This thread is just a reminder from the conversation in Taipei this morning. Wanted to open up the conversation to the entire Ethereum community. cc @vlad @vbuterin @karalabe @pipermerriam

context

cross-contract communication may push complexity out of the protocol layer to the developer experience. for example, train-and-hotel problem requires service providers to provide ability to hold and reserve but revert within a time out period.

tl;dr

we should engage with the dapp developer community and the broader developer communities to identify the best programming patterns & abstractions to handle this problem, and design sharding with the end product (devEx) in mind.

Relevant posts:

Merge blocks and synchronous cross-shard state execution [

Sharding

](/c/sharding)

We know that with merge blocks and clever use of fork choice rules, we can have a sharded chain where a block on one shard is atomic with a block on another shard; that is, either both blocks are part of the final chain or neither are, and within that block there can be transactions that make synchronous calls between the two shards (which are useful for, among other things, solving train-and-hotel problems). However, there remains a challenge: how do we do state execution for such blocks? That …

Cross Shard Locking Scheme - (1) [

Sharding

](/c/sharding)

@vbuterin @JustinDrake The basic idea is to use read and write locks to ensure a transaction that references data on many shards can be executed atomically. Suppose we had a set S for the state/storage required by a transaction T– which specifies:

The address of a blob of data; Whether a read or write lock is required The ID of each shard where the blob of data is held

Prepare Phase Before we can execute T, we require that a set of locks L for storage S be added to the block-chain. A lo…

concurrency and async models from js, golang, rust and erlang were mentioned this morning. also, locks, mutex, channels, promises etc. everyone feel free to chime in. just starting the thread for everyone