Replay protection is the characteristic of preventing a user that signs a transaction from having that transaction executed multiple times on chain. Imagine Alice forming a transaction to send 5 TIA to Bob. Alice would want to know that the nodes she hands the transaction to won't repeatedly submit the transaction such that Bob ends up with 10 or 15TIA.

In Celestia, this is done through a nonce system. There has been some research into using an alternative system whereby each transaction is hashed, and a map is kept onchain that prevents a transaction with the same hash (i.e. the same transaction) from being executed within some defined window (say 10 blocks). If your transaction specifies a TTL of 10 blocks then you know that it will never be executed again. That way you get replay protection without the strict serialisability property of nonces.

Celestia doesn't apply this same concept for blobs. One or more users can submit the same blob multiple times. In a scenario with a centralised sequencer (i.e. a single signer), this isn't a problem as the sequencer can keep track of what blobs they have submitted and can rely on the nonce system. However, imagine the case for a decentralised shared sequencer. The network agrees on the inclusion and ordering of a set of blobs. This is aggregated into a single blob to be submitted on Celestia. The system is permissionless. Anyone can submit the blob. Now arises a coordination problem. There could be one or more nodes that submit the blob. Thus the blob gets paid for several times when it's only needed once. The same problem was present with IBC and their relayers.

I would propose, in a similar way as was done with IBC, that we introduce an antehandler in CheckTx

which has a bounded-sized map and ensures uniqueness of the blob commitment. This isn't a consensus rule and so would be non-breaking.

I am also merely predicting that this could be a problem and so would like to hear back from relevant users, specifically decentralised shared sequencers like Astria.