I saw a plan for cross-sharding, named Byzantine Shard Atomic Commit (Atomix) protocol

The program has the following steps：

1 Initialize. A client creates a cross-shard transaction (crossTX for short) whose inputs spend UTXOs of some input shards (ISs) and whose outputs create new UTXOs in some output shards (OSs). The client gossips the cross-TX and it eventually reaches all ISs.

2 Lock. All input shards associated with a given cross-TX proceed as follows. First, to decide whether the inputs can be spent, each IS leader validates the transaction within his shard. If the transaction is valid, the leader marks within the state that the input is spent, logs the transaction in the shard's ledger and gossips a proof-of-acceptance, a signed Merkle proof against the block where the transaction is included. If the transaction is rejected, the leader creates an analogous proof-of-rejection, where a special bit indicates

an acceptance or rejection. The client can use each IS ledger to verify his proofs and that the transaction was indeed locked. After all ISs have processed the lock request, the client holds enough proofs to either commit the transaction or abort it and reclaim any locked funds, but not both.

3 Unlock. Depending on the outcome of the lock phase, the client is able to either commit or abort his transaction.

In this plan， If all transactions touch all the shards before committing, then the system is better off with only one shard.

What do you think of this plan or any suggestions? Let's discuss it together.More details