TL;DR:

The goal of this project is to allow smart contracts to subscribe to events emitted from other contracts, on-chain and in a verifiable manner.

Introduction

Event logs produced within the context of a transaction are only accessible off-chain and by querying clients. Enabling smart contracts to take action based on these events, could potentially introduce interesting use cases.

Related Work

The architecture resembles two others, namely, oracles and subscription-based payment models.

It can be seen as an oracle, the query response of which is Ethereum transaction data. Current oracles usually have a pull-model, where the contract requests their required data from an off-chain source. The proposed approach however, employs a push-model, where any normal event emitted by any contract could potentially trigger the invocation of some subscribers. This means, contracts could subscribe to events from already deployed contracts. The downside being that, they have less control over how often their callbacks are invoked.

Moreover, the mechanism could allow conditional delayed execution, meaning, the end user would not need to start a transaction manually, and actions could be taken conditional on the occurrence of an event elsewhere in the network. It can be seen as a generalization of the current subscription-based payment models. The difference being that it's not only for payments, and the actions need not take place at regular intervals. However, regular events can be achieved for example with the help of a cron contract that incentivizes users to emit events at regular intervals. This generalization however comes at a cost.

Method

A Registry

contract would store subscription information, and additionally deposits from subscribers and the price they're willing to pay per invocation. Upon emission of an event, if there are any subscribers with an offer >= gasEstimation

, miners would submit the event log data, along with a proof that the log exists within a particular block, to the registry, earning ether, and causing the invocation of the subscriber's callback. Because it's the miners who pay the gas fee, it might be possible to improve UX by making some of the transactions async, and having other means of funding them.

Ideally, the verification could be done via a merkle proof that shows log l

is in receipts trie of the block b

with block hash h

(b

should be among the most recent blocks, as contracts only have access to most recent block hashes, and older events are probably not useful anyway), or if such a proof is infeasible or inefficient, via zero knowledge proof systems. In the worst case, it would be necessary to revert to stakes and challenge games, which would introduce some delay from the emission of an event, and its corresponding callback being invoked, and additionally would have weaker guaranties.

Implementation

You can find a very primitive example of a Registry

(along with a subscriber and an emitter) here. The incentive and proof parts are still being worked on.

Final Remarks

Thanks for reading the proposal. I apologize in advance if there are glaring mistakes in the proposal, we're less experienced than many here, and as such we'd love to hear your feedback on whether you think this is a direction worth pursuing, and if so, how it can be improved, the possible attack vectors, the verification, etc.