Summary

tl;dr wonderings about using an oracle like Augur as the dispute mechanism for an on-chain data feed

Context - The Layer 2 Pattern

The pattern of “Layer 2” systems entails using the “expensive and slow”, but decentralized (and therefore trustworthy, uncensorable, and legit) base chain as a dispute and settlement layer for a separate-but-dependent “cheap and fast” centralized (and therefore potentially censor/attackable) system. The idea is that the centralized system can be designed in such a way that any actions or operations taken produce artifacts that can be used on the decentralized layer to enforce the rules of the system and place blame for misbehavior if need be.

In Josh Stark’s words from his Making Sense of Ethereum’s Layer 2 Scaling Solutions article [0]

Cryptoeconomic consensus gives us a core hard kernel of certainty

— unless something extreme like a 51% attack happens, we know that on-chain operations — like payments, or smart-contracts — will execute as written.

The insight behind layer 2 solutions is that we can use this core kernel of certainty as an anchor — a fixed point to which we attach additional economic mechanisms.

This second

layer

of economic mechanisms can extend the utility of public blockchains outwards, letting us have interactions off

of the blockchain that can still reliably refer back

to that core kernel if necessary. [0]

Disputable data feeds

Piggy backing off this framing, I’ve been thinking about the viability of using an “expensive and slow”, but decentralized oracle as the dispute mechanism for a “cheap and fast”, but centralized data feed.

The data feed would consist of a single actor, a bonded operator, having access to periodically post data chunks to an on-chain feed. This data would ideally be something that is “objective common knowledge” and highly available, that anyone could essentially look up for themselves. These data feeds have a dispute timeout period; whenever new data is posted, anyone can dispute the data’s legitimacy within the timeout period. After the timeout period, data chunks will be considered “finalized”, and applications building on top of the feed can regard the data as legitimate. The dispute process entails querying the “expensive and slow, but decentralized” oracle for the legitimacy of the contested data chunk.

For the sake of this conversation, we can assume Augur is used as the oracle; I can describe an example of what this might look like in practice to color the idea better.

Example - American Football Stat Feed

ESPN, Yahoo, and NFL all want to run their fantasy dapp without needing users to trust their on-chain data feeds. They construct a system like I’ve described above, where one of them posts the hash of some serialization of the past week’s stats. Yahoo is nominated to be the operator, and they can remain the operator until misbehavior is detected via a successful dispute. The hashes of the stats are the data feed, and the feed is updated once a week, at least 24 hours after the week’s events have ended, allowing discrepancies in the statistics to be ironed out between different sources and the information to propagate into being public knowledge. If the dispute period is a day, then 24 hours after this data feed is posted, dapps building on this feed can update their states on-chain to reflect the newly finalized new values. Since the data itself isn’t on-chain, contracts could be updated to the new state once a data chunk is finalized using a new state and a validity-proving zkstark. Note that, since the nature of the data is that it is objective and highly available, clients and frontends can optimistically determine what the stats will be and reflect the up-to-the-minute results on the frontend, trusting that on-chain finalization and subsequent settlement will happen eventually.

If Yahoo (the hypothetical current operator) decides to misbehave and post incorrect data, ESPN, NFL, or any other external actor can put up a bond to open a dispute for that data. This dispute would open an Augur market, with the query being something like Does the statistics for Week 5 of the NFL season, serialized and compiled according to , hash to

. Because the serialization format and the rules of the system are so thoroughly defined, and the data is widely available, anyone should be able to validate the data chunk and see if the dispute is legitimate. The outcome of the Augur market would determine how the dispute is resolved. If the dispute resolves in Yahoo’s favor, the disputer’s bond is added to Yahoo’s operator bond. Otherwise, if Yahoo is found to have misbehaved, their bond is slashed and some reward is paid out to the disputing actor, and the disputing actor is named the new operator, with their reward and dispute bond being used as

their operator bond.

The disputer-turned-operator could then sell/transfer their ability to be an operator, to ESPN, NFL, or any other interested party for some amount of money.

Some notes

- Data availability isn't an issue, thanks to the nature of the data. Yahoo shouldn't need to host the data and can't withold the data; if NFL or ESPN or Twitter or any other source has the relevant data, it should be straightforward to serialize the data consistently and either verify the posted hash or see that it's incorrectly calculated and subsequently dispute it. The data being correct implies its availability, since it's "objective common knowledge".
- If clients/frontends are going to be displaying stats before they're posted on-chain anyway, they could also monitor and locally verify chunks posted to the data feed once they're posted, surfacing a warning and recommendation for dispute for any users. For any erroneous chunk of data posted, the system needs only one actor to raise a dispute.
- An attack on a data feed may entail posting bad data and attempting to censor disputes from all actors while the dispute period passes. If it's assumed that disputes cannot be censored, the only way the attack can succeed is if the Augur market resolves incorrectly.
- Thus, as long as there is at least one honest actor checking the feed within the timeout period for each data chunk

, the cost to an attacker to get dishonest data finalized is the min of

and

- Thus, as long as there is at least one honest actor checking the feed within the timeout period for each data chunk

, the cost to an attacker to get dishonest data finalized is the min of

and

- The Augur market resolving in a fork is an edge case that both the data feed layer and dapps built on top of that will need to specifically account for. An easy default would be to refund everyone's money and partially slash+unbond the data feed operator, essentially "shutting down".
- In the case of a dispute being successful, it's important to slash some of the operator's stake, otherwise operators could "frontrun" their own misbehavior and grief the system.
- Ideally you'd be able to tweak the parameters of the "expensive and slow, but decentralized" oracle to adhere to different security levels for different data feeds. The dispute taking too long could be an unfortunate griefing vector. If each dispute takes a week, and data is expected to be posted every hour, a single dispute could stall the system's settlement for an unacceptably long time.
- Operator incentives aren't fully thought out here. Perhaps there's enough social incentive for Yahoo to put up a bond and run the feed if all the dapp frontends annotate that they're "Powered by Yahoo". Perhaps other types of data feeds could offer incentives paid for by canonical services on top of the data feed, similar to BTC Relay taking a fee from on-chain reads of block headers[1]. Perhaps the operator bond is partially community funded, and the operator gets paid out a small-but-more-than-the-gas amount per data chunk that is finalized.
- Needing operators to lock up capital is not great, but ideally social incentives and the reputation of the operator being a "known entity" (eg. Yahoo) can act as a deterrent as well.
- In cases where the feed can be neatly coupled with on-chain services to provide incentives for decentralized posters, the system could almost be framed as a kind of DAO

- Needing operators to lock up capital is not great, but ideally social incentives and the reputation of the operator being a "known entity" (eg. Yahoo) can act as a deterrent as well.
- In cases where the feed can be neatly coupled with on-chain services to provide incentives for decentralized posters, the system could almost be framed as a kind of DAO

- Operator reelection is tied closely to operator incentives. If there are real incentives to become the operator, then there could probably be a better way to optimize operator election than just having the first to identify a dispute win.
- Other use cases: temperature feed updated hourly, block header relay, real-world-asset price oracle

"Layer 2"

I've temporarily dubbed this type of pattern a "layer 2 oracle", since in the optimistic case it can operate entirely without touching the oracle, but in the dispute case its security falls back to the security of the oracle, assuming censorship is not

possible. I'm interested to hear if there are other framings that come to mind that may make more sense.

Closing

I'm not actually building anything on top of this, but I've had the thoughts bouncing around in my head for a while and wanted to get them written down and see if anyone has any thoughts on this or similar directions. Thanks for reading!

Links

- [0] https://medium.com/l4-media/making-sense-of-ethereums-layer-2-scaling-solutions-state-channels-plasma-and-truebit-22cb40dcc2f4
- [1] https://github.com/ethereum/btcrelay#getblockheaderblockhash