Current fee payment systems in rollups have deviated from how L1s traditionally operated with the user setting and paying for the gas for that service. RAAS (Rollup-as-a-service) providers instead have monthly subscriptions that the developers are responsible for paying not the actual users of the service themselves. RAAS appears as a temporary solution for what is still an immature framework. It is quasi-monolithic in design, doing everything but the DA itself. Thus it's not as robust as a fully trust minimised modular system.

Base rollups could be considered the closest frameworks to what we're looking for but it's inefficient to require the entire network to execute pay for blob transactions for just 250 bytes of data (a typical transaction size). Right now, 250 bytes would require 2,000 gas, but there's also another 70,000 gas in overhead (signature verification etc.). It becomes cheaper if Celestia supports permissionless batching (or sequencing). But how do these payment systems look like. Celestia has an opportunity to dictate these systems by supporting some form of off-chain micropayments whereby Celestia (and TIA) is used to settle (and I mean the financial term) small, out of band payments for services such as: sequencing, batching, proving, executing etc… allowing for a plethora of permissionless systems.

I propose a new sdk module that supports the following flow. For the example I am talking about an interaction with a rollup user, say for an orderbook, and an aggregator that batches and submits blobs on Celestia.

1. Rollup user initiates a cheque account with the aggregator whereby the user decides a value to be locked up. In this example we pick a value of 10 TIA. It could have on chain state that looks as follows (naive solution):

type State struct { owner sdk.AccAddress recipient sdk.AccAddress lockedUp uint64 nonce uint32 }

1. Rollup user submits a trade transaction to the aggregator to be published on Celestia. As well as the data itself, the submitter includes a cheque

. This is a signed over message for the amount to submit the transaction, perhaps 100 utia (a really small amount). Note in this example, the transaction is prepaid. If it were to be postpaid the rollup user would provide the cheque

only after they had proved inclusion of the blob.

1. As well as validating the rest, the aggregator searches for a cheque account under the users address and their address, checks that the locked amount is sufficient and accepts the payment.

2. The next time the rollup user submits a trade, they append to the previous value. If they wanted to pay 50 utia, then they would sign over 150 utia.

3. When it makes sense, the aggregator can cash in the cheque. They will receive the funds from the locked account. They can continue to use it and accept messages so long as they see sufficient funds locked.

4. If the user wants to withdraw the funds from their locked account, similar to staking, they wait three weeks. In that time, if there is outstanding cheques, it's up to the creditor to redeem them.

The example here uses the term "aggregator" but theoretically any service provider building with Celestia could be used. A RAAS provider could offer a contract whereby the users pay (instead of developers) for both the inclusion, the sequencing and the execution all together.

I haven't spent much time yet in the solution space but I have a hunch that this could be combined with the existing staking component to have less on-chain state.