As a summary from this week, I think I have realized that the execution and data structure components of the ETH 2.0 roadmap are still very much in flux, especially the data structures.

One idea I have been thinking about this week is a few concepts relating primarily to the “actor model” that Erlang and Elixer uses (Read this for more info), thanks to some conversations with@expede. Another undercurrent is the idea of treating assets (tokens, Ether, other forms of controllable state with value) as first class citizens in this “asset control network” we’re building called Ethereum. Agoric is also doing some neat things along this concept (see talk here). @johba’s Plasma Leap (see thread here) also has some very similar concepts of a token that controls it’s own state.

More narrowly, the idea I have been playing with would be functionality that allows certain “actors” (like token contracts) to issue pieces of state to other actors (regular users). We would store these pieces of state in a new key-value data structure called “asset storage” (similar to how Ether balances are kept). Each asset could contain specific logic for transfer to other actors, or could link back to a function in that contract for more complex coordination logic (conditional transfers). They could issue logs, etc.

In this data structure, the issuing contract would be the key, so access control is built into the data structure as we could enforce only the issuing contract is allowed to modify given state or make external calls, etc. For upgradability, there might be a way for the issuer to change the key by calling a specific operation. Transfer of simple, uncontrolled tokens could even happen without ever requiring interaction with the issuer contract. Minting comes from the issuer, and burning could be a feature of the token internally. I am sure there are other useful features for an asset to have, but I think this can be reduced quite nicely to a few core features, and augmented with allowing the asset to store it’s own rules.

A nice side effect of all of this is that tokens (and other relevant state e.g. access rights to the issuing contract) no longer fully depend on a central contract for their control, which means the central issuer contract has much less authority over these valuable pieces of state in case of a hack.

This also has another nice side effect relating to storage rent: since much less state is being stored in central contracts, the concept of who pays for a particular contract to stay alive is reduced (we could define minimums to eliminate it). Each account is now in charge of managing their own assets, and in a rent scheme, this means ensuring the upkeep of their assets too. Rent could allow you to “drop” tokens you don’t care about.

This idea is definitely not fully baked, but I would say that now is the time for re-architecting the data model in order to make sure that the future state of Ethereum makes more sense to developers (I think the actor-asset model is very clear), and can be sustainable as well.

Thoughts?