Fast and Furious Withdrawals from Optimistic Rollups

Abstract. Abstract goes here.

Keywords: blockchain · Ethereum · layer 2 · rollups · bridges

1 Introductory Remarks

Ethereum-compatible blockchain environments, called Layer 2s (or L2s), have demonstrated an ability to reduce transaction fees by 99–99.9% while maintaining similar guarantees over the integrity and availability of the blockchain. The subject of this paper concerns one sub-category of L2 technology called an optimistic rollup. The website L2 Beat attempts to capitalize all tokens of known value across the top 25 L2 projects, and finds that two optimistic rollups—Arbitrum and Optimism—respectively account for 50% and 30% of L2 value—$4B USD at the time of writing.[1]

We will describe the working details of optimistic rollups later in this paper but here are the main takeaways. Currently, rollups are faster and cheaper than Ethereum itself, however each Layer 2 is essentially an isolated environment that cannot instantly and trustlessly interact with accounts and contracts that are running on either L1 or other L2s. An optimistic rollup project will typically provide a smart contract, called a validating bridge [], that can trustlessly move ETH (and other tokens) between L1 and its own L2. It does this by locking the ETH in the L1 contract and later releasing it, on request, according to who its new owner is on L2 at the time of the request. If finalized, the ETH will be destroyed on L2 and will be released by the bridge on L1.

Due to how optimistic rollups convince the bridge contract on L1 of who the current owner of withdrawn ETH is on L2 (explained below), the bridge has to first wait a period of time called the dispute window. The current default is 7 days in Arbitrum Nitro, however the presence of disputes can extend it. The bottom line is that users have to wait several days to withdraw ETH and other tokens from an optimistic rollups.

Contributions. In this paper, we compare several methods—atomic swaps and tradeable exits—for working around this limitation. While we argue workarounds cannot be done generally, some circumstances allow it: namely, when the withdrawn token is liquid, fungible, and available on layer 1 and the withdrawer is willing to pay a fee to speed up the withdraw. While these techniques work easily between human participates that have off-chain knowledge, such as the valid

[1] L2 Beat, Oct 2022.

state of the L2, it is harder to make them compatible with L1 smart contracts that have no ability to validate the state of L2. We propose a solution using tradeable exits and prediction markets to enable an L1 smart contract to safely accept withdrawn tokens before the dispute period is over. Finally, we modify the most popular optimistic rollup, Arbitrum Nitro, to support our solution and provide measurements.

2 Background

Inbox. Roll-ups have emerged as a workable approach to reducing fees and latency for Ethereum-based decentralized applications. In a roll-up, transactions are recorded in an Ethereum smart contract (called the inbox) maintained by the roll-up system, but the transactions are not executed on Ethereum. Instead, they are executed in a separate environment off the Ethereum chain, called the L2. This external environment will operate by different rules designed to reduce fees, increase throughput, and decrease latency.

Outbox. Occasionally (*e.g.,* every 5 minutes), validators on the L2 will record a checkpoint of all contracts and accounts in the L2 and their current state in a smart contract on L1, called the outbox. Anyone with only a view of L1 can validate that the sequence of the transactions recorded in the inbox produces the asserted checkpoint in the outbox, however asking Ethereum to validate this be equivalent to just running the transactions on Ethereum in the first place. The key breakthrough is that the assertion will be posted with evidence that the checkpoint is correct.

Optimistic vs. zk-rollups. In practice, two main types of evidence are used. In zk-rollups,[2] a succinct computational argument can be checked by Ethereum for far less cost than running all the transactions. However the proof is expensive to produce. In optimistic rollups, the assertions are backed by a large amount of cryptocurrency (acting as a fidelity bond). The correctness of the assertion can be disputed using Ethereum and Ethereum itself can decide between two (or more) disputes for far less cost than running all the transactions, and reallocate the fidelity bond to the whoever making the correct assertions. If an assertion is undisputed for a window of time (*e.g.,* 7 days), the assertion is considered final.

Bridge. A final piece of layer 2 infrastructure is a bridge, which can move ETH, tokens, NFTs, and even arbitrary messages, between layer 1 and layer 2. In this paper, we consider the case of a bridge for ETH (while discussing other kinds much later in Section **??**). If Alice has ETH on Ethereum, she can submit her ETH to a bridge smart contract on Ethereum which will lock the ETH inside of it, while generating the same amount of ETH in Alice's account inside the layer 2 environment. Since any deviation from this will result in an incorrect

[2] zk stands for zero-knowledge, a slight misnomer: succinct arguments of knowledge that only need to be complete and sound, not zero-knowledge, are used.

checkpoint which is ultimately checkable by Ethereum, the bridge does not need to be trusted (such bridges are called validating bridges []). A set of transactions on layer 2 might see the ETH move from Alice's account to Bob's account on Layer 2. Bob is now entitled to draw down from layer 2 to layer 1. He submits a withdraw request, which reduces his Layer 2 balance and ends up in the next asserted checkpoint on Layer 1. Optimistically, the checkpoint is undisputed and 7 days later it is finalized. Bob can now ask the bridge to release the ETH by demonstrating his withdraw is contained in the finalized checkpoint.

2.1 Related Work

Arbitrum is described at *USENIX Security* []. Gudgeon *et al.* provide a systemization of knowledge (SoK) of Layer 2 technology (that largely predates rollups) [], while McCorry *et al.* provide an SoK that covers roll-ups and validating bridges []. Some papers implement research solutions on Arbitrum for improved performance: decentralized orderbooks [] and secure multiparty computation []. Further academic work on optimistic rollups and bridges is largely missing at this time, but we anticipate it will become an important research area. Other related topics are atomic swaps and prediction markets. Too many papers propose atomic swap protocols to list here but see Zamyatin *et al.* for an SoK of the area (and a new theoretical result) []. Decentralized prediction markets proposals predate Ethereum and include Clark *et al.* [] and Truthcoin []. Early Ethereum projects Augur and Gnosis began as prediction markets.

3 Proposed Solution

- Optimistic rollups are being used
- Problem: multi-day (*e.g.,* seven) window for withdrawal
- Why is this a problem? Others are solving this problem (alternative bridges with a TTP). Some examples include: Speculators want to move fast, voting in a DAO (red tape), sell them on L1 or another L2, DApp access on L1 (trading if efficient on L2 but you need to do something on L1).
- Disadvantage relative to zk-rollups
- Solution: scope to liquid tokens (ETH and ERC20), open problem: NFTs or other non-substitutable tokens.
- Solution: (1) tradeable exits; (2) market to trade; (3) guaranteed exit (buyer runs ArbOS validator); (4) prediction market solution to guarantee exits to non-validating entities (importantly includes smart contracts)
- Testing: we implemented (1) and (3); for (2) and (4), use your favourite DeFi project.

3.1 Design Landscape

- Landscape: atomic swaps
- Landscape: Change the bridge instead of using a third party contract (reason: too late once you withdrawal)

3.2 Fast Bridge

- Allow trading of exits (atomic unit) -> track most recent owner (constant time) in outbox. Authorization (only current owner can transfer) -> on execute, check for current owner.
- Self-insurance (staking a fidelity bond) (you need liquidity)
- Prediction market: someone else will insure it

3.3 Implementation

- Arbitrum's Nitro. Bridge: inbox, sequencer, outbox.
- Implemented a market, following is the gas cost related to the market: gasUsed for opening a market on an exit: 328,029 gasUsed for transferring the exit to the market: 86,701 (it was the first transfer so a bit more expensive that the 2nd, 3rd,.. see below) Gas cost for submitting the Bid for when the Bid is greater than ask -> trade occurs and settles in one single submitBid tx: 105,287 Gas cost for execution is: 92,148
- Modify outbox to allow tradeable exits
- Modify the Nitro codebase (arbnode and validator) to shrink the fraud proof window from 7 days to 1 minute: (1) Modify the confirmPeriod variable in the arbnode/node.go file, (2) modify the MakeAssertionInterval variable validator/staker.go
- To make the bridge prediction market friendly: Modify the outbox: (1) added a Mapping that maps the proposed arbitrum block number (also known as assertion and node) to the pending state. (2) added a function which accepts a proposed block number and adds it to the pending assertions mapping. Modify the rollupcore: The validator acts through the RollupCore.sol contract when making an assertion by calling a createNewNode() function. We modifies this function so that every time a node is created by the validator it's also added to the outbox's pending assertions mapping (outbox.addToPendingAssertions(latestNodeCreated()))
- L1 gas costs: new function (transferSpender) : First Transfer: 1) Alice withdraws ETH from L2 2) She transfers her exit to Bob transferSpender in this case costs : gasUsed : 85,945 Second Transfer: 2) Now Bob transfer his exit to Carol transferSpender in this case costs : gasUsed : 48,810 Third Transfer: 2) Now Carol transfer his exit to Nancy transferSpender in this case costs : gasUsed : 48,798
  (difference of two mappings)
- L1 gas costs: execute the exit: 91,418
- Unit tests: say something
- What happens when an assertion fails? (pro: ticket fails with assertion, better for prediction markets (betting on assertion which is a batch of exits); ticket passes even if assertion fails, sounds better (caveat: probably won't happen)
- Challenges: SDK, where to change, unit test failed assertions (good assertion, bad assertion where withdrawal is ok, bad assertion where the withdrawal is problematic)

- Expose assertion failures/successes to external contract (submit ID for assertion, get back status: pending, finalized, or discarded). Write down the SDK call. (what happens to a failed assertions???)

4 Discussion

4.1 Where to Solve Problem?

- Exit to a third party collateral contract, implements trading

4.2 Withdrawal Format

- Divisible exits
- ERC20 / ERC721? -> allowances or not?

4.3 Non-Substitutable Withdrawals

- Illiquid tokens
- NFTs (substitute for ETH -> support)
- Messages (oracle updates, read calls, any L2-to-L1 message...) (still offer insurance)

4.4 Assertion Failures

- does exit go away with the assertion or not? Pros/cons

4.5 Markets

- Can't use an AMM
- Can't run out
- Simple auction
- Divisible exits
- Exit pools