One of the requirements of phase 2 is the ability to move ETH quickly from one shard into another shard. Though cross-shard transactions in general

are possible through application of the usual receipt mechanism, with the protocol itself needing only to provide access to each shard's state roots to each other shard, cross-shard ETH requires a little more enshrined in-protocol activity. The reason for this is that we need to keep track of how much ETH there is in each shard, and we need an enshrined mechanism for preventing replay of cross-shard transfers.

The usual receipt-based mechanism does solve this, but it does so by having a state tree of "already consumed receipt IDs", which would be considerable complexity to add to a currently nominally stateless system. The reason this receipt ID tree is required is that we allow receipts to be consumed out of order. That is, if Alice sends a transaction from shard A -> B and then Applebaum sends a transaction from shard A -> B, it's possible that Appelbaum's transaction gets received in shard B first. This is necessary because with the gas-market-based approach for handling receipt-consuming transactions, it's possible that Alice may decide to just not pay for the transaction to finish the transfer on shard B.

So a question arises: can we replace the mechanism for handling receipts with one where receipts are processed sequentially

, so we only need one state variable for "the ID of the receipt shard B last received from shard A" that can just be incremented?

That is, every shard A maintains in its state, for every other shard B, two values: (i) the nonce of the next receipt that will be sent

from A to B, and (ii) the nonce of the next receipt that will be received

from B to A.

The answer to "who pays for it" is easy: the second half of processing a cross-shard transaction free (block producers would be required to process a certain number of receipts from other shards per block), with the rate-limiting done by charging fees on the source shard of the receipt. However, this has a major problem: what if one does a (possibly accidental, possibly intentional) DoS attack on a specific shard by sending receipts to it from all shards?

N

shards sending N

receipts each would impose O(N^2)

load on the destination shard.

To solve this, we could impose the following mechanism. Every shard is required to process up to N

receipts (eg. N = 64) in a block; if there are fewer than N

receipts from other shards to process, it can use Merkle proofs from other shards to prove this. Each shard continually relays to the beacon chain the total number of receipts it has processed, and this is used to provide an updated "gas price" for sending receipts to

that shard. For example, the gas price could be increased by 10% for each block that a shard's receipt processing queue is full, up to a maximum of N

. This ensures that at the extreme a DoS attack eventually fails to increase the length of a receiving shard's queue, so each message gets processed, but it's always possible

to send a transaction that does some minimal amount of cross-shard activity. Alternatively, shards will already need to publish their EIP 1559 gasprices to the beacon chain to process block fees; this fee can be dual-purposed for this function as well.

If we have this mechanism for sending ETH cross-shard, we could also dual-purpose it for general-purpose receipt-sending functionality, creating an enshrined guaranteed cross-shard transaction system. The main challenge is that to compute the effect

of receipts, we need someone to voluntarily provide Merkle witnesses of state. If full state is not enshrined, one cannot force this at protocol level; but what one can

do is add a requirement of the form "in order to include one of your own transactions, you must also provide witnesses for a cross-shard receipt that is in the queue".