Intro

We have a bunch of layer 2 protocols (l2p). These protocols are build on top of layer 1. If layer 1 forks then two versions of the l2p will be created.

Replay protection exists in ethereum to prevent transactions create on one chain effecting the other chain. This is not the case on layer 2 which can lead to transactions being replayed. Here we propose a solutions.

Solution

Each layer 2 signature contains a networkID

which will be updated if the chain forks. This chain ID is defined by the eth networkID

opcode. The smart contract contains a function which will allow this value to be stored in the networkID

variable. It also stores the time stamp of each update to this variable.

Every transaction for l2p needs to include the networkID

Optimistic rollup data availability

In optimistic rollup this can be a problem because that would require you to commit to the networkID

in the data availability which can be prohibitively expensive. This can be avoided by not putting the networkID in call data. Instead inserting it in the fraud proof by reading the networkID

variable. It is important to also check the network timestamp and not revert blocks that happened before the hardfork due to a miss match in the networkID between the signatures and the message signed.

L2 replay protection

There is also the possibilty of replays between different rollups where users use the same keys + the rollup uses teh same transaction format. THerefore it makes sense to include a L2ID which is not updateable and is stored as a constant and included in the hash of the message but not the call data.

Conclusion

There seems to be some uncertainty about how the networkID opcode will be updated in the case of a fork. Some indication / expectation setting from the core devs on this point would be useful.

There are also cases where a hardfork will result in the networkID

opp code not being updated. So it seems like it make sense to explore what we should do in this situation.