Per my understanding, the following product requirementin v0.2:

On another topic

Basic distributed indexing with custom projection functions

- Nodes can subscribe to indices with custom projection functions (computed over state)
- Deltas will be broadcast to them when the state changes
- Basic DSL or analytical tools to scale this efficiently

is not yet satisfied by the v0.2 specs. Before making a PR, I thought it would be a good idea to put a quick proposal here and sync between relevant parties (paging @jonathan @AHart @tg-x @mariari @ray).

To deconstruct the product requirement a bit more, for v0.2 I think we want specifically:

1. An indexer “process” (engine instance) that can be run separately from the executor node (could be on the executor node, but could also just be another node subscribing to all the transaction results / blocks)
2. The ability to write projection functions in Juvix (compiled to Anockma) which can be sent to this indexer process and subscribed to - then associated with a particular pub/sub channel to which the indexer engine instance will broadcast updates to when the result of computing the projection function changes.

For v0.2 I think we can defer actual distribution of the computation

involved in indexing, optimization of such (e.g. avoiding recomputation, more advanced “database indexing” techniques), proofs of indexing correctness, and any incentive modeling whatsoever. We should, however, have some notion of gas limits to ensure that the indexer engine instance can’t just be trivially DoS’ed.

From the application user perspective, this kind of indexing is intended to provide something that feels like “state sync”, in the sense that a UI (e.g. web application) could subscribe to state updates, and display the latest state in near realtime as soon as any transaction changes it, where most of the involved computation is offloaded to servers elsewhere (for v0.2, a single indexer engine instance, with more complex topologies supported in subsequent versions).

In accordance with these specifics, I propose:

- an indexer engine as part of the ordering machine (where it seems to fit best)
- messages to subscribe to new projection functions (submitted as Anockma functions), and unsubscribe from previously-subscribed ones
- indexer engine behaviour in line with the above descriptions

How does that sound? Does this fit well with engineering’s indexer work thus far, or would it be a different direction?

Also, is there a nice way to broadcast “data structure deltas” (roughly a laJSON Patch, but perhaps in a more efficient binary format) for Anockma terms, or would we need to create such a mini-protocol?