As you might have guessed, Polkadot has already done this. XCM is the messaging format and XCMP is the messaging protocol that all substrate chains can use to communicate with each other. I won't go into the details of it but you can read about it here.

Substrate and Cumulus

Substrate is a framework developed by Parity that can be used to build blockchains. While all parachains on Polkadot use substrate, substrate is actually built in a chain-agnostic way. The framework abstracts all the common aspects of a blockchain so that you can just focus on your application logic. As we know, Madara is built on Substrate and so are Polkadot, Polygon Avail and many other projects. Moreover, Cumulus is a middleware on top of Substrate that allows you to connect your chain to Polkadot.

So continuing our analogy like before, Substrate and Cumulus can be thought of as substitutes to the rollup frameworks that allow you to build app chains and connect them to the shared sequencers.

Shared Sequencers → Relay Chains

Composability → XCM and XCMP

Rollup frameworks/Stacks → Substrate and Cumulus

[

Screenshot 2023-10-09 at 10.15.10 PM

1578×888 91.6 KB

](https://ethresear.ch/uploads/default/original/2X/0/04656d8573d423c7fa585b73dd39dab403a94365.jpeg)

So yes, it's pretty much Polkadot all over again! Apart from this, Polkadot and Parity have some of the most experienced and funded teams that continue to build Substrate and Polkadot to add more features and make it more scalable. The technology is already battle-tested for years and has a ton of dev tooling out of the box.

Settle Polkadot on Ethereum?

While it's true Polkadot did start building the multi-chain future way before Ethereum, there's no denying that as of today, Ethereum is the most decentralised blockchain and the place where most of the apps and liquidity rests. However, what if there was a way to bring all the Polkadot tech into the Ethereum ecosystem?

There is! In fact, we have already started this with Madara. Madara uses the Substrate framework to allow anyone to build their own zk-powered L2/L3 solution on top of Ethereum. What we need next is the Polkadot relay chain in the form of a shared sequencer. So basically, if we can reuse the Polkadot relay chain but

1. Remove the verification part as that happens on the L1 via zk proofs
2. Commit the order of transactions to the L1
3. Optimise the nodes and consensus algorithms to support Tendermint/HotStuff

We can get shared sequencers as mentioned before. Obviously, this is easier said than done. However, I believe this path is more pragmatic than rebuilding the sequencers, standards and frameworks from scratch. Polkadot has already solved a lot of problems in a chain-agnostic way which we can borrow for Ethereum. As a side product, we get

1. An active set of developers that continue to build and educate the world about Substrate
2. An active developer tooling set and a strong community
3. A lot of active parachains can choose to settle on Ethereum as well if they wish to do so (we saw Astar do the same recently with the Polygon CDK)

Conclusion

My main idea behind writing this post is to open the discussion amongst the broader ecosystem of Starknet and Ethereum. I feel the shared sequencing model will play an important role in the decentralisation of not only Starknet but also all the app chains that consider building on top of it. As long as we are confident about the app chain thesis and zk scaling, a thorough analysis of the shared sequencing model is inevitable. Moreover, as Madara is moving towards production and Starknet has started its work on decentralisation, I feel this topic is now important to address. Hence, I request everyone reading this to leave any feedback/suggestions you have about the topic. Looking forward to reading your thoughts!

Appendix

Polkadot vs Cosmos

Cosmos, similar to Polkadot, has been solving for a multi-chain future for many years now. As a result, it has made a lot of developments with the Cosmos SDK and IBC and we also see a lot of app chains building on top of the Cosmos ecosystem. Hence, it's only fair to consider Cosmos as well for this approach. My personal view on this topic is that Polkadot is more relevant as it solves the shared sequencers problem whereas Cosmos requires each app chain to build its own validator set. Moreover, Substrate has always been built in a chain-agnostic way to allow developers to build blockchains with no assumptions about consensus algorithms or the Polkadot ecosystem. This is also the reason why we chose Substrate for Madara. However, with that being said, my experience on Cosmos is limited and would love to hear more on this from the more experienced folks! You can also find more about the comparison of the two networks here.