I think the most underdetermined remaining part of application interface standardization is figuring out how applications interact with service commitments (promises to store data, perform computation, perform ordering, etc.). In particular, I would expect that applications rely on service commitments in the following ways:

- In order to maintain state over time, applications rely on storage services.
- In order to access state efficiently, applications rely on availability services (serving data over the P2P layer), which will likely be tightly coupled with some compute services to index over that data and serve computed results.
- In order to compute longer-term aggregate statistics, applications will rely on perhaps a separate subclass of compute services.
- In order to provide counterparty discovery for user intents, applications will rely on compute and bandwidth services (i.e. solvers).
- In order to order user transactions, applications will rely on ordering services.

I think we should be able to clearly eludicate what services a specific application requires by:

- describing the state of the application (which will be sharded over the network) and who is expected to store each part at each point in logical application time
- describing the services the application wishes to provide to users, which will entail availability, compute, networking, and ordering services

We will need to figure out which parts of these definitions are parts of what exact data structures - the same application (in the resource machine sense) can be coupled with different sets of promises chosen by different users. Perhaps it would be helpful to introduce a notion of an “application service configuration” or something like this. With a sufficiently clear definition of a service configuration, we should be able to analyze whether or not an application can “keep promises” to its users - of a more gestalt form - on the basis of whether the service providers keep their constituent component promises which are part of the application service configuration.

Curious for thoughts from @nzarin @vveiln @Michael.