Eventual Finality

Abstract.

1 Everything About L1/L2 ETH Bridging

Standard bridges are Ethereum-based smart contracts that facilitate the process of transferring Ether (and other assets) between Ethereum and an off-chain system (`a.k.a` Layer 2). In Arbitrum, a bridge contains a group of Ethereum smart contracts that each acts as a record-keeper and allows users to *deposit* and *withdraw* Ether to/from the off-chain system without taking custody of their funds. Here we briefly describe Arbitrum's asynchronous messaging systems for bridging ETH.

1.1 Transferring ETH from Ethereum to Arbitrum (`a.k.a` Deposits)

To execute *any* transaction on the Arbitrum chain, users need to put that transaction into the *inbox* contract on Ethereum. This contract is part of the Arbitrum bridge and has the responsibility of recording the calldata of transactions it receives on the Ethereum blockchain, note that this data can be used for verification of correct execution of transactions. Just like any smart contract, Inbox has a set of functions, among which `depositEth` allows users to transfer ETH to Arbitrum chain. To transfer ETH into Arbitrum via `depositEth` function, users must specify (1) the amount of ETH they are willing to bridge and (2) the amount of ETH to pay for the storage costs of storing their transaction calldata on Arbitrum (`a.k.a` base submission fee). Once called with this parameters, the bridge generates a "pre-packaged" transaction to be executed on the off-chain system. This transaction includes the caller address on Ethereum, the caller address on Arbitrum, 0 callvalue, and an empty calldata. The transaction will be stored on the Arbitrum storage, hence the base submission fee is deducted from the amount deposited and the remaining ETH value will be credited to the sender's account.

1.2 Transferring ETH from Arbitrum to Ethereum (`a.k.a` Withdrawals)

As mentioned in Section 1, Arbitrum bridge allows users to transfer ETH from Ethereum without holding those ETH in escrow. Once bridged, these ETH can be transferred back to Ethereum by sending an Arbitrum transaction to the *ArbSys*; Arbitrum pre-compiled contract that lives at the same address on every Arbitrum chain (`address(100)`) and provides system-level functionalities. Users

can withdraw ETH back to Ethereum by calling a payable `withdrawEth` method and providing (1) their Arbitrum address from which ETH is being transferred and (2) the amount of ETH they are willing to transfer to Ethereum. Once called, `withdrawEth` publishes an Arbitrum transaction which will be **only** executed after the dispute window is over. Unlike Ethereum, Arbitrum transactions do not have instant finality *i.e.,* they can only be finalized on Ethereum after the so-called dispute period is expired.

This time window allows Arbitrum to provide better security guarantees against censorship attacks and dispute wrong assertions coming from malicious validators, if any. Once the dispute period is over, the withdrawal transaction (together with all other outgoing messages) will be Merklized and entered in the *Outbox* contract on Ethereum; a member of the bridge that manages the messages originating from Arbitrum chain (*e.g.,* withdrawals). Only if the dispute period is expired, users can execute their withdrawal transactions by calling a certain *Outbox* method and providing their transaction ID

2 Research Problem: Slow Withdrawals

As mentioned in Section 1.2, because Arbitrum execution happens optimistically, transactions can be considered final only after the dispute period (7 days) is over. This imposes a significant delay for any message originating from Arbitrum to Ethereum, especially withdrawal transactions. Funds will be only released from the off-chain system after the withdrawal transaction is valid on Ethereum. In this paper, we propose three solutions that allow users to sidestep the confirmation delay entirely when withdrawing assets from Arbitrum t o Ethereum.

3 Potential Designs and Solutions

Describe 3 solutions we have the slides for

4 Evaluation Framework

Come up with the evaluation framework to compare the designs provided in he previous section, this will (hopefully) help us to choose one out of 3 to implement

5 Economics

How much would you pay for a tradable ticket? Knowing that they are weird assets (+ non-fungible)

Carrying costs + fees + ..

6 Implementation

We have to find a market to buy/sell these tradable claims; order-book? AMM?
We need liquidity (note that they are non-fungible)
11:40 Dec 20