We've gone over specs-code differences in preparation for our discussions next week, here's the notes:

Proper Scry and Deletion

Deletion is an overloaded word; our storage is architected to be very much append-only in semantics, and I've posted before about the difference between "delete" (just a write) and "expunge" (a rare and destructive operation, you do this if you e.g. leaked your valuable private key into state, or choked it with a 100TB blob, or something)

(Anoma Node: Proper Scry support)

Notes about the namespace buzzword

Transport Engine and Supervision Around it

Transport Engine Design

The transport design currently has some differences; it's poor code design for a message sender to say "I am now sending a message to the Transport Engine," since this means it has to think about the local/remote distinction in ways which are well outside of its scope. Transport, in the above design, lives on the message-reception side, handling getting messages to their destination.

Pub Sub System

Anoma Topics Meeting Digest: Global Data Brokers [Protocol Design

](/c/protocol-design/25)

The general design We wish to design a global data broker that gets messages about a particular topic. This would likely be an actor or a pool of actors. From there, we would make sub brokers as a custom query. These would have a defined elixir module type with potential arguments. Users can make more sub queries filters from there, having a chain of topic filters for whatever information someone would want. Let us exemplify this by having 2 examples:

I am interested in all spawned workers f…

There's implicitly three sorts of message possible:

- one which is fire-and-forget: you send it. if you do happen to get a "reply" causally downstream of this, it's in the form of an entirely new message. This is GenServer cast, and is the default sort of message.
- one with synchronization requirements on the sender: when you send such a message you stop and wait for a reply to it. This is GenServer call.
- one which isn't sent to a destination, but broadcasted to anyone interested in it. This is the "pubsub" pattern, implemented as EventBroker (see my design at Anoma Topics Meeting Digest: Global Data Brokers)

Replay System

There are two kinds of state changes in Anoma Node:

1. User-input states, i.e. states of form change_state_X_to_Y

which puts :X

field to have value Y

in the actor. Example: set_timer(X)

sets a timer for the pinger execution to X

. (or smth like that)

1. Anoma-generated states, i.e. all other states. Semantically, all states which are deterministically changed via user-input yet which is not directly identical to the input. In other words, the state-changes are "caused" by elixir functions producing some computations using the user-specified input or some input which is provided via a causal chain from the user input. Example: user submits transaction code using Mempool.tx

. The resulting state changes are provided by "internal computations" (if you accept the separation between user and system).

Problem: snapshotting. Getting state from all the actors synchronously via state-dumps can cause synchronization failures.

Solution:

a. Forget about state information of type 1, i.e. user specified. The user should know themselves what they want to set it to. They either have the info in a config file (maybe best to have all such info in config) or set them directly

b. For all state changes achieved via 2, we replay them from a specified checkpoint.

What we mean by checkpoint

: can vary, a good time to do this is block commit.

What we mean by replay

: we have a limited number of user-inputs of type 2 which affect state. Specifically, Mempool.tx

and Mempool.execute

. If we save in a linear fashion which transactions were submitted when (with which ID's and with which orders) and when execute commands were sent, we can ask the system to just recompute all the submitted transactions to bring us to a latest checkpointed consistent state.

This isn't particularly groundbreaking, log-and-checkpoint is the way every replicable database works, but unfortunately mistakes were made and "state dumping" was pursued for a while.

Philosphical Difference on Transparent RM

Being discussed actively in this thread:

ART Report: Resource Machine [Protocol Design

](/c/protocol-design/25)

There still appears to be a great deal of confusion about why there is even a transparent resource machine at all, which I struggle to put into words differently than I already have. Here is another attempt. The transparent resource machine is not a real thing. It isn't a product for doing things outside privacy of one's own computer. It should probably be disabled entirely on public-network nodes. It's not meant to have features ripped out or put in compared to the shielded resource machine (h…

Philosphical View on Checkpointing

On new block creation, the block itself is a good time for a checkpoint.

If a block is not created, we can roll back the events and come up with new ones that satisfy consensus.

It's not precisely a rollback, consensus are also events, but it's similar to one in that the state reset to comes from the last checkpoint.

It would be good if specs could discuss this, our latest scry changes:

Anoma Node: Proper Scry support [Protocol Design

](/c/protocol-design/25)

Proceedings from SKAN Meetings: Jul 18/24 Currently as of v0.17.0 of Anoma, our implementation of snapshotting and our mnesia tables are laid out, leaves a lot to be desired. Thus the focus of this proceeding was to tackle pain points in our architecture. The Rough Plan During the proceedings we came up with a proper way that our scry ought to work and how storage should interact with it: [Untitled-2024-07-18-0013] This diagram describes a layout of a few things:

key_A has 3 timestamps fo…

includes a good point for actually making this work from the codebase.

In some threads we might overload "Block" to mean "checkpoint" where in specs it means "narwhal root node" or something; we should fix this language.

Hardware Abstraction Machine

Doesn't seem to interact with the system at large and does things which can't be made distributed; it's unclear what this is for. V2 might make it clearer.

Measurement Engine

The following section is personal opinion inserted by Jeremy, I don’t really have a comment:

[

](https://www.youtube.com/watch?v=JpGrIA-aUfE)

The ideas of trying to use meta information to influence decisions is nice, however I think it ought to be throught out from a holistic POV.

The talk above I give about trying to understand reflective design and examples like CLOS and smalltalk are a good starting point for doing something like this correctly.

Identities

This is discussed at https://github.com/anoma/anoma/issues/571

Anything else?

What do specs v2 drafters have in mind?